Class #9

03. Designing Software architectures

Software Architectures
Master in Informatics Engineering

Cláudio Teixeira (claudio@ua.pt)

Agenda

- Designing Software Architectures
  - Systematic approaches:
    - architecture centric design method (ACDM)

Referências extra - Design Patterns & more

- design_pattern_cheatsheet_v1.pdf
- https://medium.com/cp-massive-programming/design-patterns-cheat-sheet-list-e9a55d82de5d
- https://learning.oreilly.com/library/view/design-patterns-elements/0201633612/
- https://learning.oreilly.com/library/view/software-architecture-the/9781492086888

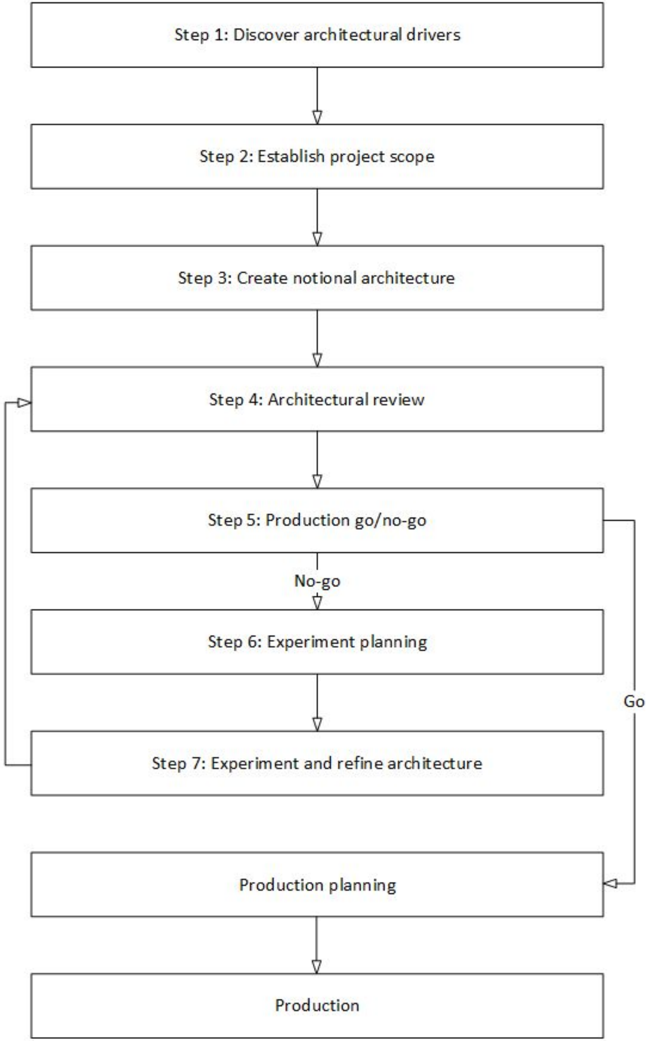
Designing Software Architectures:

Systematic approaches:

Architecture Centric Design Method (ACDM)

Architecture Centric Design Method (ACDM)

The architecture-centric design method (ACDM) is **an iterative process** used to design software architectures. It is a lightweight method with a **product focus** and seeks to ensure that the software architecture maintains a balance between business and technical concerns.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

Healthcare Patient Management System (HPMS) - Overview

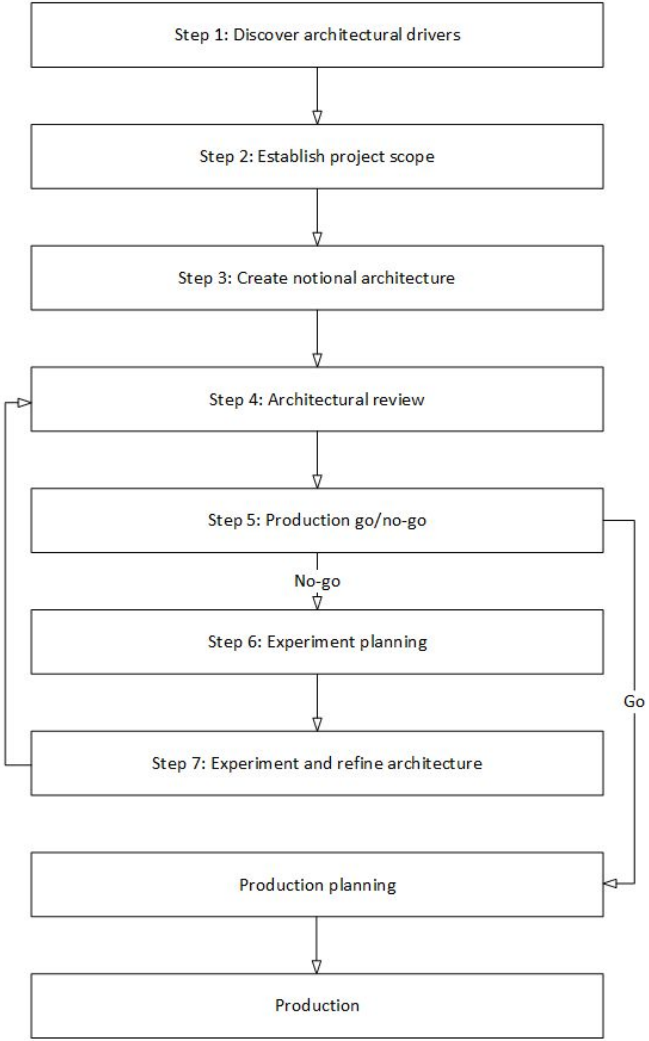
The Healthcare Patient Management System (HPMS) aims to innovate the management of patient-centric information and healthcare delivery.

It is intended to integrate seamlessly with existing healthcare infrastructures, facilitating comprehensive management of patient records, appointments, and treatment protocols.

1 - Discovering architectural drivers

Meet with stakeholders to determine the architectural drivers, which include **design objectives**, primary **functional requirements**, **quality attribute scenarios**, **constraints**, and **architectural concerns**.

Define prioritization of quality attribute scenarios.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Go
Step 7: Experiment and refine architecture
Production planning
Production

HPMS
Step 1

Functional Requirements (FRs)
FR1: The system must allow healthcare professionals to create, read, update, and delete (CRUD) patient records.
FR2: The system must manage appointment scheduling between patients and healthcare professionals.
FR3: The system must support the recording and tracking of patient treatments and medication.
Quality Attributes (QAs)
QA1: Security - Patient data must be protected from unauthorized access. Compliance with HIPAA (Health Insurance Portability and Accountability Act) for patient privacy is mandatory.
QA2: Scalability - The system must support a growing number of users (patients and healthcare professionals) and records without degradation in performance.
QA3: Usability - The interface must be user-friendly, ensuring that healthcare professionals can efficiently manage patient information and appointments.
Constraints (C)
C1: The system must be developed within a 9-month timeframe to meet regulatory requirements.
C2: Integration with existing healthcare systems for seamless data exchange is required.
C3: The system must be deployable both as a web application and on mobile devices to ensure accessibility.
Principles (P)
P1: Interoperability - Adherence to healthcare data exchange standards, such as HL7 or FHIR, to ensure compatibility with other healthcare systems.
P2: Modularity - The system architecture should be modular to facilitate easy updates and maintenance.
P3: Continuous Delivery - Adoption of continuous delivery practices to allow for incremental updates and rapid deployment of new features.

2 – Establish project scope

Review architectural drivers established in Step 1.

Consolidate information gathered and remove duplicate architectural drivers

If any information unclear, missing or incomplete, should be completed in this step

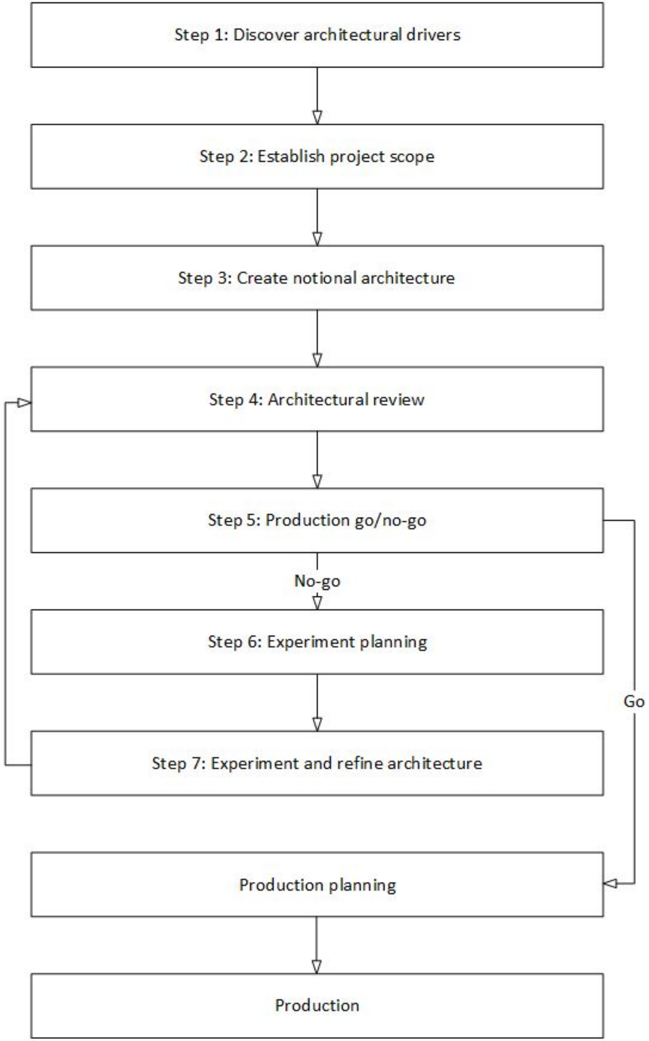
- applies to both drivers and requirements and software attributes scenarios that are not measurable or testable.

If needed, talk to stakeholders until information complete.

Define the boundaries of the project, including what will and won't be included in the system. This step helps to manage stakeholders' expectations and focuses the architectural efforts on the core objectives of the project.

Output:

Revised list of Functional and NF Requirements, defined attribute scenarios and quality metrics, timelines, development boundaries, etc.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

HPMS
Step 2

Scope Definition: The project will cover the development of a web-based platform for patient information management, appointment scheduling, and treatment tracking.
Boundaries: The project will not include developing an in-house electronic health record (EHR) system but will integrate with existing EHR systems.
Depth: Initial deployment will focus on core functionalities such as patient record management and appointment scheduling, with treatment tracking to be developed in subsequent phases.
Stakeholder Feedback:
Feedback from healthcare professionals indicated **a strong need for mobile access** to the system and the inclusion of additional functionalities such as prescription management and lab results integration earlier in the deployment phases.

HPMS
Step 2 - Revised

Scope Definition: The project will cover the development of a web-based platform ***and mobile app*** for patient information management, appointment scheduling, and treatment tracking.

Boundaries: The project will not include developing an in-house electronic health record (EHR) system but will integrate with existing EHR systems ***and with a wider range of healthcare systems for a holistic approach to patient management***.

Depth:

- ***Phase 1:*** Initial deployment will focus on core functionalities such as patient record management and appointment scheduling, with treatment tracking to be developed in subsequent phases.
- ***Phase 2: Expand functionalities to include lab results integration and treatment tracking.***

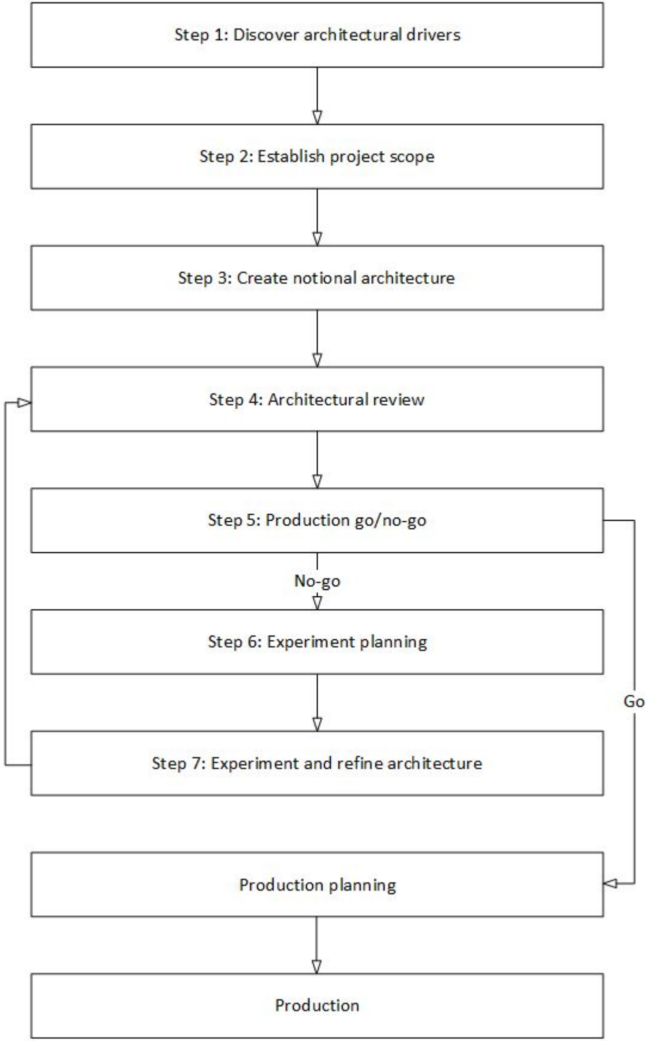
Details:

- Functional Requirements: Detailed list including patient record management specifications, appointment scheduling features, and prescription management workflows.
- Quality Attributes: Defined metrics for security (e.g., compliance with HIPAA), scalability (e.g., support for X number of concurrent users), and usability (e.g., compliance with WCAG 2.1).
- Constraints: Clear definitions of development timelines, technological limitations, and integration standards.
- Principles: Defined architectural principles such as adopting a microservices architecture for modularity and scalability, and continuous integration/continuous deployment (CI/CD) practices for rapid development cycles.

3 – Create notional architecture

A notional architecture diagram is a **high-level depiction of a system design** detailing the **logical components** of a system and their potential **interrelationships**. It is used to provide a conceptual overview of a system, detailing the interactions and data flow between the components.

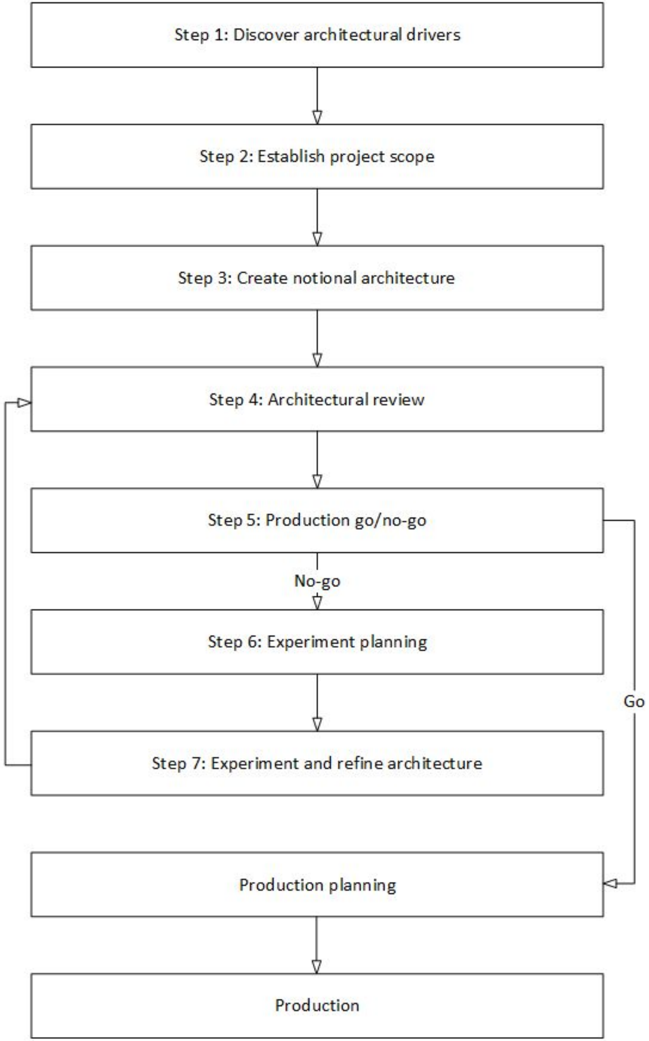
It depicts the **baseline architecture** by identifying key architectural elements based on the refined architectural drivers from Step 2. This step sets the foundational structure of the system, identifying **main components, modules, services, or subsystems, and how they interact** to fulfill the system's requirements and adhere to its constraints and principles.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Go
Step 7: Experiment and refine architecture
Production planning
Production

3 – Create notional architecture

Results:

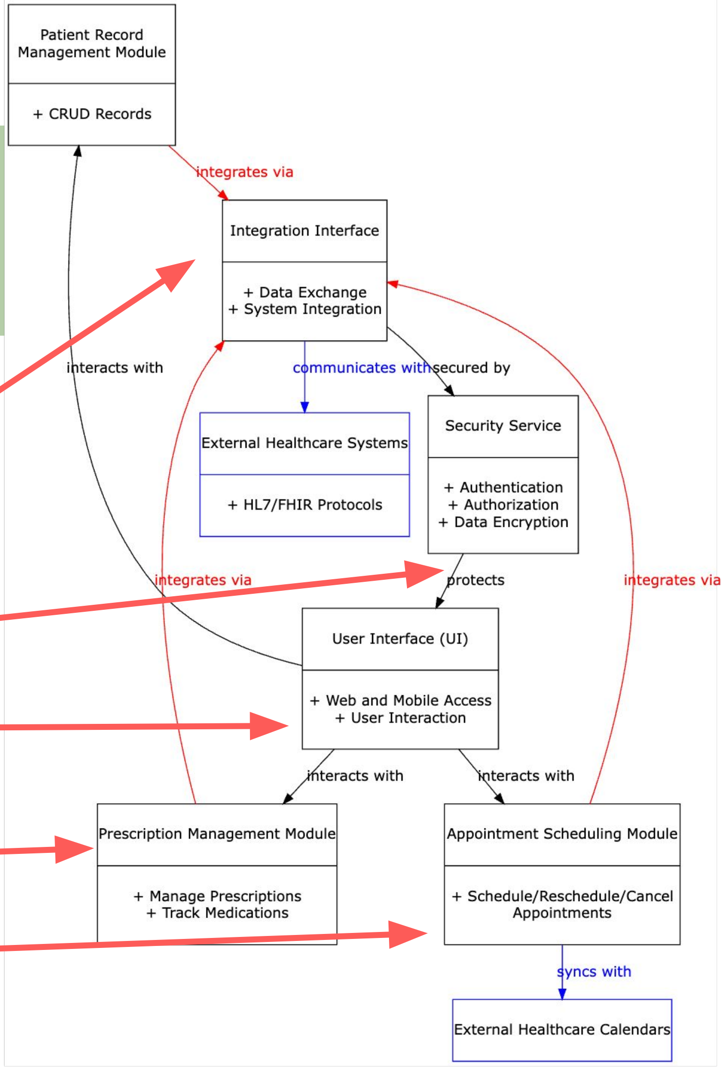
- Define modules, interfaces, components and interactions
- Define basic UI components
- Identify cross cutting concerns
- UML diagrams with basic data flow between major interactive components
- Cross check results with architectural drivers and project scope
- Check with stakeholders
  - iterate if needed


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

HPMS
Step 3

Architectural Elements Identified

- Patient Record Management Module: Handles CRUD operations for patient records, ensuring data is stored securely and accessible only to authorized personnel.
- Integration Interface: Facilitates seamless data exchange between the HPMS and external healthcare systems, including EHRs and pharmacy management systems.
- Security Service: Enforces authentication, authorization, and encryption, ensuring compliance with HIPAA and other healthcare regulations.
- User Interface (UI): Offers a user-friendly interface for both healthcare professionals and patients, accessible via web and mobile platforms.
- Prescription Management Module: Tracks prescriptions issued to patients, their fulfilment status, and medication schedules.
- Appointment Scheduling Module: Manages scheduling, rescheduling, and cancellation of appointments, integrating with healthcare professionals' calendars.



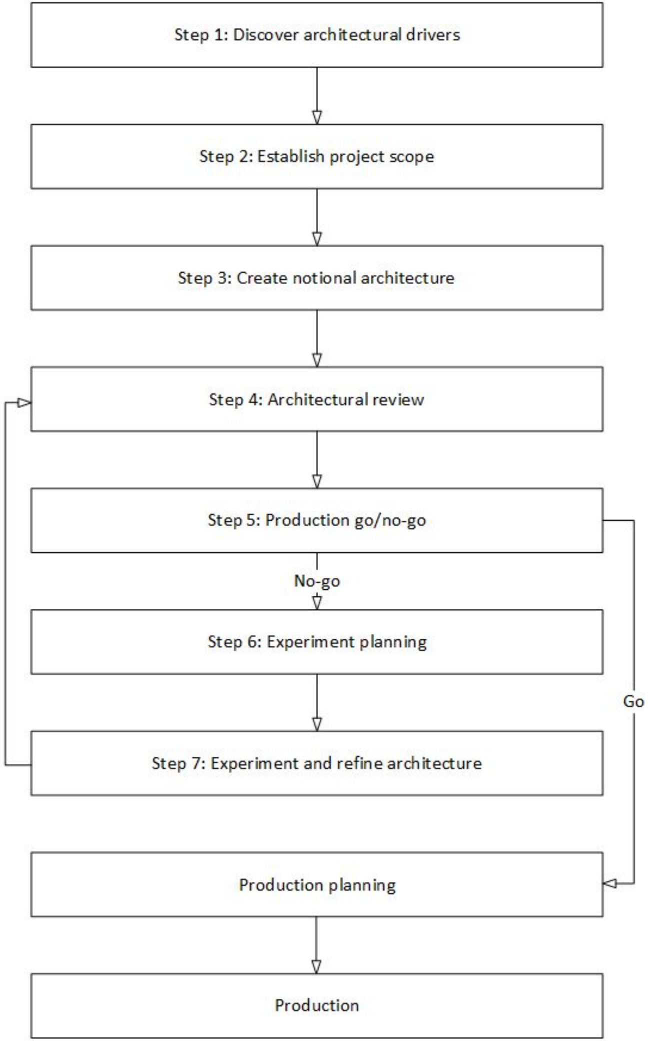
4 – Architectural review

Review current architecture detail - internally, externally, with stakeholders, ..

Goal: Ensure that all of the design decisions are correct and uncover any potential issues or problems with the architecture. For a given design decision, alternative approaches can be discussed, along with the trade-offs and the rationale behind the decision, in order to determine whether the best alternative was taken.

Review focuses on **identifying and evaluating architectural tactics and patterns** that can address the architectural drivers established earlier.

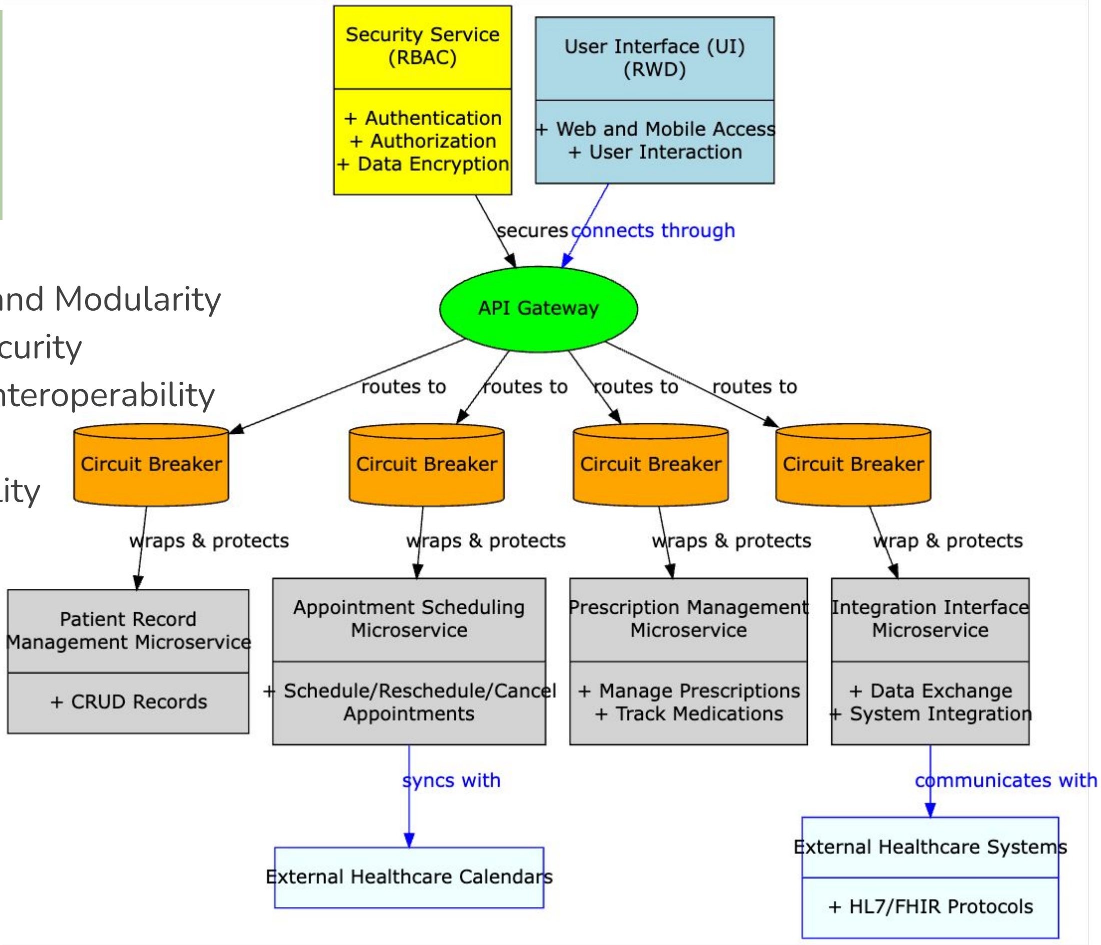
This step involves selecting **specific strategies** to achieve the desired quality attributes, meet functional requirements, and adhere to constraints, thereby **refining the baseline architecture into a more detailed and actionable design**.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

HPMS
Step 4

Tactics and Patterns:

- Microservices Architecture for Scalability and Modularity
- Role-Based Access Control (RBAC) for Security
- API Gateway Pattern for Integration and Interoperability
- Circuit Breaker Pattern for Reliability
- Responsive Web Design (RWD) for Usability



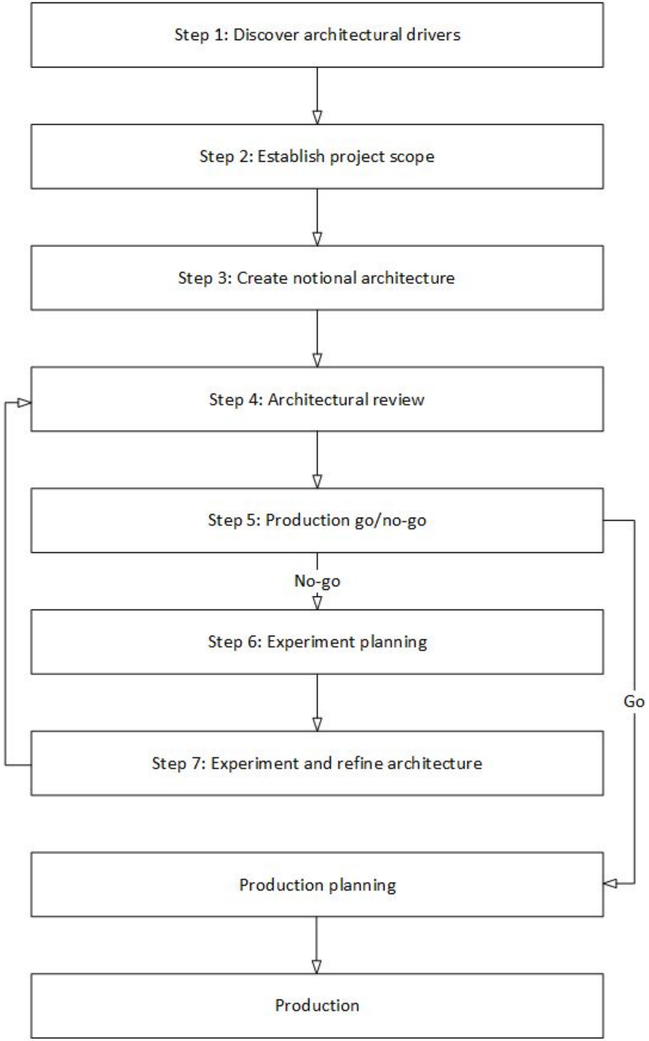
5 – Production go/no-go

Decide whether the project is ready to proceed to production based on the outcomes of the architectural review and other considerations such as risk assessments, resource availability, and alignment with strategic goals.

Architecture is complete and ready for production? Production Planning! (go)

No-go? Any parts can be considered ready? Production planning for those parts.

The remaining ones: experiment planning.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

HPMS
Step 5

Architectural Review Outcomes:
Strengths:
1) The modular design for patient record management, appointment scheduling, and prescription handling was well-received. **2)** Stakeholders appreciated the focus on security and privacy, considering the sensitive nature of healthcare data.
Areas of Concern: 1) The integration with legacy systems in hospitals and clinics (for patient records and lab results) was found to be more complex than initially anticipated.

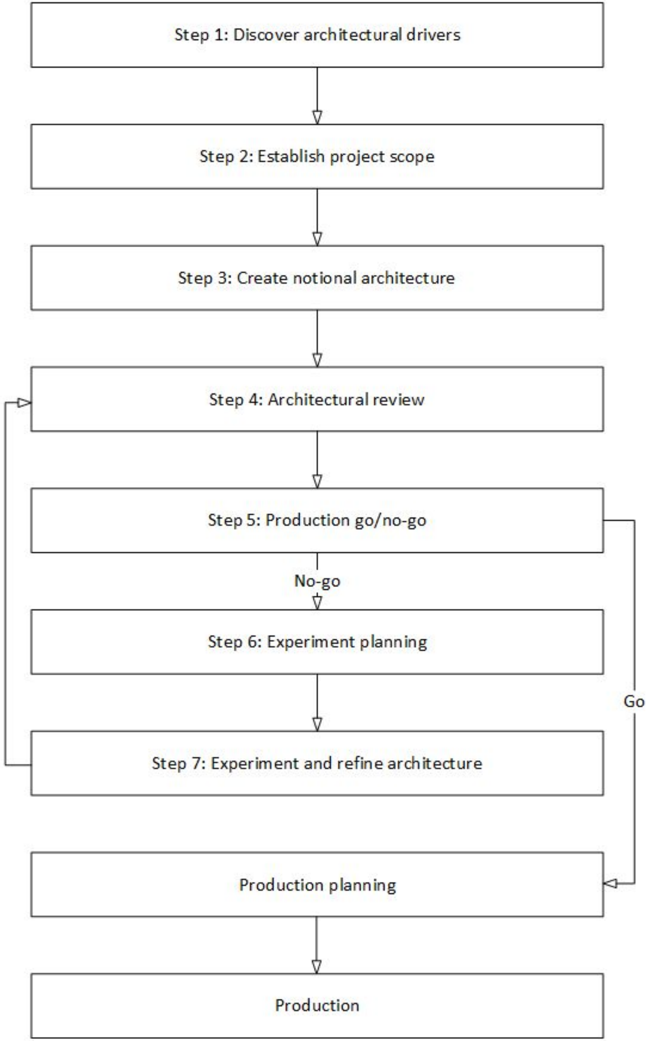
Decision Go:
Module Development: Proceed with the development of individual modules for patient record management, appointment scheduling, and prescription management. These modules have clear requirements and their design aligns with the project's goals.
Security Implementation: Continue implementing robust security measures, given the positive feedback and the critical importance of data protection in healthcare applications.

Partial Hold:
Legacy System Integration: Initiate a detailed analysis phase to better understand the variability in legacy systems and explore middleware solutions or adapter patterns that could simplify integration.

6 – Experiment planning

The primary aim is to design experiments that will test critical aspects of the system's architecture, **focusing on areas that are uncertain** or risks pose significant risk.

The purpose of an experiment may be to resolve an issue uncovered during the architectural review, to gain a greater understanding of one or more architectural drivers, or to improve elements and modules of the design before they are committed to the overall architecture.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

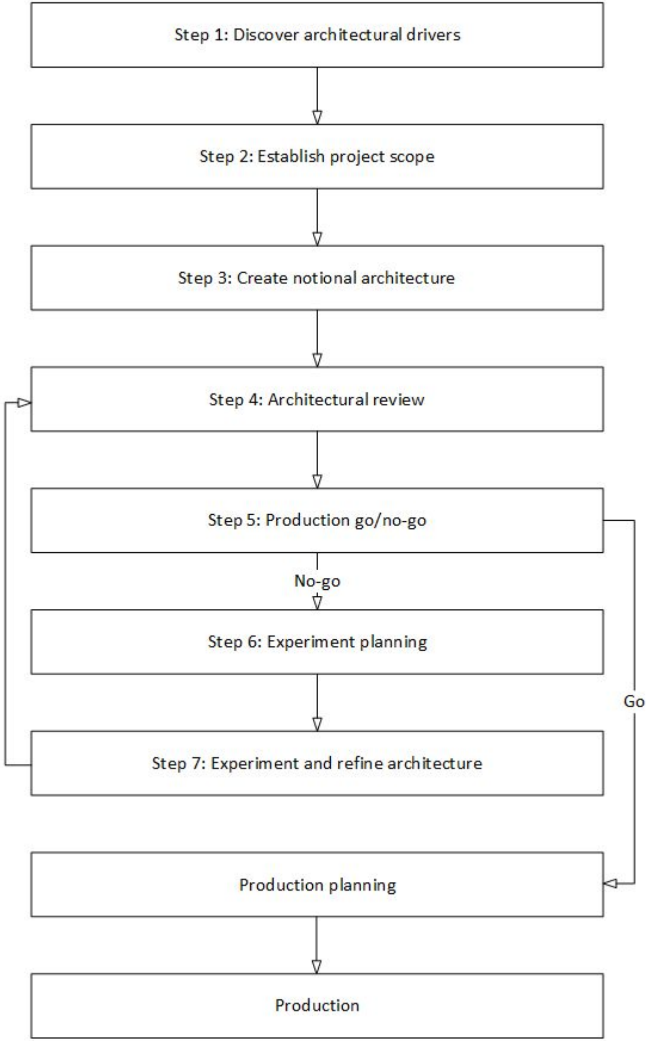
HPMS
Step 6

- Review no-go and partial hold decisions and identify actionable tasks
- Task Forces: Establish dedicated task forces to address each area of concern, with clear timelines and milestones for reevaluation.
  - no-go / partial hold: Integration with Legacy Systems
  - Planning: seek out existing Enterprise Service Bus that offers built-in connectors for common healthcare systems and develop rapid proof of concept on the ease of integration.
- Stakeholder Engagement: Schedule regular update meetings with stakeholders to report progress on the addressed concerns and integrate feedback continuously.
- Risk Management Plan: Update the project's risk management plan to include strategies for mitigating potential delays or challenges arising from these areas of concern.

7 – Experiment and refine architecture

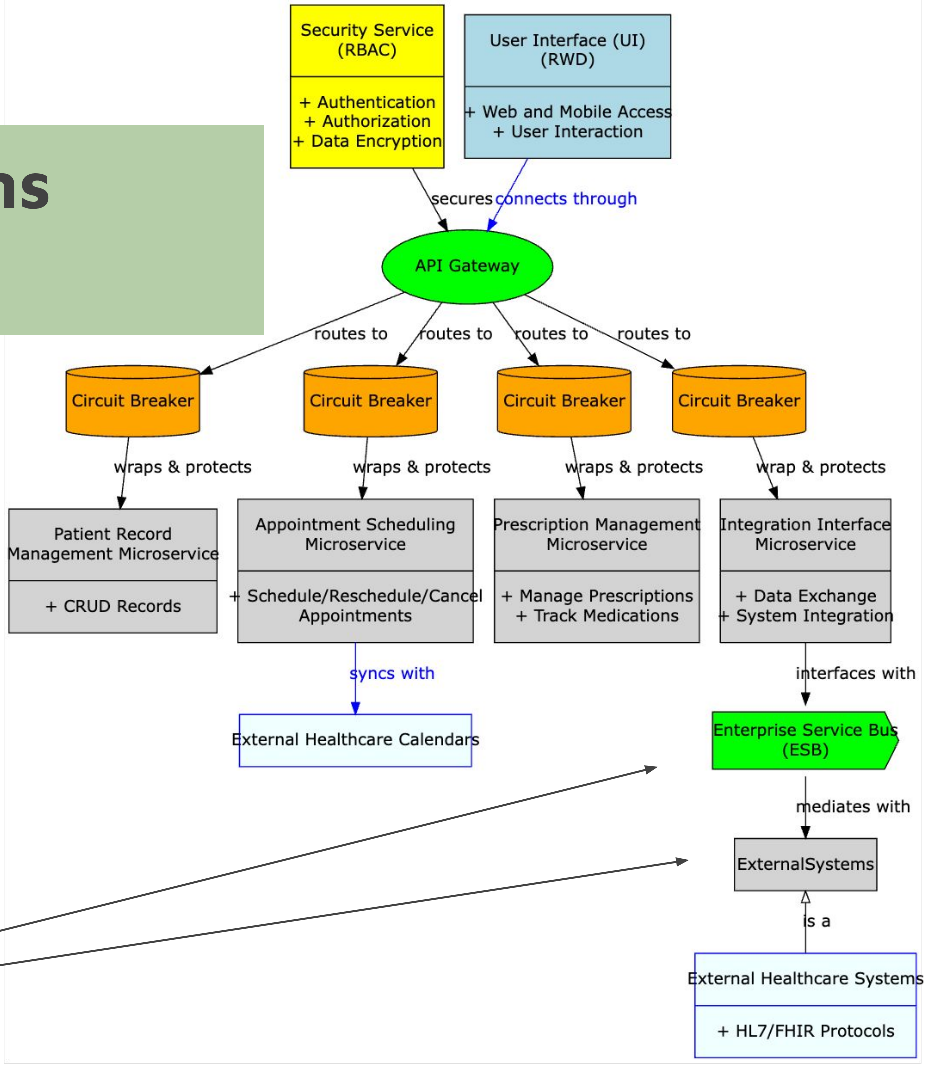
Any experiments that were planned are executed during this step. The results of the experiments are recorded. Based on the results of the experiment, if the architecture needs to be refined, it is done during this step.

After the refinement is complete, **the process goes back to Step 4** so that another architectural review can take place.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Go
Step 7: Experiment and refine architecture
Production planning
Production

HPMS -Step 7 - actions and results

No-Go from step 5: **Integration with Legacy Systems**
Goal: Identify and validate an efficient method for integrating with diverse legacy systems.
Action: The team experiments an Enterprise Service Bus (ESB) that offers built-in connectors for common healthcare systems and supports custom adapter development. A proof-of-concept (POC) is developed to test the ESB's effectiveness in simplifying integration.
Result: The POC demonstrates that the ESB significantly reduces the complexity and effort required for integration, providing a flexible and scalable solution. The team decides to incorporate the ESB into the HPMS architecture.



Production planning and production

Refers to using the architecture in implementation!

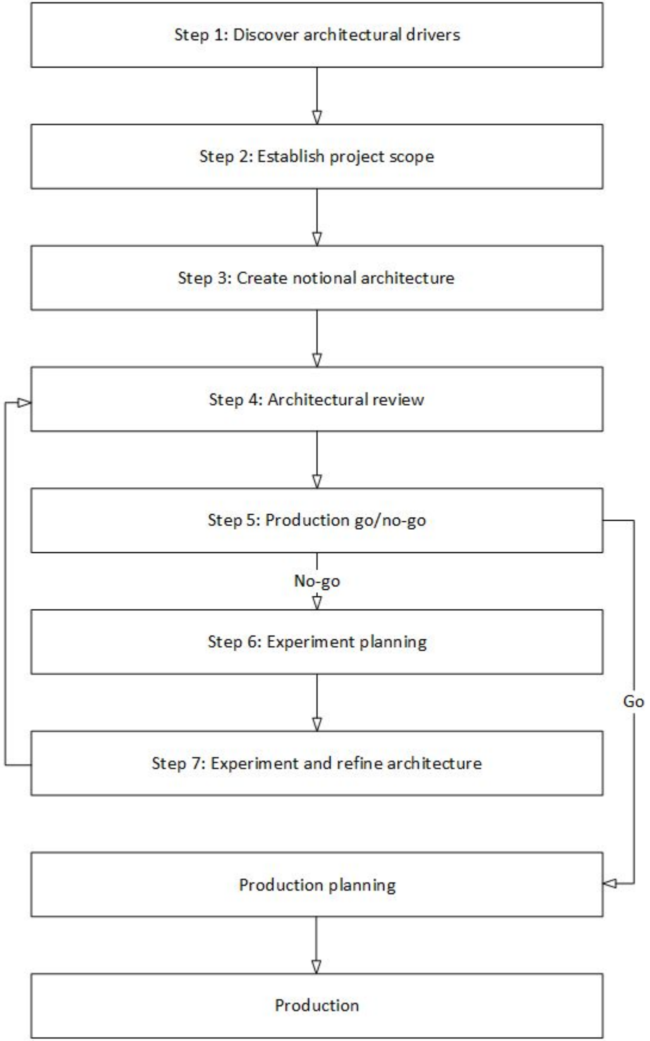
Production Planning

Production planning involves planning the design and development of elements, scheduling the work, and assigning tasks to resources.

The project management team creates plans for the work, and bases them, in part, on the architecture.

Production

Once the architecture can be moved to production, it can be used by development teams for the detailed design of elements, coding, integration, and testing.


Step 1: Discover architectural drivers
Step 2: Establish project scope
Step 3: Create notional architecture
Step 4: Architectural review
Step 5: Production go/no-go
No-go
Step 6: Experiment planning
Step 7: Experiment and refine architecture
Go
Production planning
Production

Smart Home Energy Management System Design using ACDM

https://docs.google.com/document/d/1XIw12qxY3Wgm_0SbMfJVmBIMbxaGeMu4Qbwdc6TZJeA/edit

Group assignment
60 min

Bibliography

- https://learning.oreilly.com/library/view/software-architects-handbook/9781788624060/9236a39f-8469-4632-b013-4ed9582aab77.xhtml
- https://learning.oreilly.com/library/view/software-architecture-in/9780136885979/part02.xhtml
- https://learning.oreilly.com/library/view/software-architects-handbook/9781788624060/0adb7344-e386-4f19-8493-55d6d02c2345.xhtml
- https://learning.oreilly.com/library/view/software-architecture-the/9781492086888/ch01.html