Solution Architecture Training Notes

By Aung Kyaw Minn

License Information

Preface

The information presented in this document is a compilation of:

- **Experience**: Insights and lessons learned from previous projects.
- **Knowledge**: Understanding acquired through trainings, experiments, and interactions with professionals.
- **Skill**: Practical skills having been refined through hands-on practice and application.

Aung Kyaw Minn

Goal & Objectives

- The primary goal of this training is to equip experienced engineers with the knowledge and skills required to transition into the role of a Solution Architect.
- This training will cover essential concepts, methodologies, and best practices in solution architecture, enabling participants to design robust, scalable, and efficient systems that align with business goals and technical requirements.

Expected Outcomes

By the end of the training, participants will be able to:

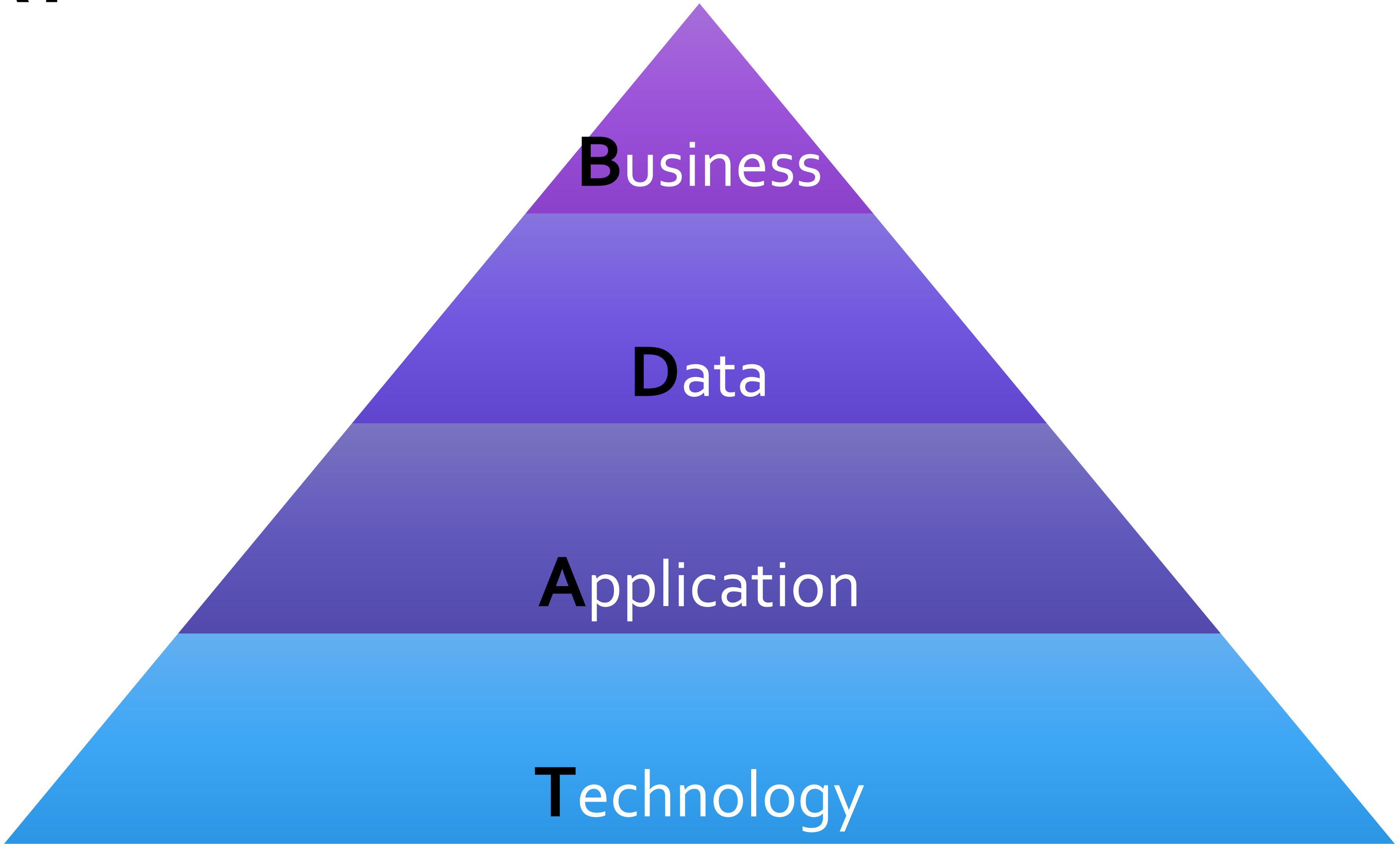
1. Understand the core principles of solution architecture.
2. Design end-to-end solutions that address complex business problems.
3. Evaluate and select appropriate technologies and frameworks.
4. Create architectural blueprints and documentation.
5. Ensure solutions are scalable, secure, and maintainable.
6. Collaborate effectively with stakeholders, including business leaders, developers, and other architects.

List of Contents

Architecture Domains

Basic Concepts

BDAT



Business Domain

Considerations, Roles and Responsibilities

Considerations

- Business Requirements
  - Start by understanding what the business needs from the software, defining the problem it's supposed to solve, and how it aligns with the business strategy.
- Business Processes
  - Identifying and analyzing the business processes that will be impacted or improved by the software.
- Stakeholders
  - Ensuring that the views and needs of all stakeholders are considered during the software development process.

Roles

- Business Analysts
- Business Architects
- Product Managers
- Product Owners

Responsibilities

- Defining business strategy and goals.
- Mapping out business processes and capabilities.
- Communicating requirements to technical teams.
- Ensuring that IT solutions align with business objectives.

Deliverables

- Business Strategy Documents
  - Outlines the goals, objectives and tactics of the organization.
- Business Process Models
  - Visual representation of the business processes, often created using BPMN (Business Process Model and Notation).
- Value Stream Maps
  - Identify the value-adding steps in business processes to enhance efficiency and effectiveness.
- Capability Maps
  - Depict the core capabilities of the business and maturity of these capabilities.

Data Domain

Considerations, Roles and Responsibilities

Considerations

- Data Modeling
  - Designing the data structures that will be used by the software, ensuring that the data model supports the business processes.
- Data Management
  - Considering how data will be captured, stored, retrieved, archived, and deleted by the software.
- Data Security and Privacy
  - Applying policies and mechanisms to protect data security and privacy, which is especially important if you handle sensitive or personal information.

Roles

- Data Architects
- Data Analysts
- Database Administrators

Responsibilities

- Developing data models and database designs.
- Implementing data quality and data governance processes.
- Managing data security, compliance, and privacy.
- Supporting data analytics and business intelligence efforts.

Deliverables

- Data Models
  - › Include conceptual, logical and physical data models.
- Data Dictionary
  - › A catalog of all data elements, detailing attributes like data type, source, usage and ownership.
- Data Flow Diagrams
  - › Visualize the flow of data through the system, showing how data is processed and stored.
- Master Data Management Strategy
  - › Define how key enterprise data (like customer or product information) is managed across the organization

Application Domain

Considerations, Roles and Responsibilities

Considerations

- Application Architecture and Design
  - Deciding on the architecture of the software, including how it will be broken down into components, services, or microservices.
- Development Tech-Stack and Frameworks
  - Choosing the right frameworks, programming languages, and tools that will be used to build the software.
- User Experience
  - Ensuring that the application is user-friendly and meets the end-users' requirements.

Roles

- Application Architects
- UI/UX Designers
- QA Engineers
- Software Development Team Leads

Responsibilities

- Designing the architecture of the application.
- Ensuring application scalability, reliability, and security.

Deliverables

- Software Architecture Documents
  - System Design Diagrams
  - Detailed Descriptions
  - Quality Attribute Requirements
  - Technology Choices
  - Risk Analysis
  - Data Integration and Management Strategies

Technology Domain

Considerations, Roles and Responsibilities

Considerations

- Development
  - Development environment setup, actual coding, and integrations
- Deployment
  - Planning how the software will be deployed, updated, and maintained.
  - Determining what kind of infrastructure is needed to support the software, whether it's on-premises servers, cloud-based services, or a hybrid approach.
- Performance and Scalability
  - Ensuring that the technological solutions chosen can handle the expected load and can scale as the number of users or the volume of data grows.

Roles

- Software Engineers/Developers
- SREs
- DevOps Engineers
- Network Engineers
- Cloud Specialists
- Security Architects
- System Administrators

Responsibilities

- Designing the technology infrastructure to support business operations.
- Ensuring IT systems are scalable, secure, and resilient.
- Managing the deployment and ongoing maintenance of technology systems.
- Ensuring compliance with IT policies and standards.

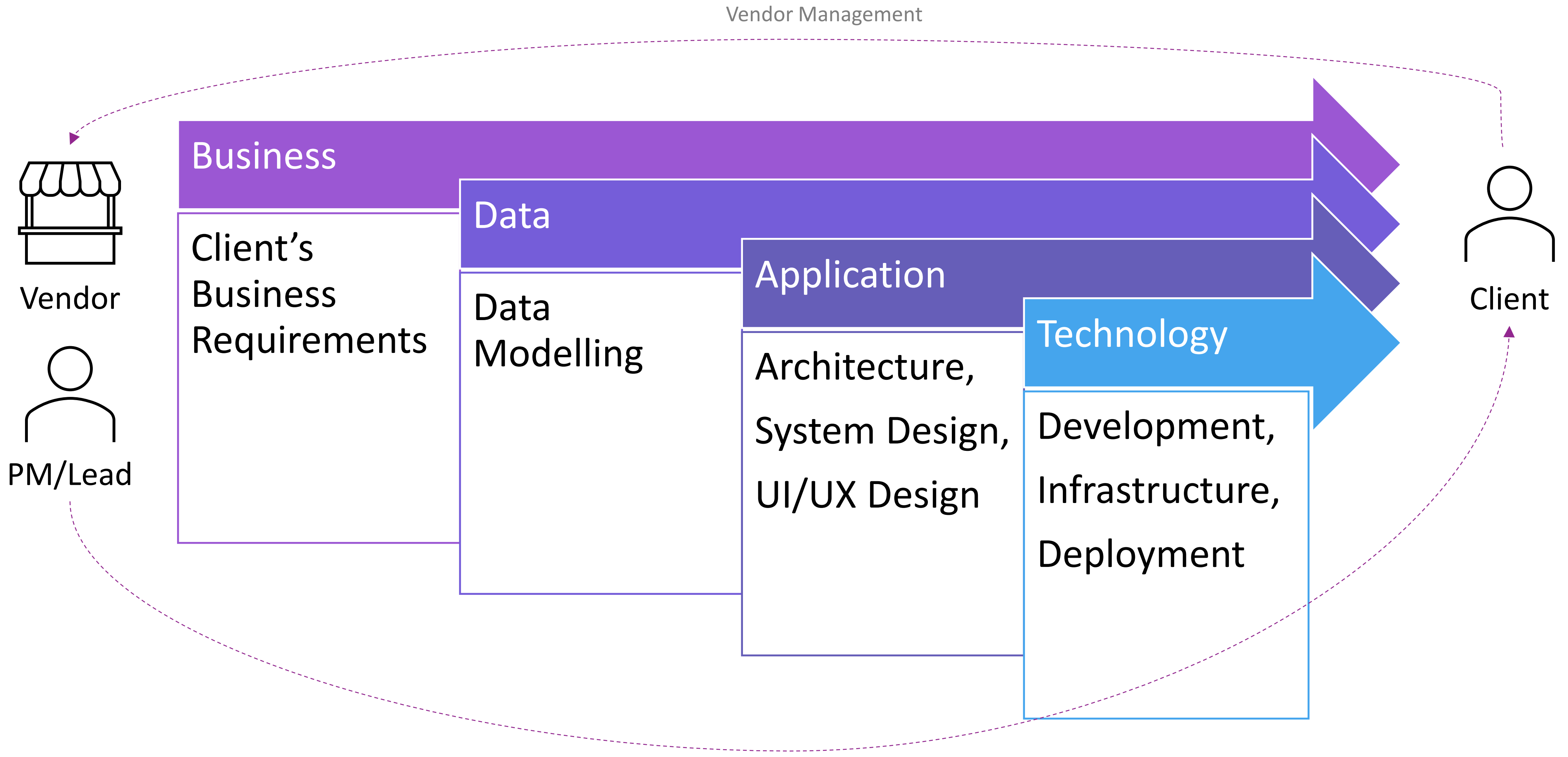
Deliverables

- Application/Software
- Infrastructure Diagrams
  - Show the physical and logical layout of IT infrastructure, including servers, network devices, and storage systems.
- Technology Standard Documents
  - Outline the approved technologies, platforms, and tools for use within the organization.

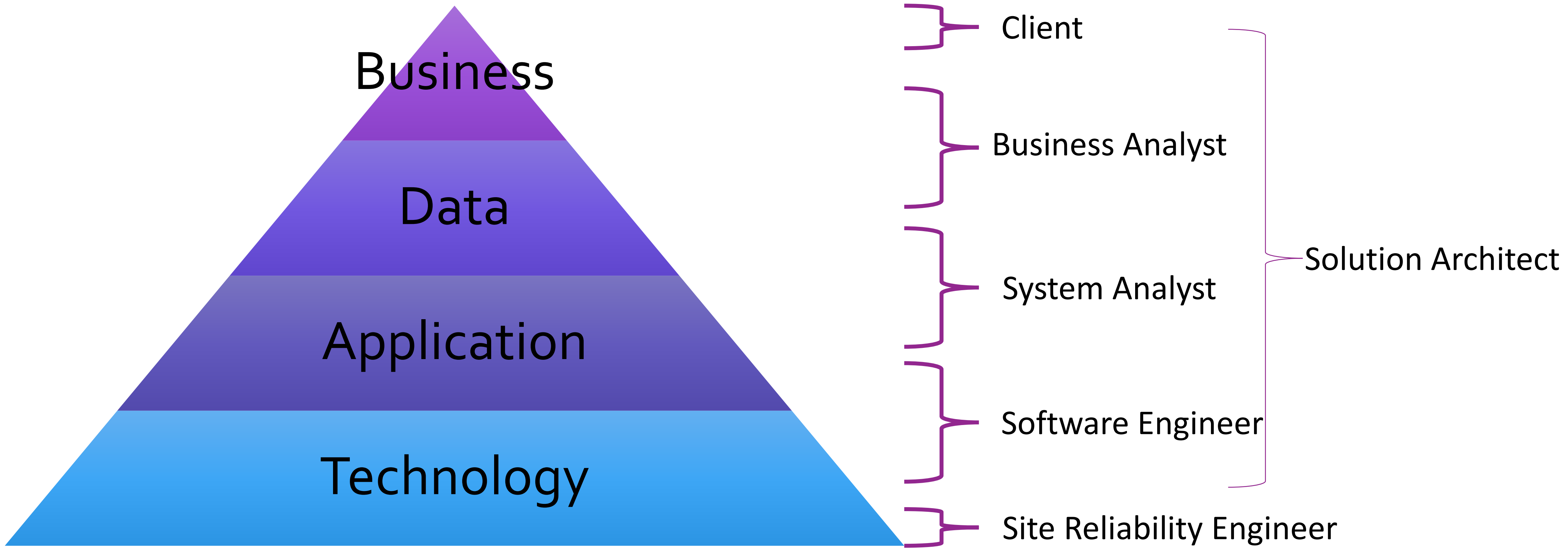
Software Delivery

From building the software to deploying it into a production environment, where it becomes operational for end-users

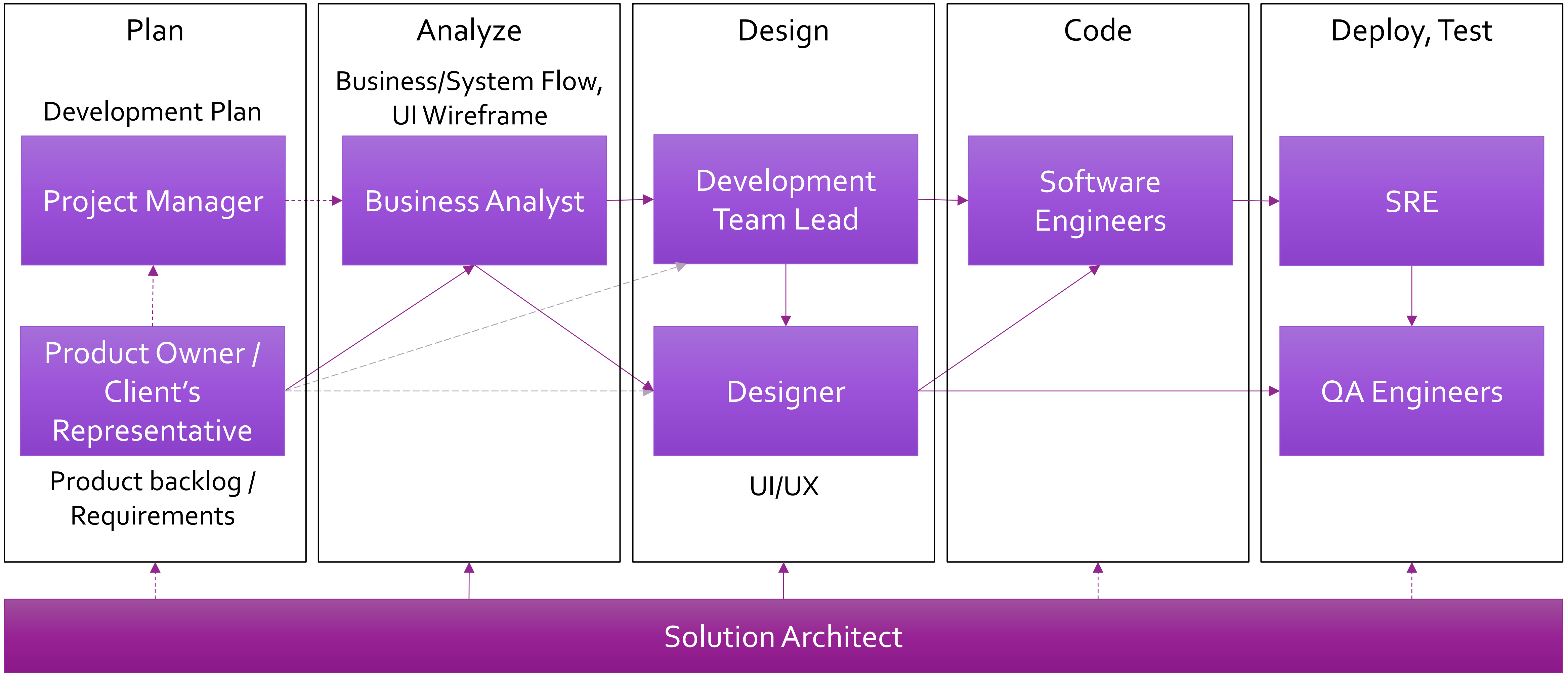
Domains and Process



Domains and Roles



Workflow and SDLC



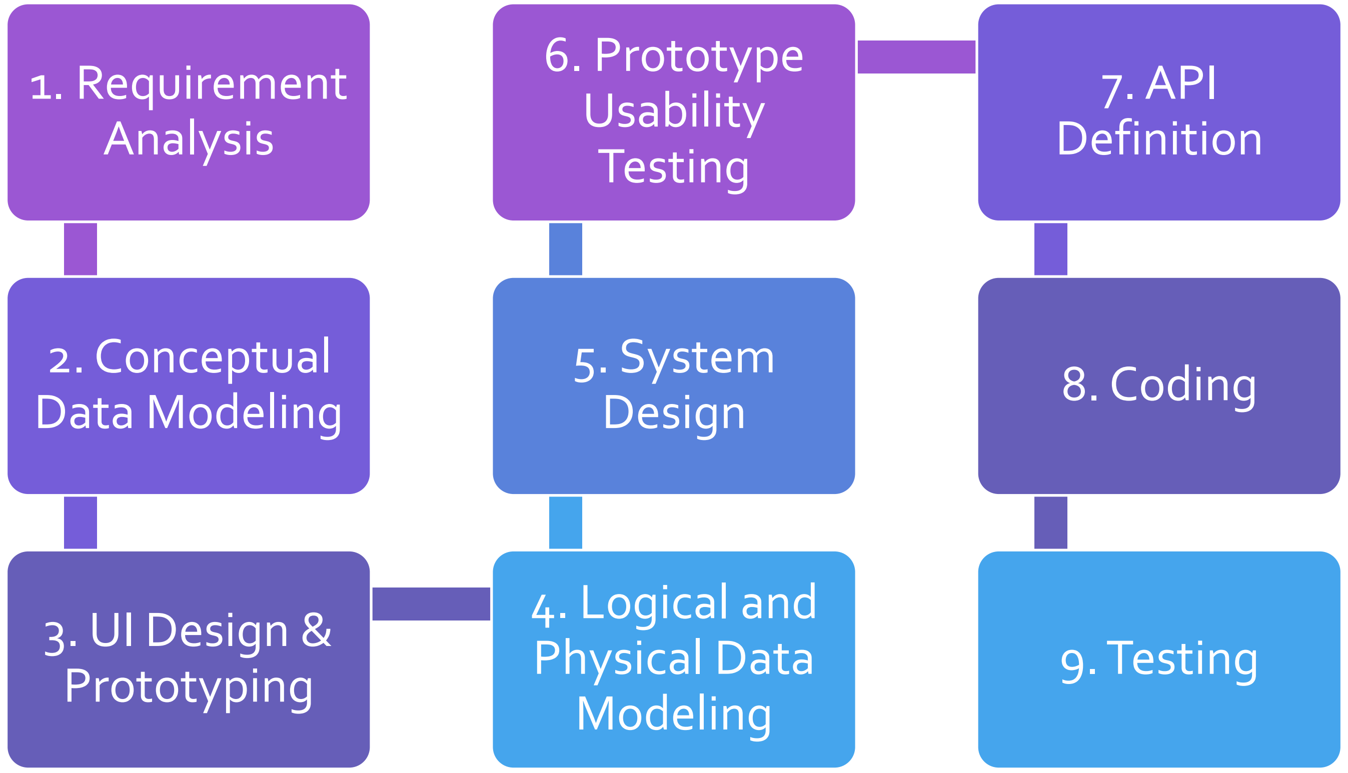
Software Development Activities

#	Activity	Role	Output	Document Tools
1	Requirement Analysis	BA	Diagrams, Wireframe, BRS	whimsical, Word
2	Conceptual Data Modeling	SA	ER Diagram	drawio
3	UI Design & Prototyping	Designer	Design Mockup, Prototype	Figma
4	Logical & Physical Dat Modeling	SA	DB Schema	dbdiagram / dbdocs
5	System Design	SA	System design artifacts	drawio
6	Prototype Usability Testing	BA, QA	Prototype, Test Plan/Case	Figma
7	API Definition	SA / Lead	API Documentation	Postman
8	Coding	SE	Software	Scribe / Swagger UI
9	Testing & Creating User Manual	QA	Issues, User Guide Docs	Excel, Clickup

Development Process

Order of activities might be adjusted according to the situation. That's about your **Wisdom**.

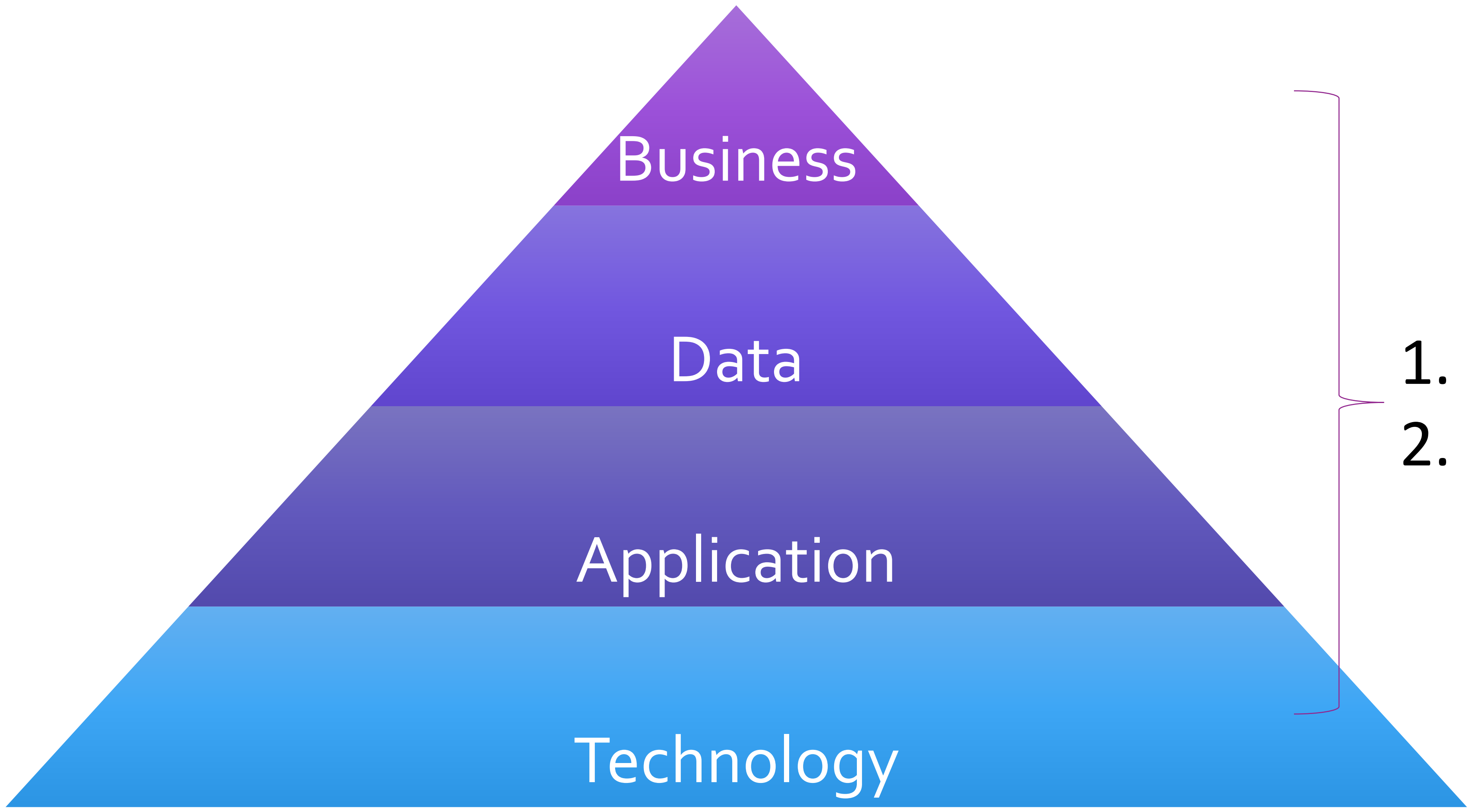
How wise are you?


1. Requirement Analysis
2. Conceptual Data Modeling
3. UI Design & Prototyping
4. Logical and Physical Data Modeling
5. System Design
6. Prototype Usability Testing
7. API Definition
8. Coding
9. Testing

Solution Architect

Serves as a bridge between business and technology domains, ensuring that technology solutions align with business goals and requirements.

What’re the works



1. Architecture
2. Solution

Business Domain – CRPM

- **C**ontext
  - Environment and Stakeholders
- **R**equirements
  - Key requirements and goals
- **P**rocesses
  - Outline of relevant business processes
- **M**odels
  - How the business deliver value to customer and revenue is generated

Business Domain – CRPM Example

1. Business Context
   - E-ComMax is an online retail platform designed to facilitate the buying and selling of products. It serves a diverse customer base with a wide range of product categories.
2. Business Requirements
   - Enable customers to browse and purchase products online.
   - Support vendors in listing and managing their products.
   - Facilitate payment processing and order fulfillment.
   - Provide customer service and support features
3. Business Processes
   - Product Listing – Vendors list their products with description, images and pricing.
   - Order Processing – Customer place orders, and the system manages order workflows.
   - Payment Processing – Integration with payment gateways to handle transactions.
   - Customer Support – A dedicated support team handles customer queries and issues.
4. Business Models
   - E-ComMax employs a commission-based business model, where the platform earns a percentage of each sale.

Data Domain – SMFSG

- Sources
  - › Master data and transactional data – table names
- Models
  - › Data structure, schemas and relationships – field names and relations
- Flow
  - › How data moves through the system
- Storage
  - › Storage solutions and data management
- Governance
  - › Policies and compliance

Data Domain – SMFSG Example

1. Data Sources
   - Product Data: Product details from vendors.
   - Customer Data: Information about customers and their purchase history.
   - Transaction Data: Records of sales and financial transactions.
2. Data Models
   - Product Schema: Fields for product – id, name, description, price, category
   - Customer Schema: Fields for product id, name, contact information, order history
   - Order Schema: Fields for order – id, product details, customer id, order status
3. Data Flow (include both text description and diagrams)
   - Data flows from product listing to order processing to payment and customer support.
   - The system interacts with external services for payment processing and delivery logistics.
4. Data Storage
   - E-ComMax uses a relational database to store product and customer data. Transaction data is stored securely and encrypted for compliance with security standards.
5. Data Governance
   - The platform adheres to GDPR and other privacy regulations, ensuring customer data is protected and confidential.

Application Domain – ACILI

- Architecture
  - Overview of application structure
- Components
  - Main components and their roles
- Interfaces
  - User interfaces and interactions
- Logic
  - Business rules, conditions and decision-making
- Integration (Optional)
  - Integration points and APIs

Application Domain – ACILI Example

1. Application Architecture
   - E-ComMax uses a microservices architecture with a front-end website and a series of back-end services.
   - The front-end is built with a React framework, while the back-end uses Node.js and Express
2. Application Components
   - Frontend Website – User facing interface for customers
   - Vendor Portal – Separate portal for vendors to manage their products
   - Order Service API – Handles order processing and status updates
   - Payment Service API – Integrates with external payment gateways
   - Customer Support Service API – Manages customer inquires and support tickets
3. User Interfaces
   - Customer UI: Designed for easy navigation and product browsing
   - Vendor UI: Offers tools for product management and sales tracking
4. Application Logic
   - Product Categorization – user assign category manually when creating a product
   - Order Validation – before order is accepted, the system checks if the requested products are in stock. If a product is out of stock, the order is flagged or blocked and "Out of stock" message is returned to the customer
   - Payment Processing – system checks if the payment method provided by the customer is valid
5. Integration
   - E-ComMax provides RESTful APIs for external integrations, allowing vendors to automate product uploads and updates. It also integrates with third-party services for payment processing and logistics

Technology Domain – DIDSP

- Development
  - Frameworks, Languages, and Tools
- Infrastructure
  - Domain, Server, and Networking
- Deployment
  - Process and Practice
- Security
  - Measures and Practice
- Performance
  - Benchmarks and Scalability

Technology Domain – DIDSP Example

1. Development
    - Frontend – Vue 3 Composition API, Pinia
    - Backend – PHP, Laravel
    - Database – PostgreSQL
    - DevOps – Gitlab CI/CD
2. Infrastructure
    - E-ComMax operates on a cloud infrastructure, using Amazon Web Services (AWS) for hosting and storage.
3. Deployment
    - The platform uses a CI/CD pipeline for automated deployments.
    - Pipeline will be appeared after developer merged to related environment branch, then can click deploy button manually.
4. Security
    - Security measures include encryption for sensitive data, firewalls, and regular security audits
    - Implementation AWS WAF on ALB
5. Performance
    - Performance is monitored using AWS CloudWatch, with alerts set for high latency or system failures
    - The platform scales automatically based on traffic and resource requirements by implementing AWS Auto-Scaling

Responsibilities

- Architecture
  - › Developing data models and database designs.
  - › Application Architecture, System Design and UI Flows
  - › Infrastructure and Deployment
- Solution
  - › Data Management, Security and Privacy
  - › Application Performance, Security, and Scalability
  - › Infrastructure Performance, Security, and Scalability

Architecture

Creating high-level structure or blueprint that defines how the components and systems within the software are organized and interact with each other.

In this document, I described 4 basic parts of architecture, those are "Patterns", "Principles", "Quality Attributes" and "Diagrams".

Architecture Patterns

- Web
  - Monolithic
    - A single unified architecture containing all components.
  - Modular
  - Microservices
  - …
- Android
  - Clean
  - …
- iOS
  - VIP
  - …

Architecture Principles

- Single Responsibility
  - Each component or module should have a single responsibility or purpose
- Modularity
  - Organize the system into modules for better maintainability and reusability
- Encapsulation
  - Hide implementation details and expose only necessary interfaces.
- Separation of Concerns
  - Divide the system into distinct sections that address different concerns.
- Scalability
  - Design to accommodate growth in user load and data volume.

Architecture Quality Attributes

- Performance
  - How quickly the system responds to requests
- Security
  - Protecting the system from threats and unauthorized access.
- Scalability
  - Handle growing amounts of work or an increased number of users gracefully and efficiently
- Maintainability
  - Ease of modifying the system to fix bugs or add new features.
- Availability
  - Ensuring the system remains operational as much as possible.

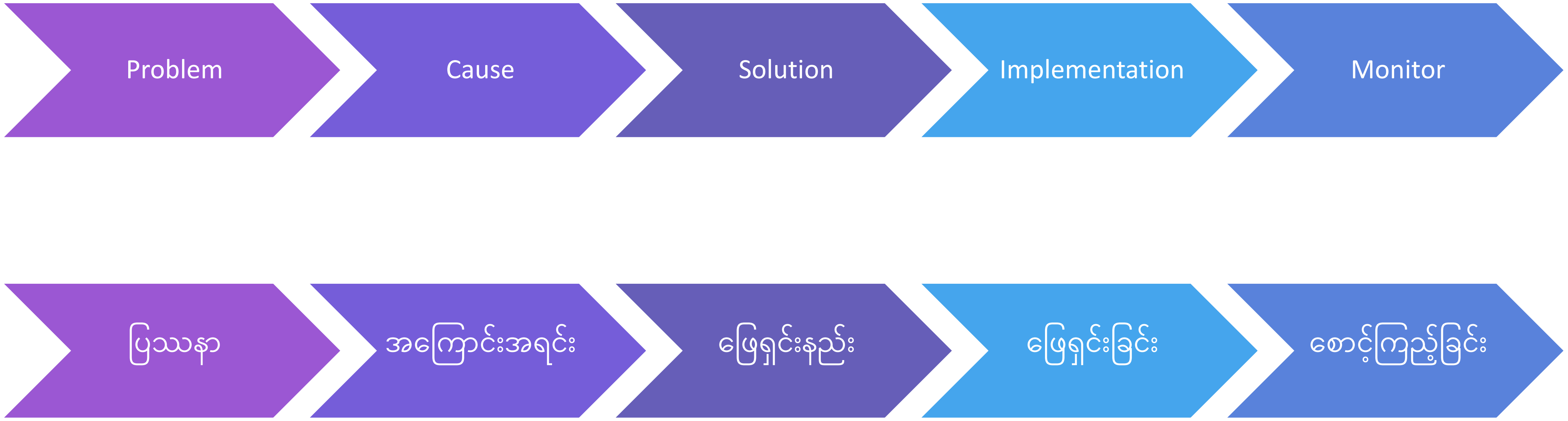
Architecture Views – Diagrams

- Logical View: Describes the system's functionality and components.
  - High-Level Component Diagrams
- Development View: Focuses on the system's organization from a developer's perspective.
  - UML Class Diagram, Package Diagram, ...
- Process View: Describes the dynamic aspects and runtime behavior.
  - Flowchart Diagram, Sequence Diagram, Activity Diagram, ...
- Physical View: Maps logical components to the hardware infrastructure.
  - Deployment Diagram, Infrastructure Diagram, ...

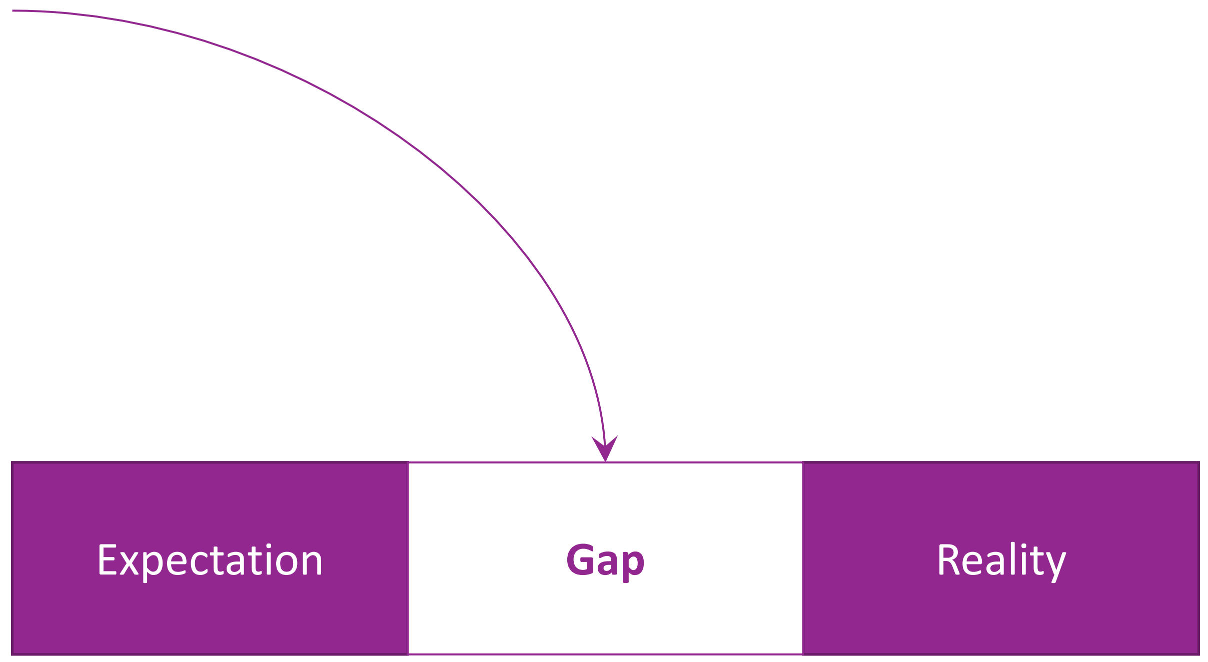
Solution

Solving the challenges and bridging the gaps among architecture domains of business, data, application, and technology

Problem solving in short



Problem



Problem: There are 3 Types

	Knowns	Unknowns
Known	1 **Known knowns** You know how to solve them	2 **Known unknowns** You know ways to find out how to solve them
Unknown	3 **Big unknowns** You don’t know how to solve them because you don’t know the root cause.	

Known Known

- To-Dos
  - Making the checklist
  - Implement the checklist

Known Unknown

- To-Dos
  - Search – Explore Options
  - Select – Best Option
  - Test – POC (Proof of Concept) of Best Option
  - Solve – Implementation of Best Option

Big Unknowns

- To-Dos
  - Empathize – understanding impacts
  - Define – finding the cause
  - Ideate – Explore Options and Select the Best Option
  - Prototype – POC of the Best Option

Learn more there about problem solving

- https://aungkyawminn.medium.com/required-mindsets-and-activities-for-solving-problems-part-1-c196b33cf1bd
- https://aungkyawminn.medium.com/required-mindsets-and-activities-for-solving-problems-part-2-bc9d3790734e
- https://aungkyawminn.medium.com/list/design-thinking-for-product-innovation-c4390c5e8008

5 Whys – Causes and Solution Example

Problem -> App is slow.

1. Why app is slow?
    1. DB queries are slow.
2. Why are DB queries slow?
    1. They are not optimized.
3. Why are they not optimized?
    1. Missing indexes on some tables.

Solution is found.

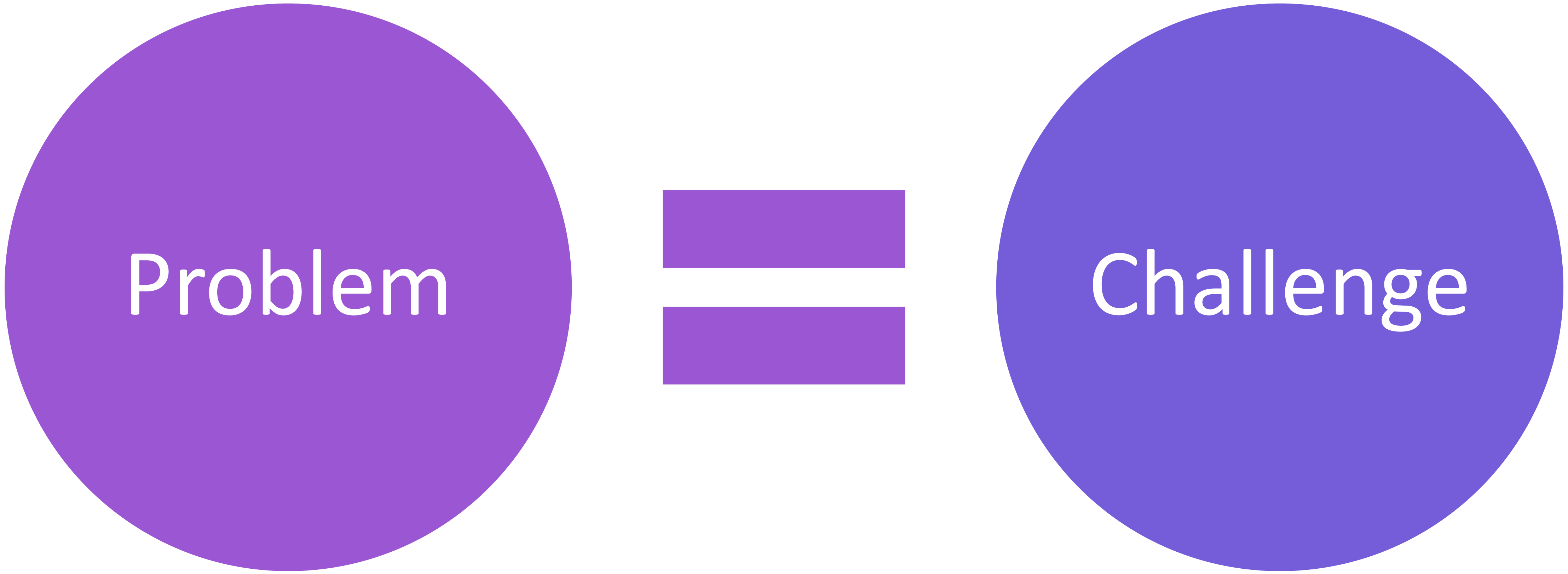
4. Why are there missing indexes?
    1. Indexes were not considered during the initial design phase.

Lessons to learn

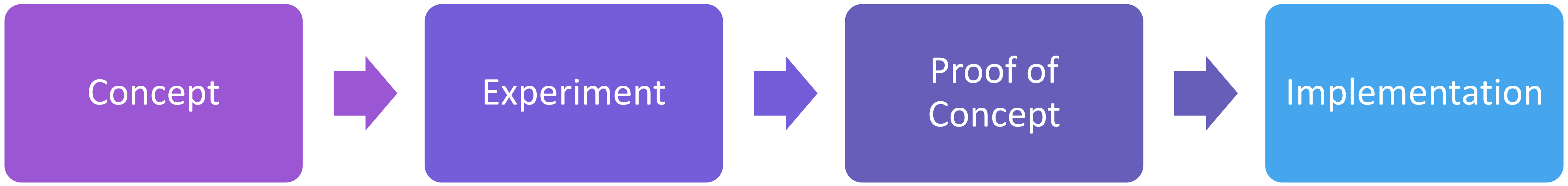
5. Why were indexes not considered?
    1. Lack of experience with large databases during the design phase.

Got the experience

Change Perspective, Change Mind



Solution Journey



Solution Concept

- Purpose
  - This initial phase involves drafting a technical solution based on past experiences and knowledge. It focuses on defining a preliminary approach to address the technical problem.
- Outcome
  - A draft technical solution that outlines the initial approach and potential strategies.

Solution Experiment

- Purpose
  - › In this phase, the draft technical solution is subjected to experiments to test its feasibility and effectiveness. This involves exploring different approaches and making iterative improvements.
- Outcome
  - › Enhanced insights and data that lead to a refined version of the initial concept, showing measurable improvements.

Solution POC (Proof of Concept)

- Purpose
  - › Building on the experiment phase, the POC involves delving deeper into the technical details to develop a more precise and validated solution. This phase aims to solidify the approach and confirm its viability.
- Outcome
  - › A precise and validated technical solution that demonstrates the feasibility and potential success of the proposed approach.

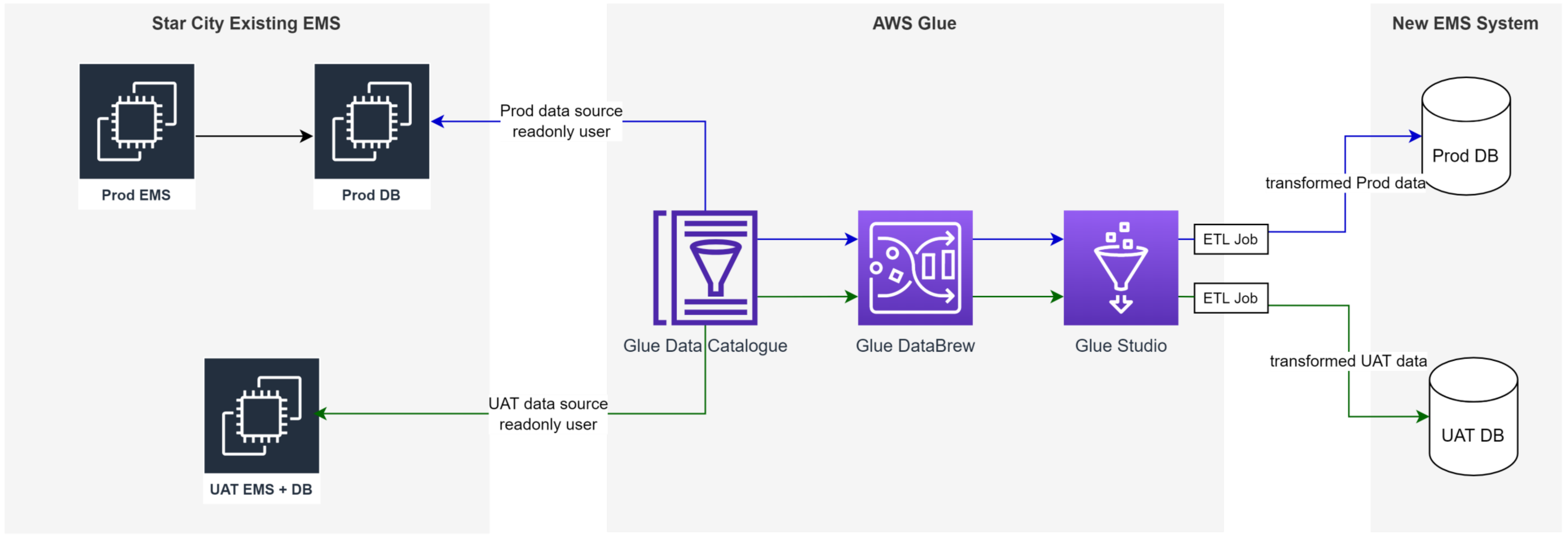
Solution Implementation

- Purpose
  - The final stage involves finalizing the solution with specific tools and techniques and executing its full-scale development and deployment. This includes detailed design, coding, testing, and deployment.
- Outcome
  - A fully developed and implementable solution that is ready for production use, incorporating all the necessary tools and techniques.

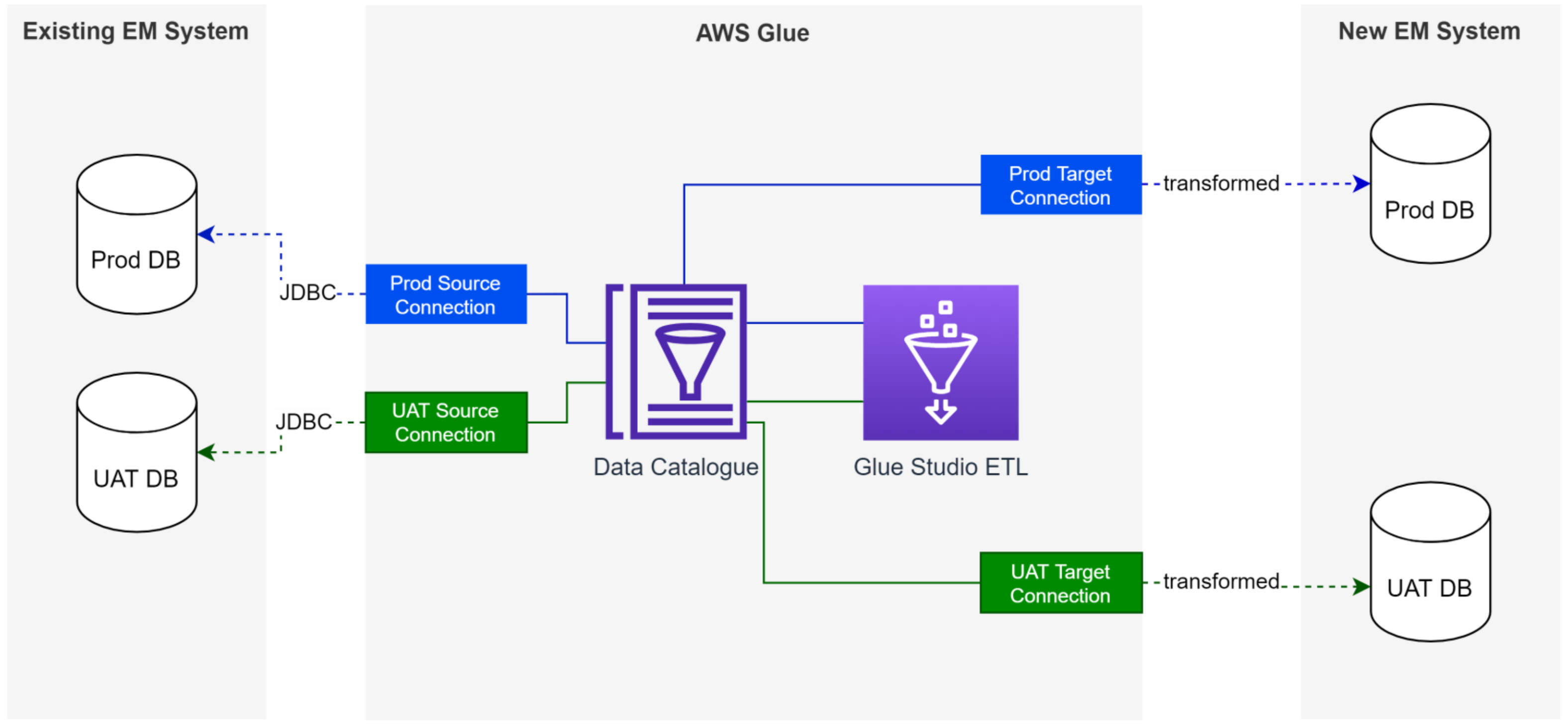
Solution Journey – Example

From Concept to Implementation of Estate Management Software Data Migration from Old System to New System that has different data schema and definitions.

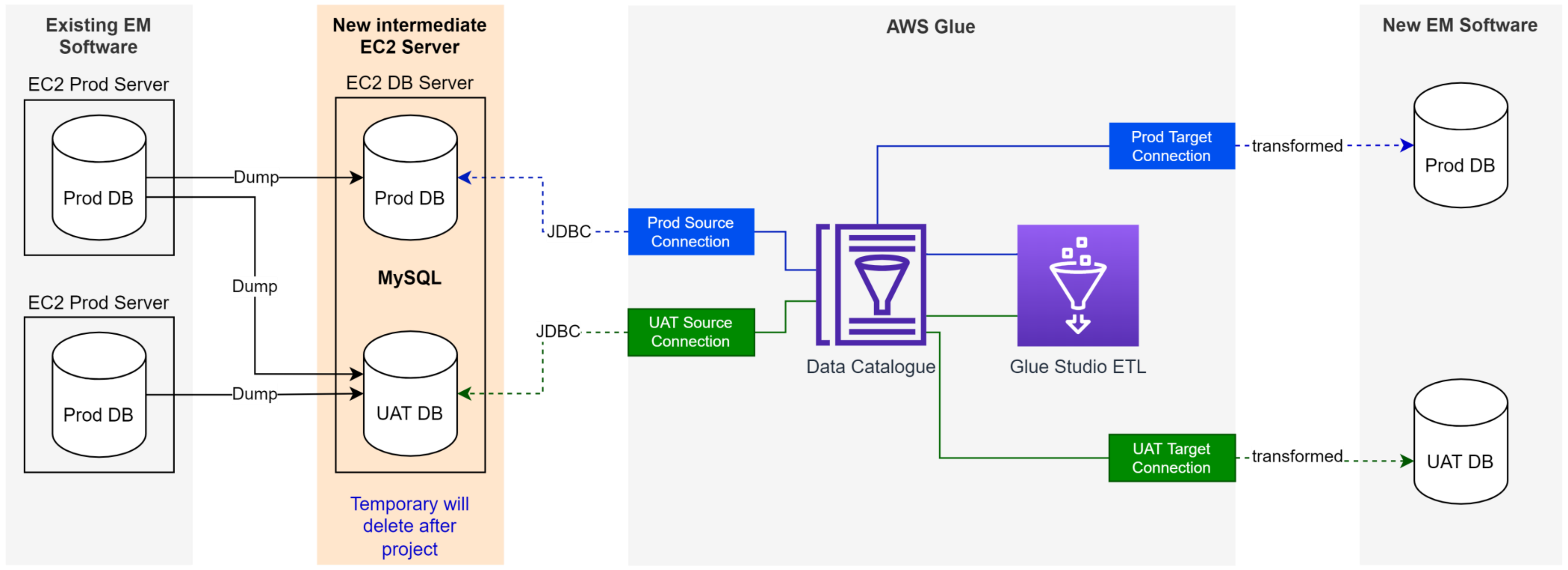
Solution Concept



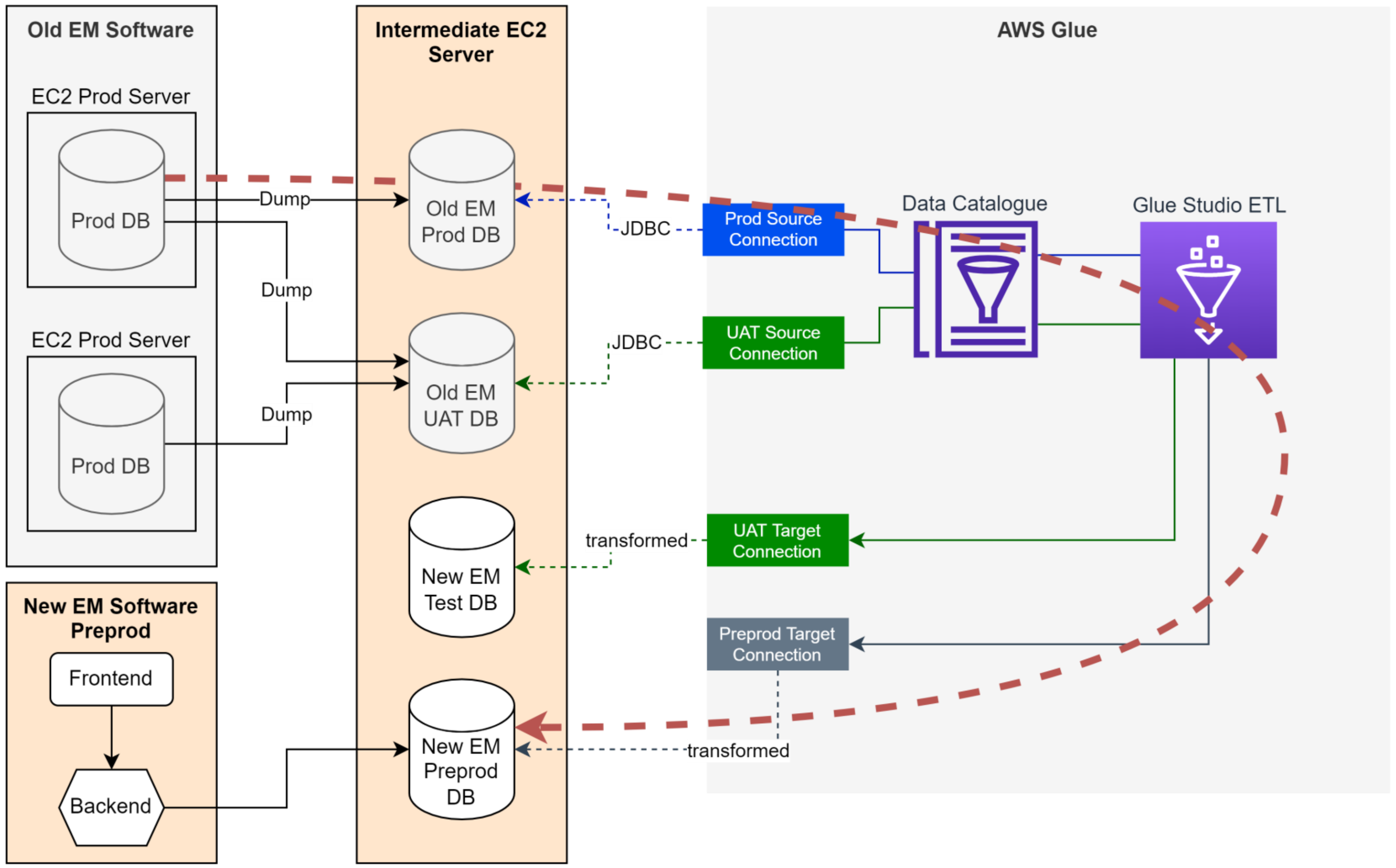
Solution Experiment



Solution POC



Solution Implementation



Business-Technology Gap

Challenge

- Business goals may not align with technological capabilities, leading to unrealistic expectations or underutilization of Technology.

Solution

✓ Solution architects translate business needs into actionable technology plans, ensuring that Technology delivers value to the business.

Data-Application Gap

Challenge

- Applications may not have consistent access to high-quality data, resulting in poor functionality or performance.

Solution

✓ By designing a robust data architecture that supports application requirements, solution architects provide reliable data access.

Application-Technology Gap

Challenge

- The chosen technical infrastructure may not fully support application needs, causing performance issues or higher costs.

Solution

✓ Solution architects ensure that the technology stack aligns with application performance requirements and scalability needs.

Business-Application Gap

Challenge

- Applications may not effectively support business processes due to inadequate functionality or poor user experience.

Solution

✓ Solution architects create application architectures that directly reflect business workflows and objectives.

Making Technical Decision

Weighted Scoring – Decision Making Technique based on data

Steps for Using Weighted Scoring

1. **Identify Evaluation Criteria**
   - List all the criteria that are important for the decision. These can include factors like cost, scalability, performance, security, ease of integration, maintainability, and user experience.
2. **Assign Weights to Criteria**
   - Determine the relative importance of each criterion. Assign a weight to each criterion, ensuring that the total weight sums to 100% or 1.0.
3. **Rate Each Option**
   - For each option, rate how well it meets each criterion on a consistent scale (e.g., 1 to 5, where 5 is the best and 1 is the worst).
4. **Calculate Weighted Scores**
   - Multiply the rating of each option by the weight of the corresponding criterion. Sum these values to get the total weighted score for each option.
5. **Compare and Select the Best Option**
   - Compare the total weighted scores of all options. The option with the highest score is typically the best choice, considering the weighted importance of each criterion.

Selecting Video Streaming Service for HEAL

Example Scenario

Step 1: Identify Evaluation Criteria

Criteria
Web JavaScript SDK support
Native Android SDK Support
Native iOS SDK Support
Cost-effectiveness
HIPAA Compliant
Easy Integration
Maintenance-free

Step 2: Assign Weights to Criteria

Criteria	Weight
Web JavaScript SDK support	20%
Native Android SDK Support	20%
Native iOS SDK Support	20%
Cost-effectiveness	10%
HIPAA Compliant	5%
Easy Integration	10%
Maintenance-free	15%

Step 3: Rate Each Option (1 to 5)

Criteria	daily.co	doxy.me	twilio.com	Infobip.com	antmedia.io
Web JavaScript SDK support	5	5	5	5	5
Native Android SDK Support	5	0	5	5	5
Native iOS SDK Support	5	0	5	5	5
Cost-effectiveness	3	2	4	1	5
HIPAA Compliant	5	5	5	5	1
Easy Integration	5	5	5	5	1
Maintenance-free	5	5	5	5	0

Step 4: Calculate Weighted Scores

Criteria	Weight	daily.co	doxy.me	twilio	Infobip	antmedia
Web JavaScript SDK support	20%	5x20%=1	5x20%=1	5x20%=1	5x20%=1	5x20%=1
Native Android SDK Support	20%	5x20%=1	0x20%=0	5x20%=1	5x20%=1	5x20%=1
Native iOS SDK Support	20%	5x20%=1	0x20%=0	5x20%=1	5x20%=1	5x20%=1
Cost-effectiveness	10%	3x10%=0.3	2x10%=0.2	4x10%=0.4	1x10%=0.1	5x10%=0.5
HIPAA Compliant	5%	5x5%=0.25	5x5%=0.25	5x5%=0.25	5x5%=0.25	1x5%=0.05
Easy Integration	10%	5x10%=0.5	5x10%=0.5	5x10%=0.5	5x10%=0.5	1x10%=0.1
Maintenance-free	15%	5x15%=0.75	5x15%=0.75	5x15%=0.75	5x15%=0.75	0x15%=0
	Total Score =	4.8	2.7	4.9	4.6	3.65

Step 5: Compare and Select the Best Option

Option	
daily.co	4.8
doxy.me	2.7
twilio	**4.9**
Infobip	4.6
antmedia	3.65

twilio got the highest total weighted score and would be considered the best choice based on the criteria and weights assigned.

Benefits of Weighted Scoring

- **Objective Decision-Making**
  - › Reduces bias by using quantifiable data.
- **Transparency**
  - › Makes it clear why a particular option was chosen.
- **Flexibility**
  - › Can adjust weights and criteria as needed to reflect changing priorities.
- **Stakeholder Buy-In**
  - › Facilitates discussions and consensus among stakeholders.

Data Modeling

Creating a visual representation of a system's data and its relationships

Key Concepts

- Entity
  - a real-world object
- Attribute
  - property or characteristic of an entity
- Relationship
  - how entities interact with each other
- Primary Key (PK)
  - unique identifier for an entity
- Foreign Key (FK)
  - creates a relationship between two tables

Three Modeling Levels

- Conceptual
  - Purpose: High-level view of system. Focuses on identifying and relationships.
  - Components: Entities and Relationships
- Logical
  - Purpose: Adds more detail to the conceptual model. Define attributes, primary keys and foreign keys
  - Components: Entities, attributes, primary keys, foreign keys and relationships.
- Physical
  - Purpose: Actual Implementation
  - Components: Tables, columns, data types, primary keys, foreign keys, indexes, constraints.

Implementation and Tools

- Conceptual Modeling (Diagram)
  - https://app.diagrams.net
  - E.g.: Job Portal Conceptual Model
- Logical Modeling (DBML)
  - https://dbdiagram.io
  - E.g.: Job Portal DBML Script
- Physical Modeling
  - https://dbdiagram.io
  - Laravel Migrations & Seeding

API Fundamental

Basics of Application Programming Interface

Types of API

- **REST (Representational State Transfer)**
  - › Uses standard HTTP methods with multiple endpoints, focuses on resources and stateless communication.
- **GraphQL**
  - › Single endpoint with flexible queries, strong typing, and real-time capabilities.
- SOAP (Simple Object Access Protocol)
  - › Protocol using XML for structured information exchange with built-in security and error handling.
- RPC (Remote Procedure Call)
  - › Executes remote procedures using XML or JSON, simple to implement.
- gRPC
  - › High-performance framework using Protocol Buffers, supports multiple languages and bi-directional streaming

Open API Specification

API Definition

Basis

- A specification for building APIs.
- It offers a standardized way to describe RESTful APIs.
- Current version 3.x.x
- Defined using YAML or JSON
- Describes endpoints, methods, parameters, request bodies, responses, and authentication.

Composition

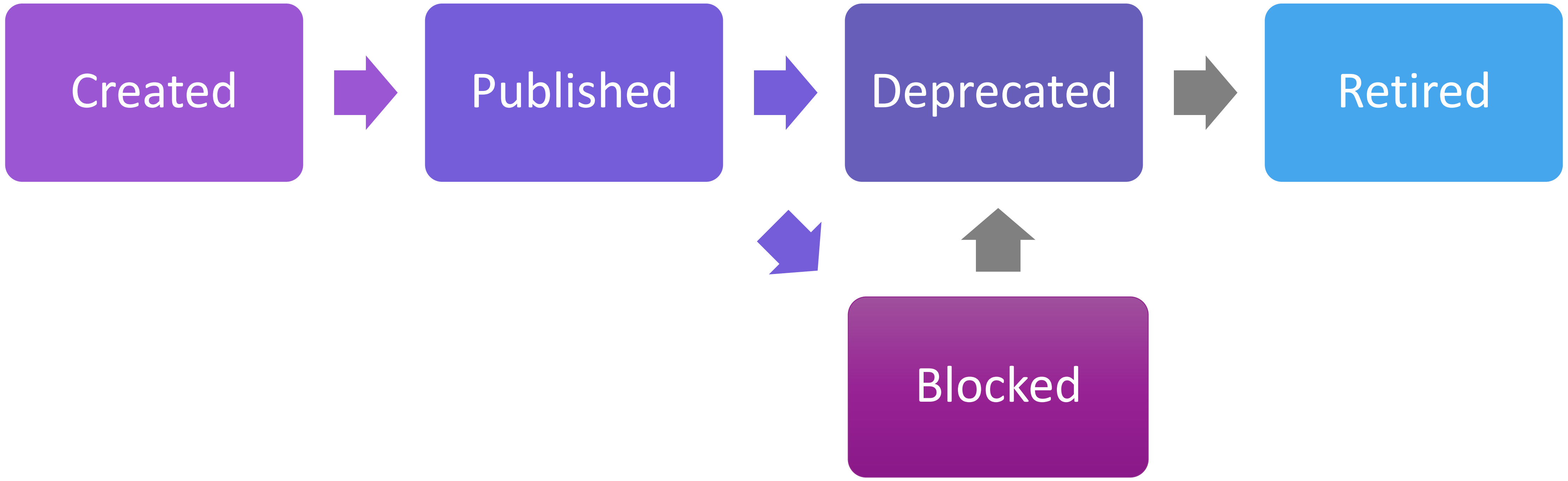
```yaml
openapi: 3.1.0
info:
tags:
paths:
components:
    schemas:
    examples:
    responses:
    securitySchemes:
```

Tools & Example

- **Swagger UI**
  - Visualization tool that renders API documentation from an OpenAPI definition.
- **Swagger Editor**
  - Helps write and validate OpenAPI definitions.
- **PostMan**
  - Define, Test, Create Documentation, convert postman collection to OpenAPI format
- **APIGit**
  - UI Designer, Documentation, SaaS platform.

Example:
https://app.swaggerhub.com/apis/aungkyawminn.me/Standard_Backend_API_Definition/1.0.0

API Lifecycle



Thanks

End of only beginning. There’s so much to learn out there …

Aung Kyaw Minn