

Comparative Study of Software Process Models for a Healthcare Management System

Analyzing the Software Development Lifecycle



**Name:** Aashna Mathias

**Course title:** Introduction to Software engineering

**course code:** is1103-1

**Date:** 14-02-2025

ABSTRACT

Keywords: Healthcare Management System, SDLC, Waterfall Model, Incremental Model, Spiral Model, Risk Management, Software Development

A Healthcare Management System (HMS) is necessary for scheduling, invoicing, patient data management, and regulatory compliance. Therefore, selecting the appropriate Software Development Life Cycle (SDLC) model is crucial and depends on risk management, adaptability, and efficiency. This report compares Waterfall, Incremental and Spiral models in context of HMS development and based on functional requirements, risk management and adaptability to regulatory changes. The result shows that iterative development approach is the way to go for the dynamic needs of healthcare industry.

Documentation: The complete project and analysis can be accessed at - <https://github.com/aashnaaaaa>

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
|  | Abstract | i |
| **1.** | Introduction | 1 |
| **2.** | Overview of Healthcare Management System | 1 |
| **3.** | Overview of Software Process Models | 2 |
|  | 3.1 Waterfall Model |  |
|  | 3.2 Incremental Model |  |
|  | 3.3 Spiral Model |  |
|  | 3.4 Comparison of Models for Healthcare Management System |  |
| **4.** | Requirements Engineering for Healthcare Management System | 9 |
|  | 4.1 Functional Requirements |  |
|  | 4.2 Non-Functional Requirements |  |
|  | 4.3 Requirements Validation Strategy |  |
|  | 4.4 Potential Challenges in Requirements Validation |  |
| **5.** | Conclusion | 12 |
| **6.** | References | 12 |

**1. Introduction**

A Healthcare Management System (HMS) is a software solution aimed at optimizing hospital operations, overseeing patient records, scheduling appointments, and ensuring the secure handling of medical data. Selecting the appropriate Software Development Life Cycle (SDLC) model is essential for fulfilling both functional and non-functional requirements while effectively managing risks, time, and costs.

**2. Overview of Healthcare Management System**

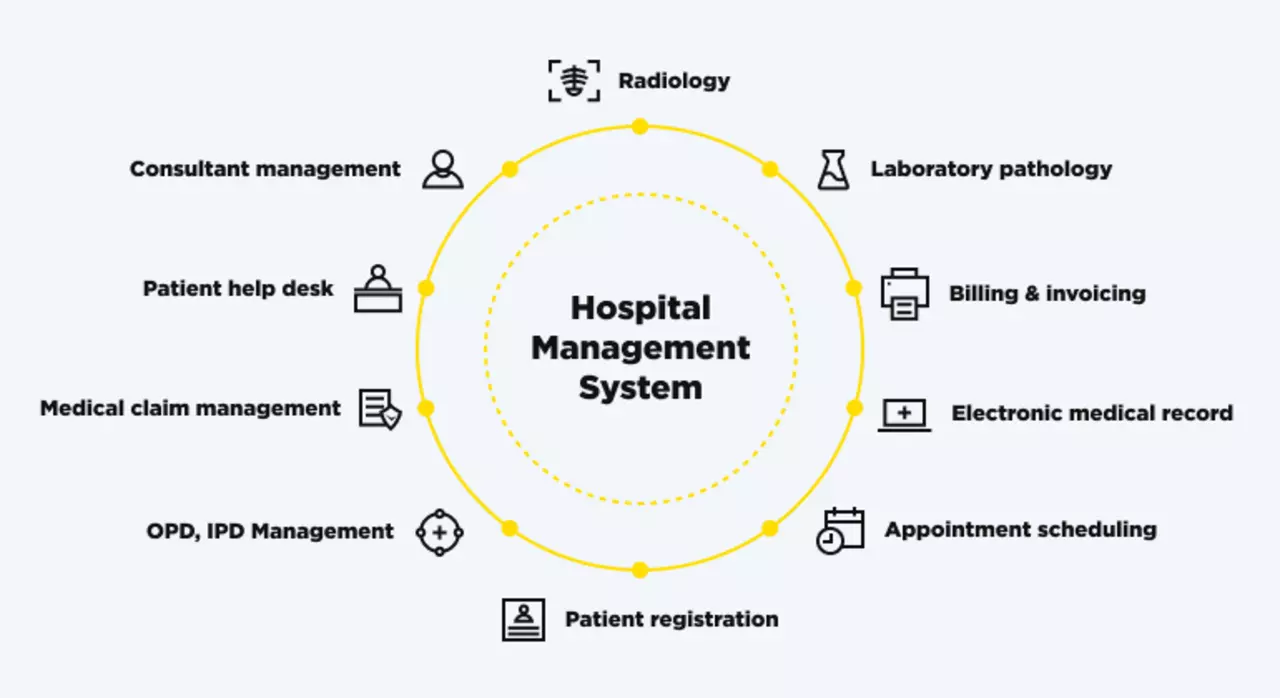


Figure 1: A view of Healthcare Management System

Healthcare is among the fastest-growing sectors in both developed and emerging economies. The integration of e-healthcare solutions has significantly contributed to this growth by utilizing the internet and digital tools to enhance communication and information management. Electronic medical records, virtual networking, cost savings, improved privacy and improved doctor-patient relationships are some advantages of e-healthcare adoption (Williams 2021).

Data security and access management are major issues in healthcare administration. In accordance with legal requirements such as HIPAA role-based access control (RBAC) models are frequently used to guarantee that medical records are only available to authorized personnel (Williams 2021). Furthermore, a service-oriented architecture (SOA)-based distributed e-healthcare system enhances interoperability amongst different stakeholders such as patients, doctors, nurses and pharmacists (Kart et al. 2020). These systems make use of open standards such as WSDL, SOAP and XML to facilitate smooth data sharing across various platforms while maintaining performance and security (Kart et al. (2020).

**3. Overview of Software Process Models**

**3.1 Waterfall Model**

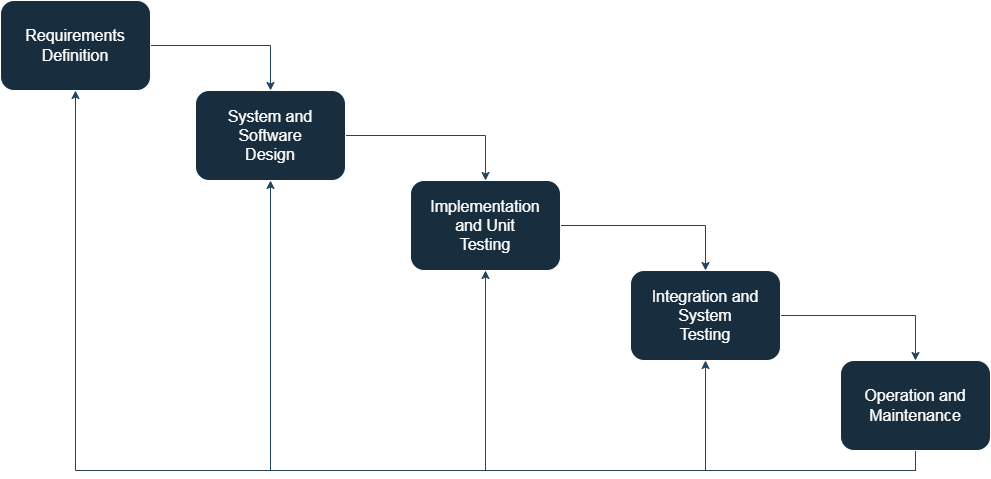


Figure 2: Pictorial Representation of Waterfall Model

**How a Healthcare Management System can be developed using the Waterfall model:**

* **Phase 1: Requirements Definition**

During this first step, system requirements are gathered and systematically documented and defined at once. This means patient record keeping, appointment management, EMR, billing, user roles access control (doctors, nurses, administrators, etc). The Waterfall model does not permit easy changes later on, hence this stage requires extensive discussions with subject matter experts, compliance team, and IT team about what exactly needs to be done, ensuring all needs have been considered at this point itself. Exhaustive documentation is prepared to serve as a foundation for the next phases.

* **Phase 2: System & Software Design**

When requirements finalize, the architecture of the system is formed. This covers everything from database schemas for patient information, network architectures to ensure secure transmission of sensitive data, and integration points for connecting with other healthcare services, such as insurance companies and laboratory results. During this phase, security protocols (HIPAA compliance, data encryption, etc.) are also defined at this stage. Since Waterfall does not accommodate mid-development changes easily, all technical things such as server infrastructure, APIs, data models, etc need to be thought out before moving forward.

* **Phase 3: Implementation & Unit Testing**

In this phase, developers start to write the code according to the design which was approved earlier. The developers start coding in modules. For eg, consider module 1 to be user authentication, patient registration and medical history tracking and module 2 to be appointment scheduling and doctor availability and module 3 to be billing and insurance claims processing. All these modules are started and finished simultaneously following a linear process. Each component is tested individually to ensure functionality, but now, at this point, no major changes to the system's core design can be done. Since healthcare applications require high reliability, extensive unit testing is performed to catch potential issues early.

* **Phase 4: Integration and System Testing**

After that, once all the modules are developed, it is integrated and tested as a complete system. This phase ensures that different components, such as scheduling, patient records, and billing, work together smoothly. Rigorous testing is conducted (which includes functional to ensure that every feature does what it is supposed to. For eg, performance testing to identify whether response times are fast enough, even with a lot of data and requests coming in; and security testing to ensure sensitive patient data is not compromised ).However, if major flaws are found, resolving them can be time-consuming because of the rigid structure of the Waterfall model.

* **Phase 5: Deployment and Maintenance**

The completed HMS is now deployed in hospitals, clinics, or healthcare networks. The system is now live, allowing medical staff and administrators to start using all its features right away. However, any issues or improvements require formal change requests and extensive rework. Because of this, introducing new updates such as regulatory compliance changes or feature enhancements can take a long time.

**Suitability for Healthcare Management System:**

We can see that the Waterfall model is useful when all functional and non-functional requirements are described before-hand. Because the healthcare system wants security, performance, and usability, this version of model guarantees structured documentation however lacks flexibility.

The Waterfall Model struggles with requirement changes because in this, once a phase is completed, changes require restarting from the development process, making it unsuitable for dynamic healthcare environments.

The Waterfall model is cost-effective if all requirements are defined already but can cause massive delays and rework charges if changes arise later.

Hence, we can come to a conclusion about waterfall model that it follows a structured approach, and this makes it not suitable for a Healthcare Management System, as healthcare technology evolves rapidly. Frequent updates, regulatory changes, and the need for continuous improvement are some examples to prove that a Healthcare Management System needs more flexible models.

**3.2 Incremental Development Model**

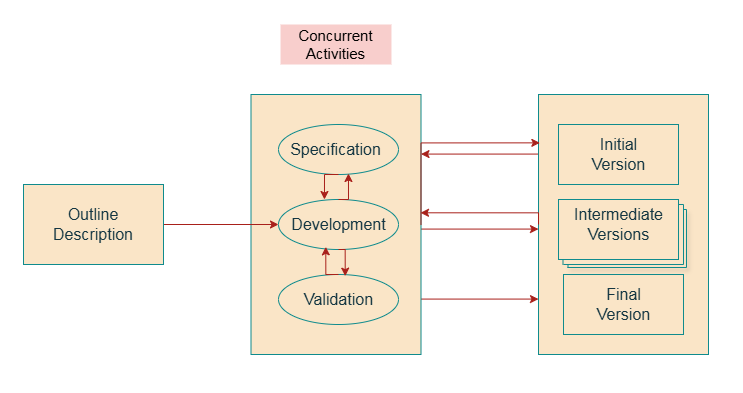


Figure 3: Pictorial Representation of Incremental Development Model

**How a Healthcare Management System can be developed using the Incremental Development Model:**

* **Phase 1: Requirements Definition and Planning**

The first phase includes obtaining high-level requirements for the Healthcare Management System. We assess and determine the important capabilities, such as patient registration, appointment scheduling, and basic record-keeping. Non-functional needs include security, scalability, and compliance with healthcare legislation (for example, HIPAA). A certain collection of characteristics is the emphasis of each increment that makes up the system. For instance:   
Step 1: Registering patients and managing their basic medical records.   
Step 2: Setting up appointments and sending out reminders.   
Step 3: Integration of billing and insurance.   
Step 4: More sophisticated reporting and analytics.

* **Phase 2: Design and Development of Increment 1**

Now, the first increment is designed and developed. For example, the patient registration module is built with a user-friendly interface and secure data storage. The architecture is made in such a way that it can accommodate future increments, ensuring scalability. Unit testing is performed to ensure the module works as intended.

* **Phase 3: Deployment and Feedback for Increment 1**

The first increment is deployed to limited number of users (For e.g., a single hospital or a clinic). The feedback from customers is gathered, and any problems will be fixed. This early deployment allows stakeholders to see visible progress and offer suggestions for next increments thanks to this early deployment.

* **Phase 4: Developing Subsequent Increments Iteratively**

Every new increment is developed, tested, and released in response to user feedback and evolving requirements. For instance, Increment 2 adds the module for scheduling appointments, while Increment 3 integrates billing and insurance. Non-functional needs, such as security and performance, are continuously evaluated and improved.

* **Phase 5: Final Integration and System Testing**

After all the increments are completed, the system is put through final integration testing to ensure that all modules function as a whole. Compliance inspections, security testing, and performance testing are done to ensure the system meets changing needs and healthcare standards.

**Suitability for Healthcare Management System:**

The Incremental Model allows essential functionalities like patient history and appointment scheduling to be performed first, and then other functions like billing or analytics to be used subsequently. The iterative approach helps adapt to evolving healthcare requirements.

The Incremental Model handles changes more effectively by including user feedback in every cycle so that modifications can be made in later stages. If there is a new compliance requirement, it can be added in later increments without replacing the whole system.

The Incremental Model could have greater up-front costs due to multiple runs but lowers future costs by permitting early release of key features and fixing problems along the way.

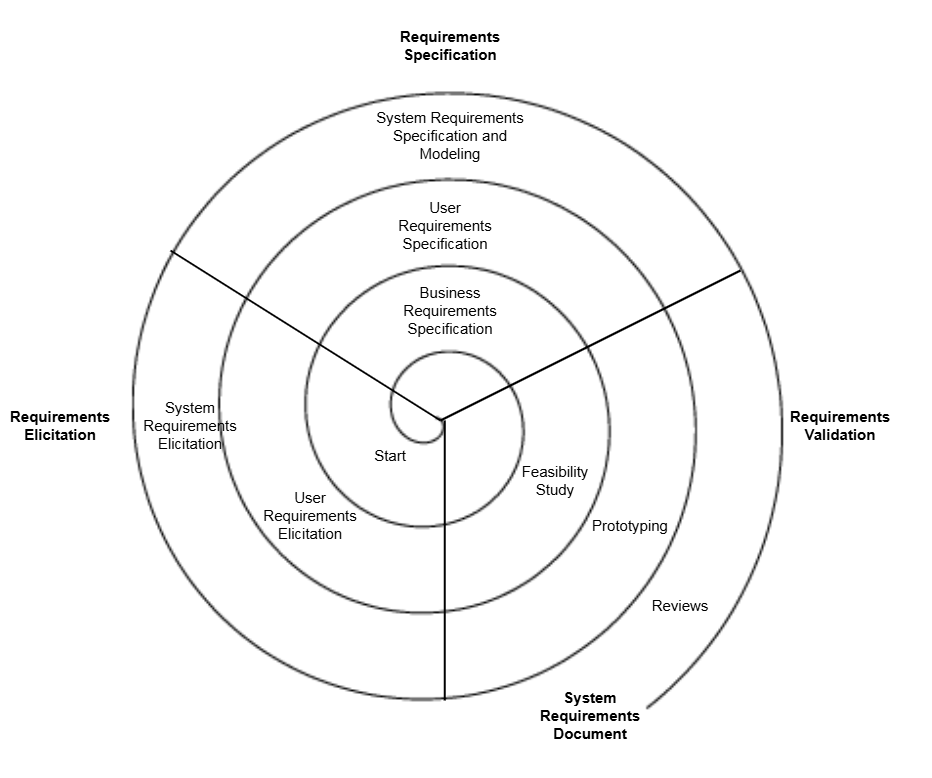
**3.3 Spiral Model**

Figure 4: Pictorial Representation of Spiral Model

**How a Healthcare Management System can be developed using the Spiral model:**

* **Spiral 1: Initial Planning and Risk Analysis**

The first spiral focuses on understanding the core requirements and identifying potential risks. Key risks for an HMS include data security, compliance with healthcare regulations, and integration with existing systems. A prototype for patient registration and basic record-keeping is developed to validate feasibility.

* **Spiral 2: Risk Assessment & Prototype Development**

This phase creates a working prototype in order to validate basic features. It will involve user authentication for the protection of private medical records, data encryption, and access control systems. Initial compliance testing ensures that they conform to healthcare standards. Risk analysis targets the performance risks, security vulnerabilities, and usability of the system. Testing a small version of the system will expose key shortcomings before large-scale development.

* **Spiral 3: Development and Testing of Core Features System**

The first phase included patient record management, appointment scheduling, and billing functionalities. An iterative development process is followed: build-test-validate each new feature before going on to the next. A subsequent round of risk assessment is done to ensure the large-scale data processing, data securing, and integrated communication with external services, such as insurance providers and laboratories. Usability testing is also carried out to confirm that the medical staff can operate the system with ease.

* **Spiral 4: Enhancements, Integration, & Final Testing**

In this case, additionally, more features such as insurance management, pharmacy integration, and an AI diagnosis support are added. More intensive testing procedures are done on the CMS software for performance (the way it performs under intense user loads), security (and its vulnerabilities), and regulatory compliance issues (such as adjustments in accordance with regulatory changes in the industry it is working with). User feedback in this situation is an iterative process that enables improvements on system usability and optimized workflow.

* **Spiral 5: Deployment and Continuous Refinement**

Once the HMS has been completed, the hospitals and other healthcare institutions will have access to it. After this, the application is subjected to constant monitoring and maintenance to fix security issues, improve performance, and implement new features driven by the latest healthcare needs. Future versions can incorporate such additional features as telemedicine, predictive analytics, and automated patient monitoring into its cycles with ease, as new regulations and medical technologies change.

**Suitability for Healthcare Management System:**

The capabilities of Spiral Model in handling complex functional and non-functional requirements, while focusing on security, scalability, and compliance with healthcare regulations such as HIPAA, can hardly be underestimated. Continuous risk assessment is included in the model to ensure that issues of compliance are taken care of at every phase of development.

The Spiral Model is excellent for health application wherein security, compliance, and scalability are of utmost importance. The model essentially introduces risk early in order to identify potential risks and, through limiting their impact, to mitigate the risk.

The costliest of these is the Spiral Model where the continuous changes due to constant risk management are a large part. Certainly, it makes more sense to think of such investment in large health systems where legal compliance and any breach must be avoided.

|  |  |  |  |
| --- | --- | --- | --- |
| **Criteria** | **Waterfall Model** | **Incremental Development Model** | **Spiral Model** |
| **Functional & Non-Functional Requirements** | Changes are challenging, and all criteria are specified up front. | Requirements provide for flexibility when they change gradually. | Core requirements are established early on and are subject to change. |
| **Risk & Change Management** | Change is hard, and risk management is delayed. | Changes are simpler and risks are controlled gradually. | Constant evaluation of risks and management of change. |
| **Time & Cost Constraints** | Rigid phases might cause time and expense to increase if changes occur. | Higher initial costs but cheaper long-term costs because of the flexibility. | Increased initial expenses as a result of continuous improvement and risk control. |
| **Suitability for Healthcare Management System** | Not suitable because of its rigidity and inability to adapt to changes. | Suitable because it fits well with feedback integration and changing features. | Best suited for sophisticated requirements including continuous risk control. |

**3.4 Comparison of Models for Healthcare Management System**

**4. Requirements Engineering for Healthcare Management System**

Healthcare Management Systems (HMS) demand intricate functional and non-functional specifications to achieve operational effectiveness while safeguarding data security and adhering to healthcare legal standards. The process of requirements engineering consists of detecting and evaluating demands which undergo verification to construct systems that achieve both reliability and scalability. The essential functional and non-functional needs for a successful HMS are listed in this section.

**4.1 Functional Requirements**

**1) Role-based access control and user authentication**

Ensures that only individuals with permission, including administrators, physicians, and nurses, can access particular data and features. This guarantees adherence to laws such as HIPAA and safeguards private patient data.  
  
**2) Management of Patient Records: Retrieval, Storage, and Modification**

This function is critical within the system, providing facilities for storing, retrieving, and modifying patient details such as diagnosis, treatment plans, and medical history. Any illness should be handled effectively, and accurate and easy access to records is of utmost importance.  
  
**3) Setting Appointments and Sending Reminders.**

Allows patients to schedule appointments and be reminded to visit the health care facility, thus enhancing patient flow and lowering patient no-show rates. At the same time, it paves the way for staff members and physicians to properly manage their schedules.  
  
**4) Invoicing and Insurance Management.**

Manage the processes involved in payment, insurance claims, and patient billing. It guarantees that all healthcare systems will continue to function and improves operational work.

**5) Portal for Patient Self-Service:**

With this web-based/mobile-based interface, the patients will be able to schedule, review medical records, request medicines, and monitor health metrics (blood pressure, weight, etc.). It reduces the physician's administrative burden while markedly improving patient interaction.

**6) Decision Support:**

Based on available technology, all system interactions, including patient record updates, login attempts, and data modifications, have an audit trail feature. Auditing trail ensures accountability, confidentiality, and adherence to regulatory requirements such as HIPAA within the healthcare environment.

**4.2 Non-Functional Requirements**

**1) Security (HIPAA Compliance and Data Encryption)**

This ensures that unauthorized access to, and breaches of, patient data can never happen. The high standards for privacy and security dictated by HIPAA make its compliance paramount in any health system.

**2) Quick Data Retrieval and Lower Downtime**

The ease and speed with which a medical professional can access patient data is very important for timely decision-making on treatment. Data being mined slow may cause delays in treatment that can risk patients or worsen patients.

**3) User-Friendly UI for Medical Staff**

The system should be user-friendly enough for the medical staff, most of whom are not computer experts. This implies an interface with the least amount of work needed from the staff will be user-friendly, thus minimizing error and ensuring staff attention and focus remains on patient care rather than having to struggle with the system.

**4) Support Increasing Patient Data and User Load**

As the healthcare system grows, it will also provide flexibility to manage growing patient data volumes and user enrolment. Without compromising functionality, scalability allows for a moderate growth in the number of patients, healthcare professionals, and features.

**5) Disaster Recovery and Backup**

An effective disaster recovery strategy, with periodic backups, is necessary to protect patient information from accidental erasure, hardware crash, or cyberattacks. This will provide for minimal disruption to services and data loss.  
  
**6) Localized and Multilingual Support**

In multicultural healthcare settings, particularly in areas of varied languages and cultural customs, the HMS has to accommodate more than one language and be adjustable to local legislations so as to be made accessible to healthcare professionals and patients alike.

**4.3. Requirements Validation Strategy**

The intricate process of requirements validation ensures that the Healthcare Management System (HMS) meets user needs while adhering to legal standards and functioning reliably before deployment. The identification of issues at initial stages reduces costs while boosting system efficiency. The validation of HMS requirements occurs through the application of these techniques.

* **Stakeholder Review:** Doctors, nurses, IT staff, and legal experts take part to make sure the system meets everyone's needs.
* **Testing:** The team checks how well the system works how secure it is how it performs how it fits with other systems, and how different roles can access it.
* **Regulatory Compliance:** To ensure that all requests are met, medical professionals, nurses, IT staff, and attorneys take part in the stakeholder review process.
* **Emergency Readiness Testing:** Functional, security, performance, interoperability, and role-based access testing are all included in this category.
* **Data Migration Testing:** Adherence to Regulations: Regular audits ensure adherence to evolving healthcare laws, GDPR, and HIPAA.
* **User Training & Feedback:** Users are continued to be trained by the team to collect their input to boost system adoption.
* **Digital Twin Simulation:** The team creates a virtual replica of the system to experiment with real-world scenarios without affecting the live environment.
* **Accessibility Testing:** The team checks if people with poor eyesight and older folks can use the system with help from special tech tools.
* **Multi-Tier Data Security Checks:** The team looks at how well the system encrypts data, keeps track of who uses it, and makes sure people prove who they are in more than one way.

**4.4 Potential Challenges in Requirements Validation**

Here are some of the key challenges that are faced during the requirements validation:

* **AI Inaccuracies –** Inaccurate AI predictions put patient safety at risk with diagnostic errors, wrong treatment plans and delayed treatment.
* **Regulatory Changes –** Healthcare regulations like GDPR and HIPAA change frequently and need continuous compliance audits.
* **Cyber attacks & Data Breaches –** Defend against malware, ransomware and unauthorized access to prevent cyber attacks and data breaches.
* **Legacy System Integration –** Because it increases the danger of data loss and corruption, integrating legacy systems is a nightmare when transferring historical patient records.
* **Interoperability –** Interoperability issues create problems for third-party technologies to talk to healthcare systems.
* **Scalability –** The system needs to handle increasing patient data and user numbers without performance degradation.
* **User Resistance to AI –** Doctors are resistant to AI recommendations which is a barrier to adoption.

**5. Conclusion**

So far, we have found the incremental model is perfect for a healthcare management system. It allows for incremental improvement, and continuous input. Easier to adapt to changing medical requirements and regulatory changes.

The Spiral Model is better for high risk situations because it does risk assessment at every stage. It reduces system failures, security flaws and compliance issues. When dealing with patient data this is critical.

In general, the Spiral Model is better for strict security and regulatory requirements, the incremental model is good for most healthcare systems.

**6. References**

* Williams, J. (2021). *Role-Based Access Control Models for E-Healthcare Systems.* Florida A&M University, Department of Computer and Information Sciences.
* Kart, F., Miao, G., Moser, L. E., & Melliar-Smith, P. M. (2020). *A Distributed e-Healthcare System Based on the Service-Oriented Architecture*. University of California, Santa Barbara.
* TeleCRM. (n.d.). *Healthcare Management System*. Retrieved from <https://telecrm.in/blog/healthcare-management-system/>
* Sprinto. (n.d.). *Compliance Audit*. Retrieved from <https://sprinto.com/blog/compliance-audit/>