Comprehensive Case Study : Phase G Implementation Governance

Files of this Case Study are to be referred and continued only in this sequence order, please :

- CompCaseStudyStart
- CompCasePhA
- CompCasePhB
- CompCasePhCAppArch
- CompCasePhCDataArch
- CompCasePhD
- CompCasePhE
- CompCasePhF
- CompCasePhG
- CompCasePhH

This Phase happens after the Work Products are handed over to the PMO.

Note that they may involve :

Own development by PMO
Outsourced development by Partner agencies
Software Packages – purchased / downloaded
Hardware Units re-configured or purchased
Cloud Solutions subscribed
Installation of Software on On-premise and Cloud systems
Training courses conducted by in-house or external personnel

The overall approach in Phase G, as noted by the EA, is to :

- Establish an implementation program that will enable the delivery of the Transition Architectures agreed for implementation during the Migration Planning phase

- Adopt a phased deployment schedule that reflects the business priorities embodied in the Architecture Roadmap

- Follow the organization's standard for corporate, IT, and Architecture Governance

- Use the organization's established portfolio / program management approach, where this exists

- Define an operations framework to ensure the effective long life of the deployed solution

---

Focusing on what ?

Architectural oversight ONLY on the implementation

Helping whom ?

Governance Over Realization of Solutions

Development Solutions

OUTSOURCING

IN HOUSE SOLUTION

Solution Partner

GOVERNANCE Solutions

V/S

COTS (Commercial Off the Shelf)

Custom Development

Parallel Solution Work By whom ?

Not by EA / Architects

PLATFORM

Hardware, Platform and Connectivity Solutions

Alternative Analysis

Contracts

- Types of architecture contract **Produced in Phase A, but referred in all other Phases, including Phase G**
  - 1. Statement of work
    - A standard deliverable of the ADM Phase A that includes a detailed description of the scope and approach used to conduct the architecture work
  - 2. Contract between architecture design and implementation partners **EA <-> Impl**
    - Signed contract of intent on designing and developing the enterprise architecture, or significant parts of it from partner organisations, including system integrators etc
    - This allows good management of out sourced components of the ADM. Typical contents include: - Scope, architecture principles and requirements, conformance requirements, architecture development, prioritised work plan, timeframe
  - 3. Contract between architecture function and business users **EA <-> LOB**
    - This is a signed agreement to conform to the enterprise architecture by the business users.
    - Similar in content to previous contract exception SLA and a more service architecture focus in the provision of service architecture

Produced in Phase F, but used heavily in Phase G

Three Contracts : One existed already in the form of

Statement of Architectural Work

Priorities for development teams

Acceptance Criteria

Risk Perception

Project Scope
deliverables, and constraints

deployment issues

Resources, Skills

Developers, Deployers

Solutions Deployment Implementation Plan

strategic requirements from architectural perspective

rules for conformance

timeline requirements

Irrelevant:
The implementation has no features in common with the architecture specification (so the question of conformance does not arise).

Consistent:
The implementation has some features in common with the architecture specification, and those common features are implemented in accordance with the specification. However, some features in the architecture specification are not implemented, and the implementation has other features that are not covered by the specification.

Compliant:
Some features in the architecture specification are not implemented, but all features implemented are covered by the specification, and in accordance with it.

Conformant:
All the features in the architecture specification are implemented in accordance with the specification, but some more features are implemented that are not in accordance with it.

Fully Conformant:
There is full correspondence between architecture specification and implementation. All specified features are implemented in accordance with the specification, and there are no features implemented that are not covered by the specification.

Non-conformant:
Any of the above in which some features in the architecture specification are implemented not in accordance with the specification.

Step : Confirm Scope and Priorities for Deployment with Development Management

- Reviewed such migration planning outputs and offered recommendations on deployment

- Identified Enterprise Architecture priorities – relative priories among the Work Products, for development teams

- Identified key and crucial deployment issues and make recommendations thereon

- Identified all the Building Blocks which are for replacement, update, etc. rather than fresh deployment

- Performed gap analysis on Enterprise Architecture and solutions framework, to make sure that all necessary inputs are provided to the PMO

  The gaps in the existing enterprise solutions framework need to be identified and the specific Solution Building Blocks (SBBs) required to fill these gaps will be identified by the Solution Architects. These SBBs may have a one-to-one or many-to-one relationship with the projects. The Solution Architects need to define exactly how this will be done. There may be other projects working on these same capabilities and the Solution Architects need to ensure that they can leverage best value from these investments.

- Produceed a gap analysis report based on the above.

Step : Identify Deployment Resources and Skills
Deployment Resources, Skills

The project resources for the Transition Architecture in the Ecommerce related Project Charters include :

Development resources

Each of the human resource personnel involved in development or installation may have to be educated in the overall Enterprise Architecture deliverables and expectations from the specific development and implementation projects. This means appraising them of the Work Products which enables Micro services slices.

These in turn may involve development time resources of :

Workbenches comprising of Studios and IDEs, Screen Painters, Compilers, Interpreters,. Debuggers, Bug Trackers, Linkers, Profilers, Build Tools, Source Code Control systems, Package Mangers, Dependency Managers, Test Benches, .. many more like this

For the Coding Teams, following resources may need to be brought to their attention :

- Coding Academies and Boot Camps.
- Online Learning Communities.
- Self-Taught Resources.

- Software Development Conferences.
- Books, Magazines and Online Publications.
- Podcasts and Web Casts; You Tube style videos
- Thought-leaders, Influencers and Blogs
- Github and similar ones

Software Engineering
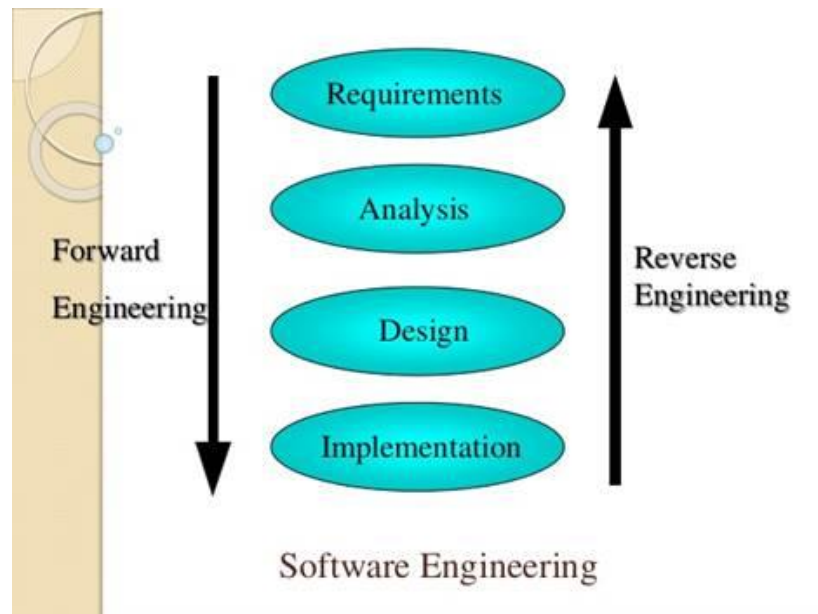
The EA Identifies system development methods required for solutions development

The method should ideally be able to interoperate with the architecture outputs; for example, generate code from architecture artifacts delivered to date.

This could be achieved through the use of modeling languages used for the Enterprise Architecture development that may be captured as inputs to the systems development tools and thereby reduce the cost of solutions development.

The UML Tool used by the Architects of this Enterprise is Sparx Systems Enterprise Architect. It has following code generation facilities :
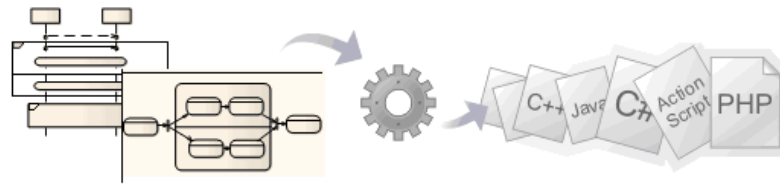
It can generate code for software, system and hardware description languages directly from Structural Model of Class Diagram and from Behavioral models, such as State Machine (State Chart), Sequence (Interaction) and Activity diagrams.

The supported languages include :

- ActionScript
- C
- C# (for.NET 1.1 and .NET 2.0)
- C++ (and .NET managed extensions)
- Delphi
- Java (including Aspects and Generics)
- PHP
- Python
- Visual Basic
- Visual Basic .NET
- Hardware Description Languages - HDLs (Ada, VHDL more.)

The Coding Team was made aware that the Code templates in this Tool are specify customizable transformations from UML elements into a target programming language.

Training sessions were arranged on this so that Coding Team was able to tailor the generated source code to suit the accepted standards.

Additionally, the following were the areas where support was extended :

Priorities for development teams

Acceptance Criteria

Risk Perception

deployment issues

Software development
web

Project Scope
deliverables, and constraints

Risk ID	Risk	Preliminary Risk			Mitigation	Residual Risk		
		Effect	Frequency	Impact		Effect	Frequency	Impact

Step : Guide Development of Solutions as per Deployment Implementation Plans

Formulate **project recommendation**

For each separate implementation and deployment project,

did the following :

o Documented **scope of individual project** in impact analysis

o Documented **strategic requirements** (from the architectural perspective) in impact analysis

o Documented **change requests** (such as support for a standard interface) in impact analysis

o   Documented **rules for conformance** in impact analysis

o   Documented **timeline requirements** from roadmap in impact analysis

▶   Documented and formalized the two Architecture Contracts which were drafted in Phase F

o   Obtained signature from all developing organizations (PMO and Operations Management) and sponsoring organization (LOB)

▶   Updated the Enterprise Continuum directory and repository for solutions in the pipeline

▶   Guided development of business & IT operating models for services and this continues as an ongoing activity

▶   Provided service requirements (SLAs, OLAs) derived from Enterprise Architecture

⊡ Guided definition of business & IT operational requirements

⊡ Thus carried out Gap Analysis between the Solution Architecture and operations

---

Typically, a build automation tool (such as Jenkins or CircleCI) builds a service artifact and pushes it to an artifact repository. An artifact repository might be a dedicated tool —for example, Docker provides a registry for storing images —or a generic file storage tool, such as Amazon S3.
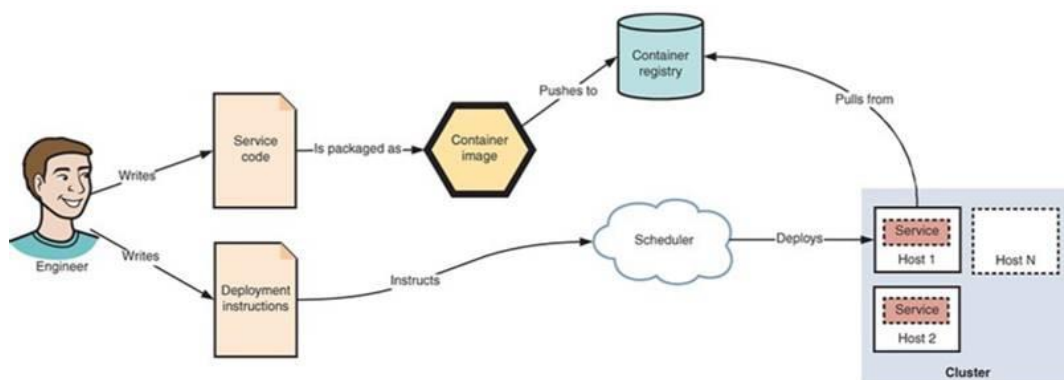
The process of deploying service code to a cluster scheduler
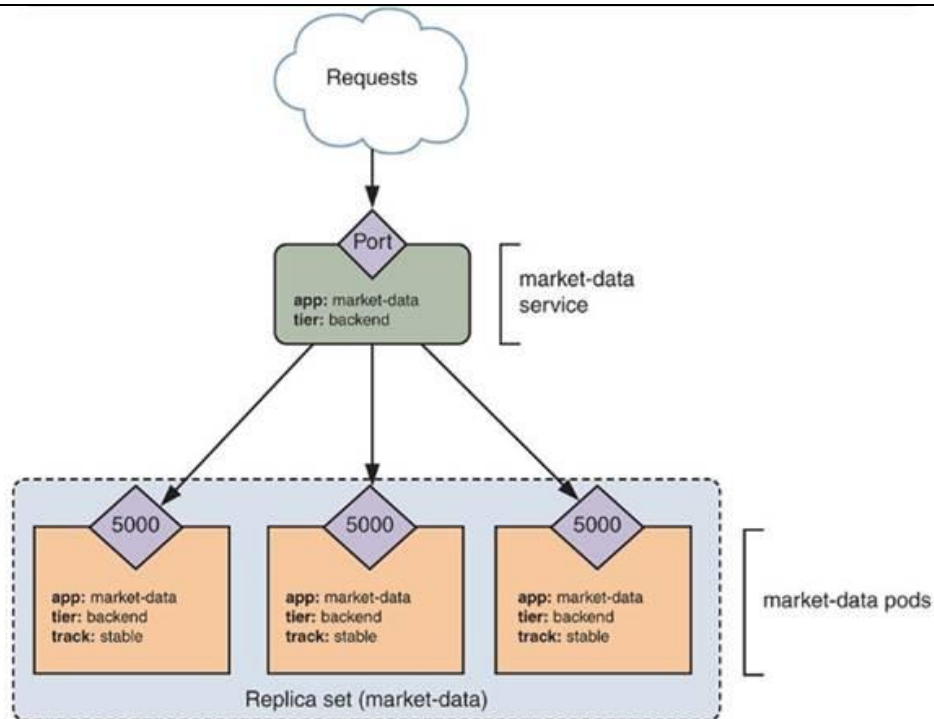
Requests made to a service are forwarded to pods that match the label selector of the service
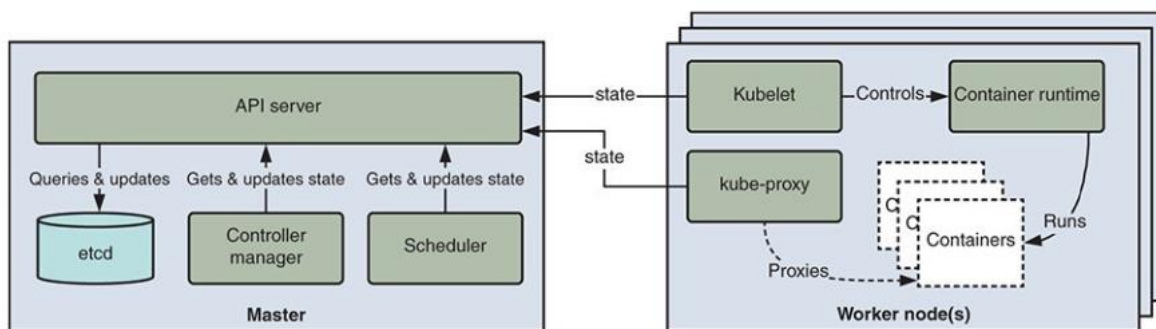
Components of the master and worker nodes in a Kubernetes cluster
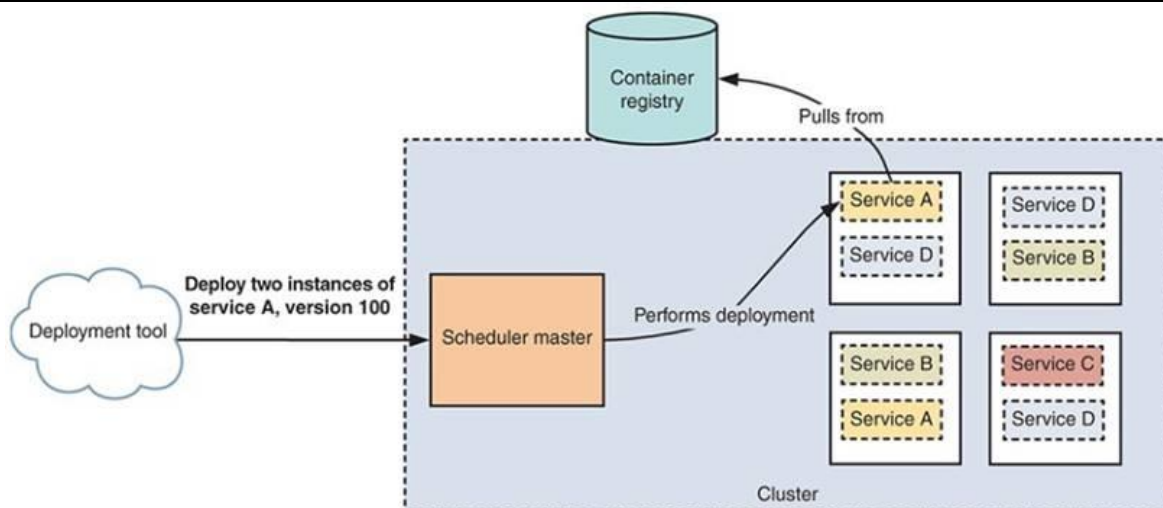
A container scheduler executes containers across a cluster of nodes, balancing the resource needs of those nodes
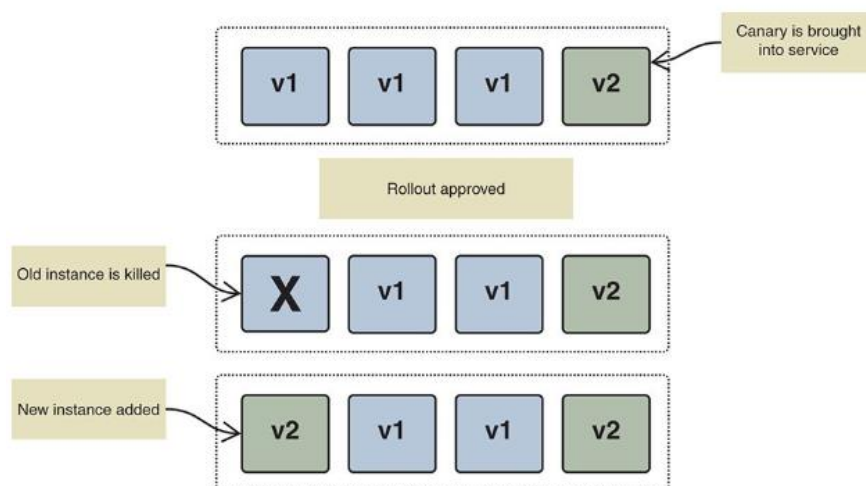
Stages of a rolling deploy, beginning with a canary instance
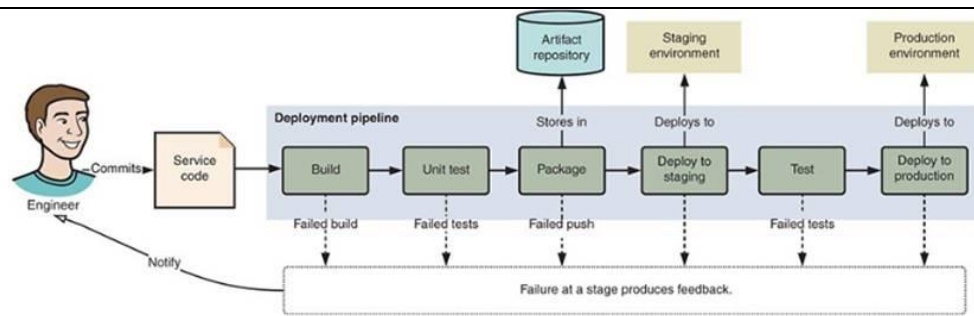
An example deployment pipeline builds, validates, and deploys a commit to production, providing feedback to engineers
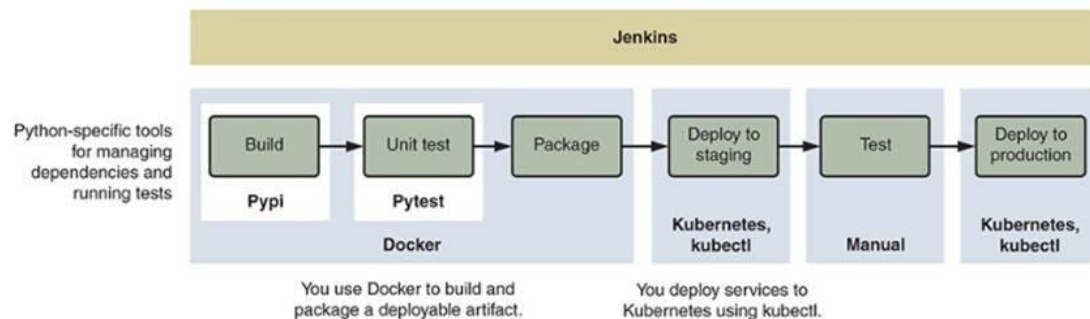
The deployment pipeline which combines multiple tools dependent on the tech stack and target deployment platform

Step : Perform EA Compliance Reviews

Performed Development time Compliance Review

 To catch errors in the project architecture early

Advised on Best Practices at implementation Level

Idioms of Languages

Coding time performance enhancement

Installation time : Configuration Practices

Fine tuning during Installation and Trial runs

Locating Enterprise services instead of application specific ones

Cross Development Team sharing Resources, collaboration, synergy

Guidance on key criteria for procurement activities : (COTS) product; RFI/RFP documents

Step : Implement Business and IT Operations

The PMO and Operations Teams keeps on carrying out the deployment projects, as per : IT services delivery implementation; business services delivery implementation; skills development & training implementation; communications documentation publication

The EA focused on publishing new Baseline Architectures to the Architecture Repository and update other impacted repositories, such as operational configuration management stores

Step : (When a portion of the Project Charter, say as per Continuous Delivery schedules)
Perform post-implementation review the and close the portion of the implementation

Conducted post-implementation reviews
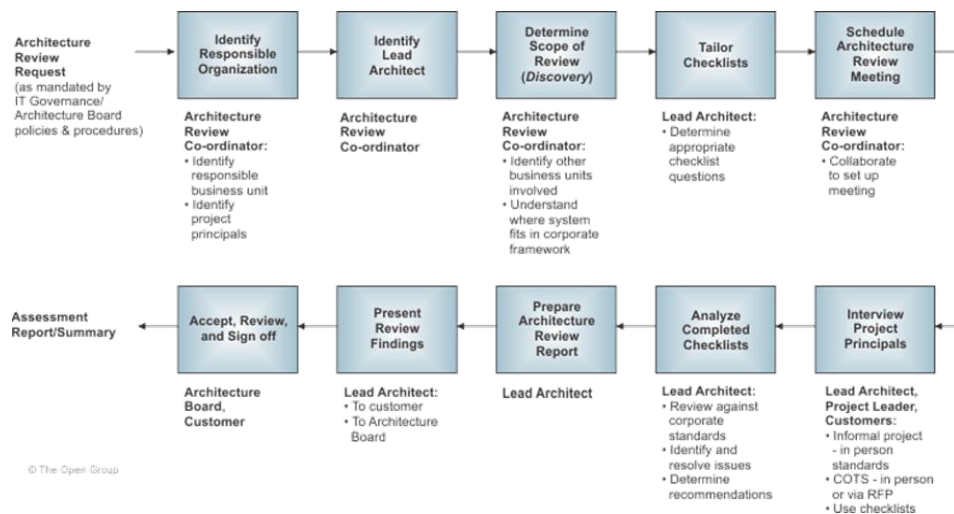
Documents Updated in this Phase :

Contract between architecture design and implementation partners

- Signed contract of intent on designing and developing the enterprise architecture, or significant parts of it from partner organisations, including system integrators etc

- This allows good management of out sourced components of the ADM. Typical contents include: - Scope, architecture principles and requirements, conformance requirements, architecture development, prioritised work plan, timeframe

Contract between architecture function and business users

- This is a signed agreement to conform to the enterprise architecture by the business users.

- Similar in content to previous contract exception SLA and a more service architecture focus in the provision of service architecture

Artifact produced in this Phase in support of Governance :

Compliance Review Report
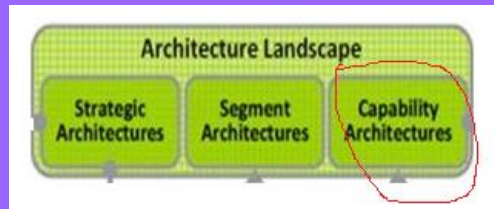
Compliance Assessment is created by the Architecture Team and is used to monitor the way the architecture is realized by the implementation teams. These teams are typically under pressure to complete their project work and achieve the project's desired outcomes which often means that corners are cut and rules are bent.

The Architect needs to work with the team and understand these pressures but also needs to have a governance role that can assess whether the fundamental tenets of the Architecture are being implemented. A Compliance Assessment is a useful mechanism for documenting the observations made about the implementations team's compliance with the Architecture.

It is a way of formally recording the findings of an Architectural Review of the implementation initiatives to unsure that they are implementing the architecture as it was specified in the architecture vision and the definitions of the architecture in the models describing the Business, Information, Application and Technology Architectures.
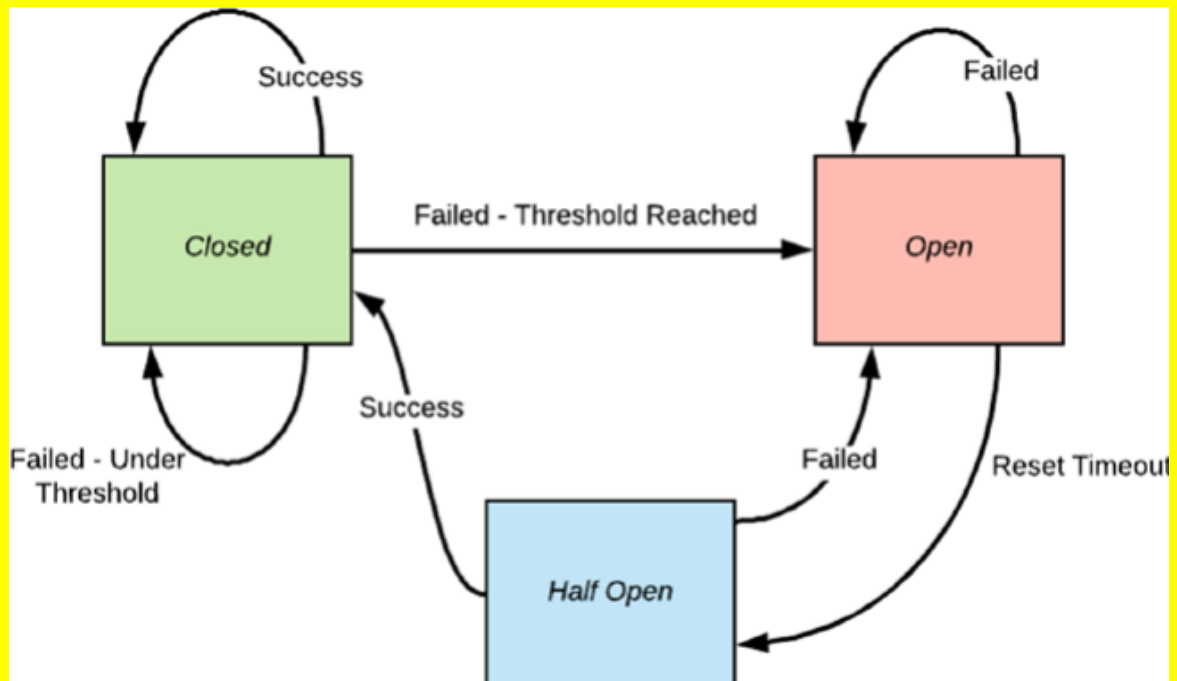
Areas where Implementation time Governance has to focus, say in a Microservices centric Realization
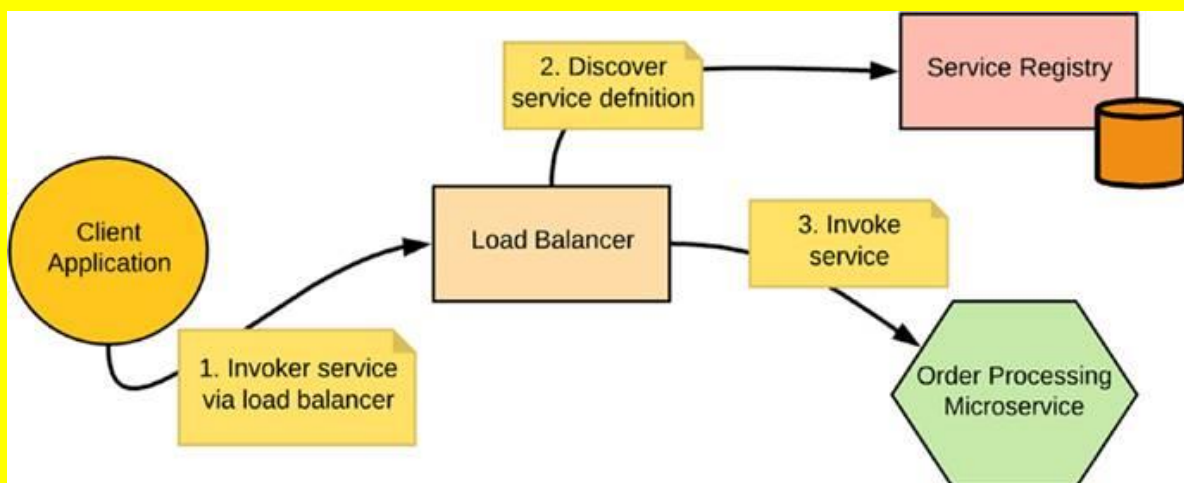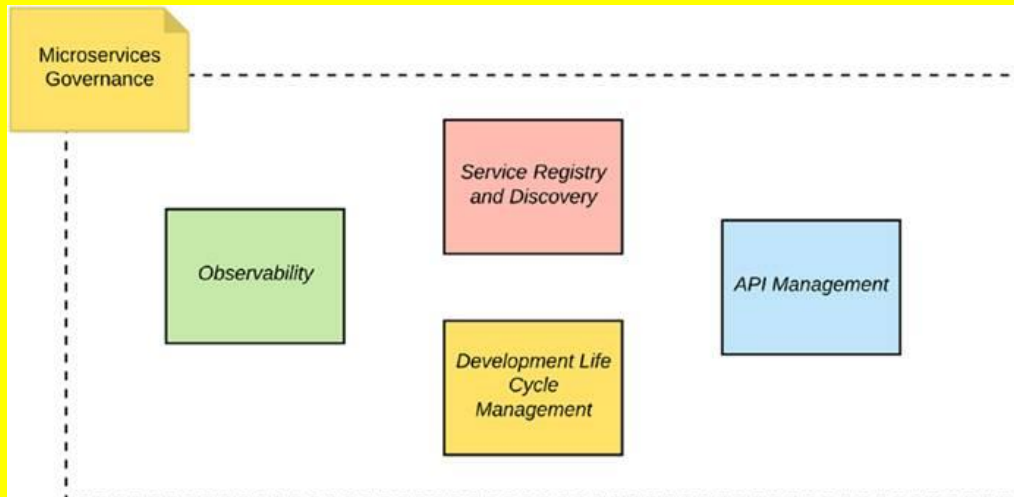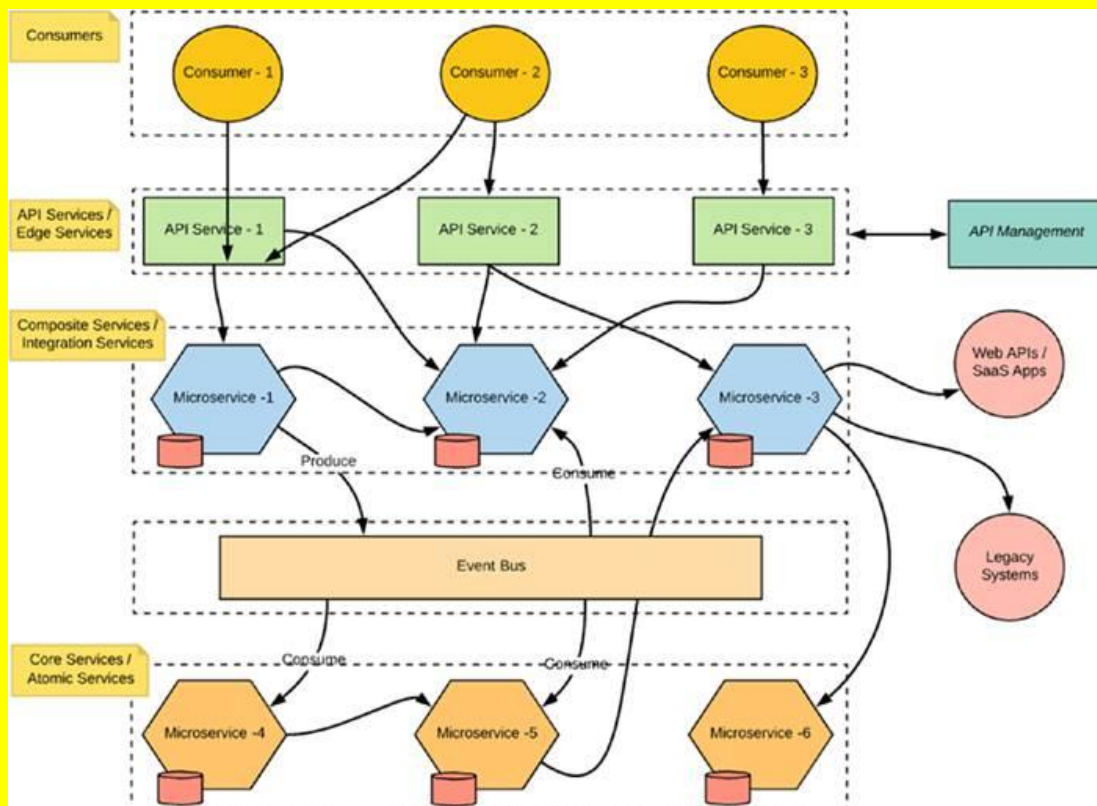

Server-side service discovery
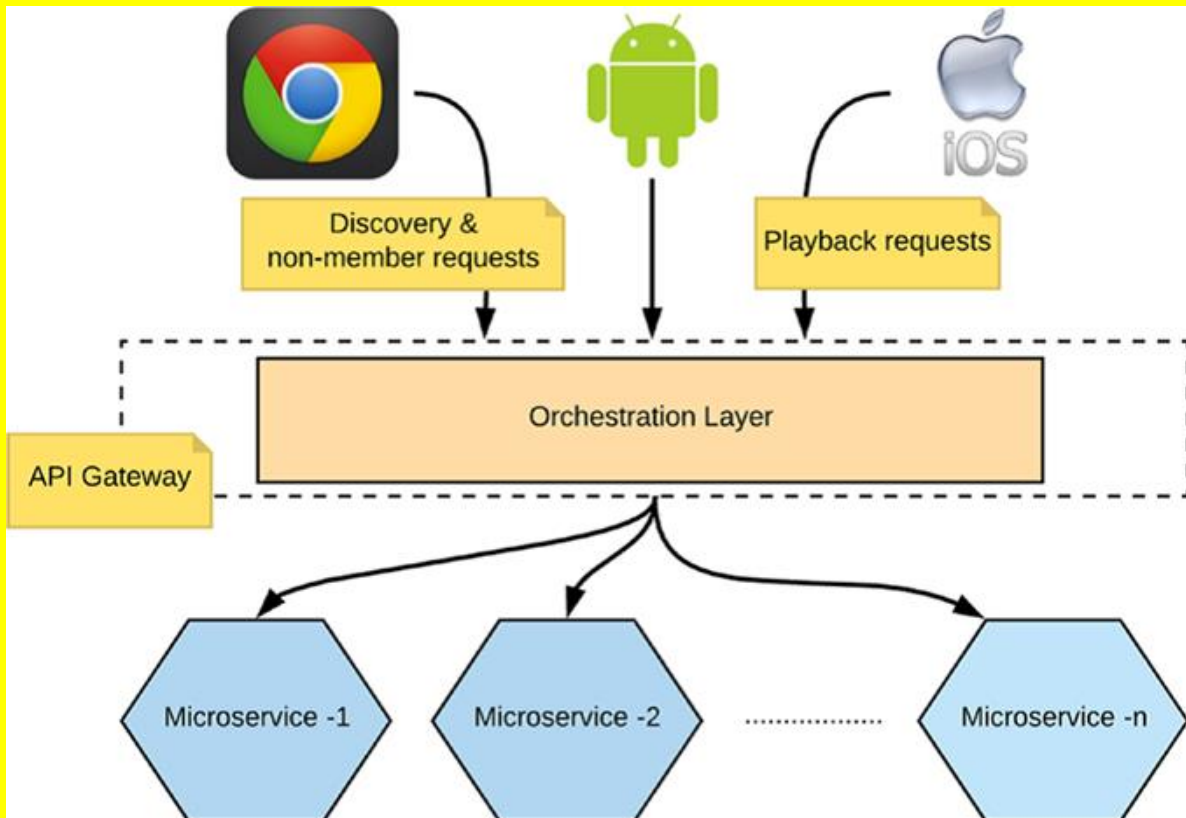
Gateway level : Possible issues of Implementation Governance

Risk and Failure analysis : Phase G task tried though Circuit
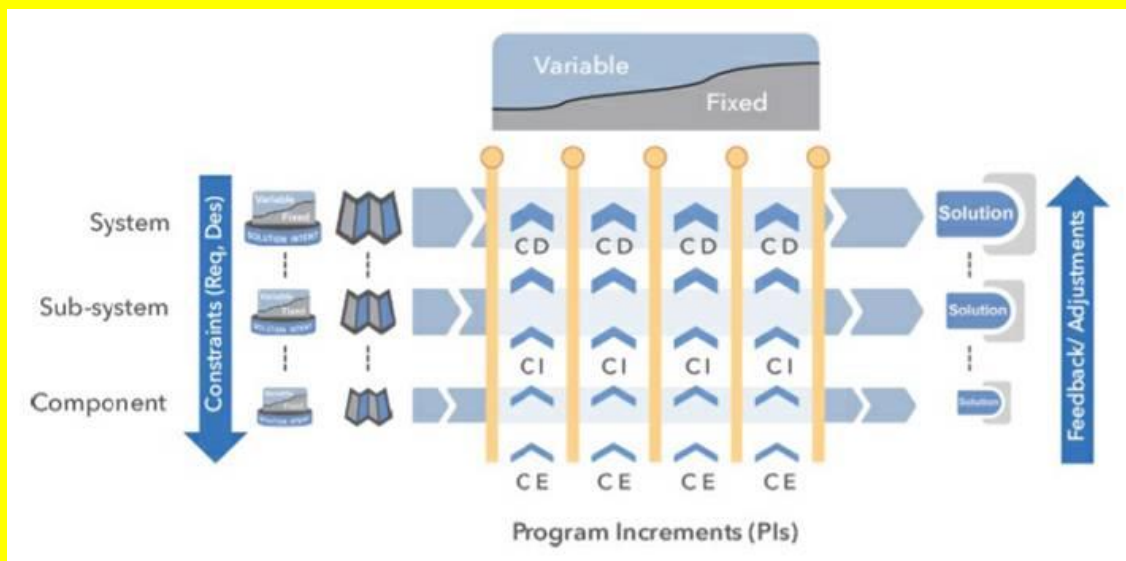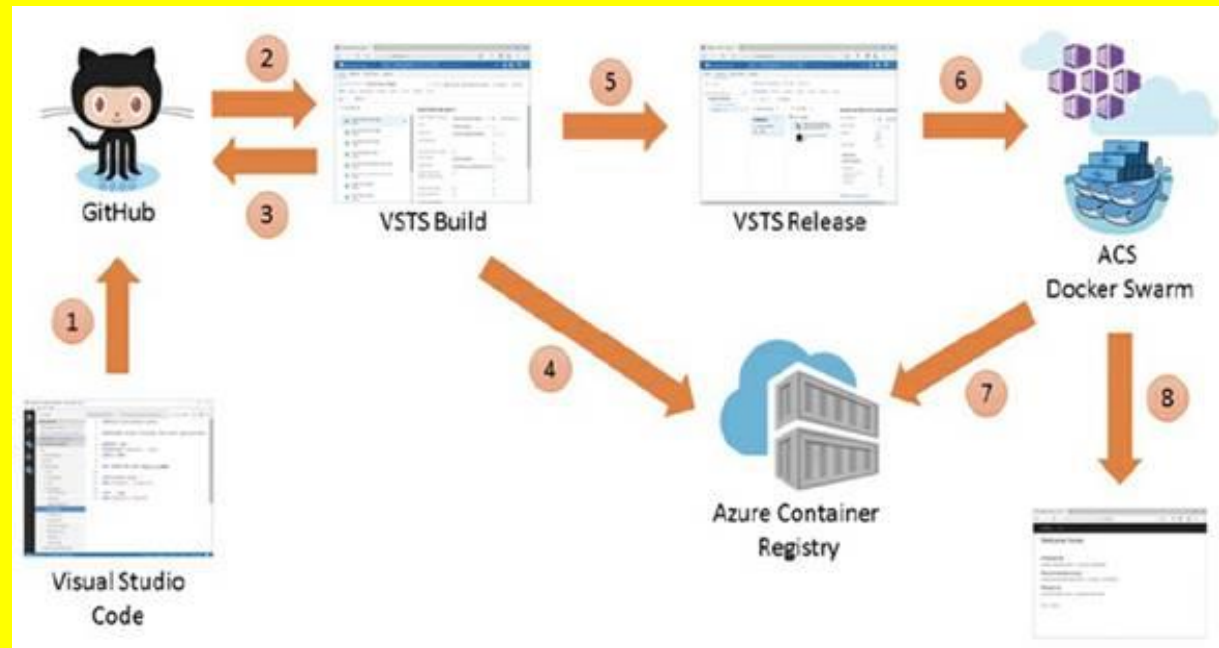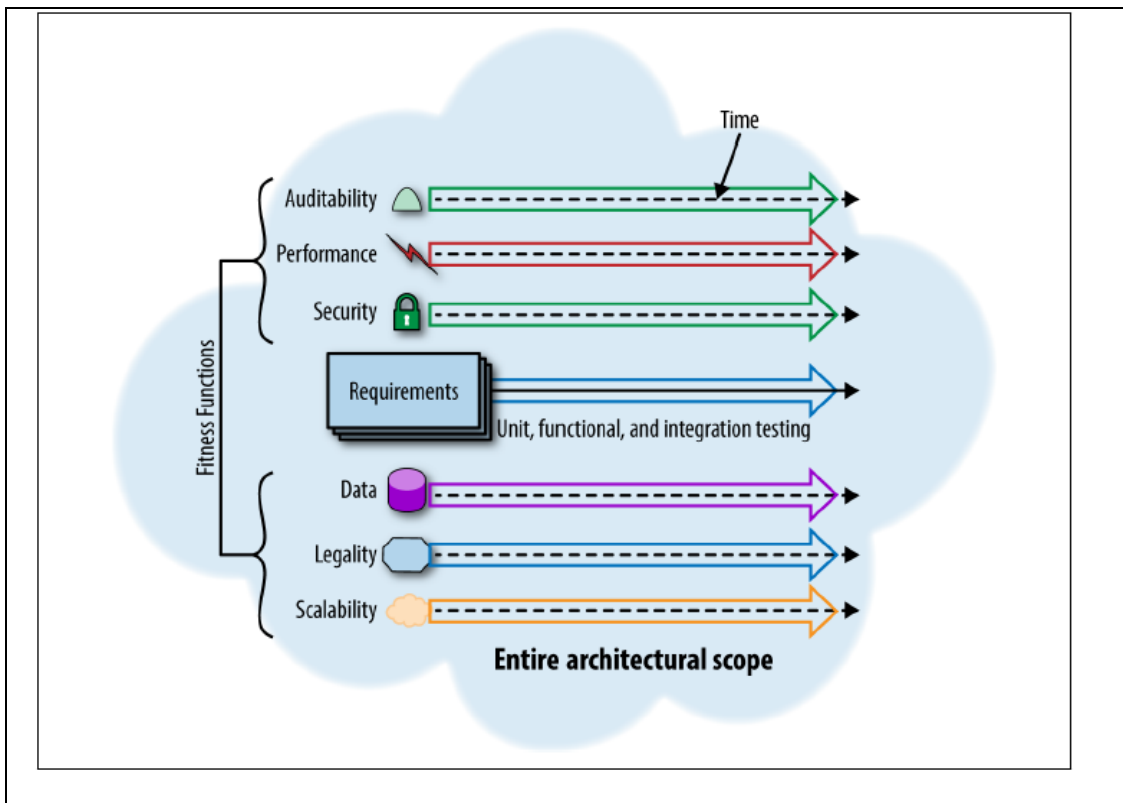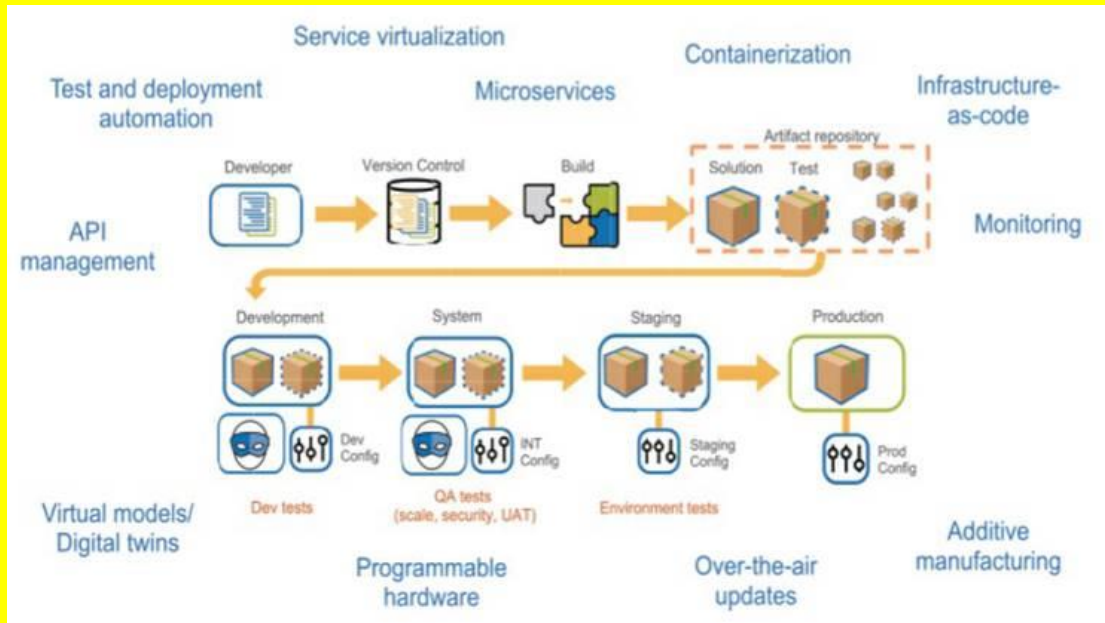
Breaker Pattern

Some governance aspects, such as observability, are extremely crucial when we build microservice integrations.

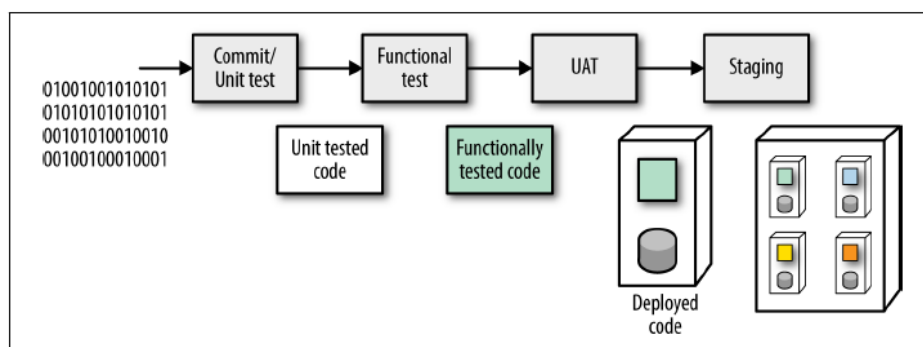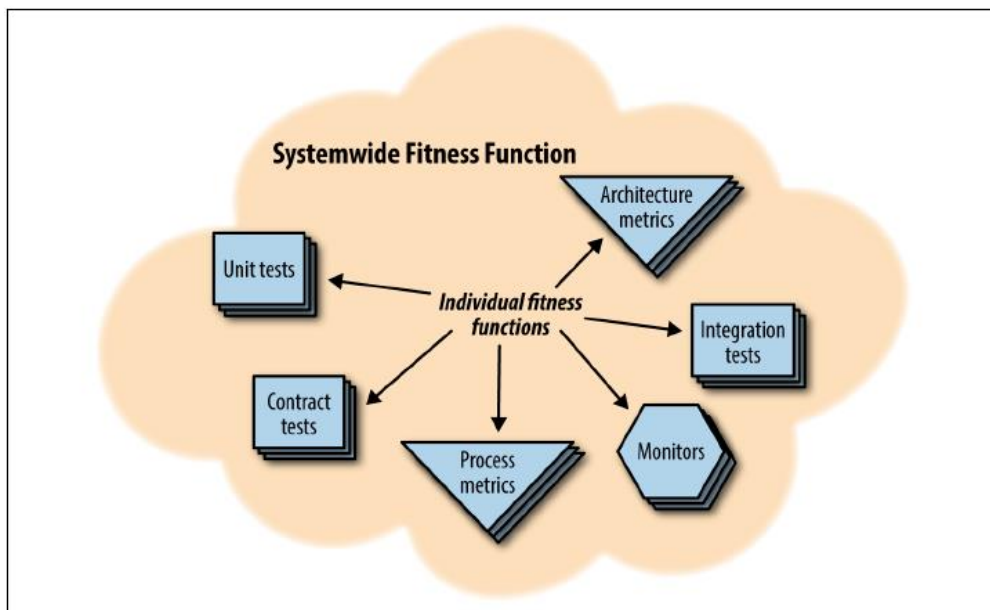As part of the integration service development technology, we need to have a seamless way to integrate existing observability tools to get the metrics, tracing, logging, service visualization, and alerting for the integration services.

In Case Study

The following style of CD/CI (Continuous Delivery and Continuous Integration), which is already in vogue, will have to be strictly followed now :

Program Increments (PIs)

Service virtualization

Test and deployment automation

Microservices

Containerization

Infrastructure-as-code

API management

Monitoring

Developer → Version Control → Build → Artifact repository (Solution, Test)

Development → System → Staging → Production

Dev Config — Dev tests
INT Config — QA tests (scale, security, UAT)
Staging Config — Environment tests
Prod Config

Virtual models/ Digital twins

Programmable hardware

Over-the-air updates

Additive manufacturing

Time

Fitness Functions
Auditability
Performance
Security
Requirements
Unit, functional, and integration testing
Data
Legality
Scalability

Entire architectural scope

Systemwide Fitness Function

Unit tests

Architecture metrics

Individual fitness functions

Contract tests

Process metrics

Integration tests

Monitors

01001001010101
01010101010101
00101010010010
00100100010001

Commit/ Unit test → Functional test → UAT → Staging

Unit tested code

Functionally tested code

Deployed code

Deployment pipeline stages
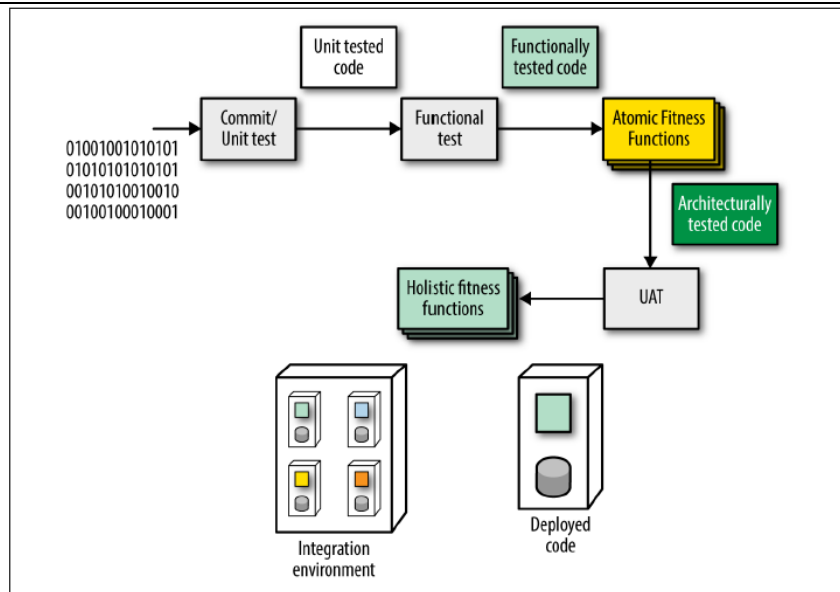
A deployment pipeline with fitness