JUPITER PRIME
Consulting • Training • Advisory
جوبيتر برايم للاستشارات والتدريب

COURSE BROCHURE

Advanced Solution Architecture Design Training Course

30 May – 10 June 2027
Dubai

jupiterprime.uk
info@jupiterprime.uk
Registered in the United Kingdom and Saudi Arabia

Advanced Solution Architecture Design Training Course

Ref: 777_29380 **Date:** 30 May – 10 June 2027 **Location:** Dubai **Fees:** 8500 GBP

Solution Architecture Design is a 2-week advanced training course for senior developers and technical leads stepping into solution architect roles, built around a solution architecture document for a real business system. Organisations now run hybrid estates of cloud services, packaged platforms and bespoke code, and the cost of a poorly reasoned structure surfaces quickly in outages, rework and integration debt. Jupiter Prime Training delivers Solution Architecture Design for architects who must justify every structural choice.

Why Solution Architecture Design Matters Now

Three forces shape solution architecture today. Cloud platforms offer dozens of managed services for every capability, so the choice between them is a design decision with long consequences. Integration has moved from point-to-point links to events, APIs and streaming, raising questions of coupling and ownership. Product owners and security officers expect quality attributes such as resilience, latency and cost to be traded openly rather than discovered in production. An organisation that handles these choices well ships change safely and cheaply; one that handles them badly accumulates fragile dependencies. The course aligns architecture work with enterprise IT governance practice such as COBIT 2019 and service management practice such as ITIL 4.

Who Gains Most from Solution Architecture Design

- Engineers responsible for turning business requirements into component and deployment structures
- Technical leads accountable for integration contracts between internal and partner systems
- Platform specialists who select cloud services, hosting models and runtime topologies
- Designers who own security, availability and performance characteristics of a solution
- Senior developers who present structural options to product and investment boards

Delivery groups that send their lead engineer, integration specialist and platform owner together gain most, because they leave sharing one vocabulary for options, quality attributes and recorded decisions.

Solution Architecture Design Capabilities

Participants practise moving from ambiguous business intent to a defensible structure that engineers can build and operators can run.

- Translate business capabilities and use cases into architecturally significant requirements — product owners
- Specify quality attribute scenarios with measurable stimulus and response — operations and security leads
- Weigh cloud hosting and integration patterns against cost, coupling and resilience — investment boards
- Design component, data and deployment views for one coherent solution — engineering teams
- Record architecture decisions with context, options and consequences — future maintainers
- Communicate structural options and risks in viewpoints suited to each audience — sponsors and steering groups

Key Solution Architecture Design Terms

- Architecturally significant requirement — a need whose satisfaction shapes structure, such as a latency limit or a regulatory data boundary, rather than a single screen or report.
- Quality attribute scenario — a testable statement of a stimulus, environment and expected measured response for a property such as availability or modifiability.
- Architecture decision record — a short document capturing one structural choice, its context, the options weighed and the consequences accepted.
- Bounded context — a boundary within which a domain model and its terms stay consistent, used to split services and ownership.
- Fitness function — an automated check that measures whether a running system still meets a chosen architectural characteristic.
- Architecture runway — the technical foundations prepared ahead of features so that planned change can land without structural rework.

Solution Architecture Design Day by Day

Day 1: The Solution Architect Role and Context

- Architect Role Boundaries — Where Solution Design Meets Enterprise Direction
- Stakeholder And Concern Mapping — Whose Viewpoint Shapes Each Design Choice
- System Context Diagram — Scope Of The Solution Under Design
- Current Estate Assessment — Constraints Inherited From Existing Platforms
- Architecture Principles Catalogue — Guardrails For Later Structural Choices

Day 2: From Requirements to Architectural Drivers

- Business Capability Map — Which Capabilities The Solution Must Enable
- Use Case And Event Storming Sessions — Domain Flows Worth Modelling
- Architecturally Significant Requirements Filter — Drivers That Shape Structure
- Constraint Register — Fixed Technology, Budget And Compliance Limits
- Driver Prioritisation Matrix — Ranking Drivers Before Option Design

Day 3: Quality Attributes and Their Trade-offs

- Quality Attribute Scenarios — Measurable Targets For Availability And Latency
- Utility Tree — Ranking Attributes By Business Importance
- Architecture Trade-off Analysis Method — Sensitivity And Trade-off Points
- Security By Design Threat Modelling — Trust Boundaries And Attack Surfaces
- Cost As An Architectural Attribute — Spend Profile Of Each Option

Day 4: Domain Decomposition and Service Boundaries

- Domain-Driven Design Bounded Contexts — Splitting Ownership Along Domain Lines
- Context Mapping Relationships — Upstream And Downstream Team Dependencies
- Modular Monolith Versus Microservices — Granularity Suited To Team Size
- Data Ownership Per Service — Avoiding Shared Database Coupling
- Strangler Fig Migration Pattern — Replacing Legacy Functions Incrementally

Day 5: Cloud Hosting and Deployment Patterns

- Hosting Model Comparison — Virtual Machines, Containers And Serverless Functions
- Well-Architected Review Pillars — Checking Reliability, Security And Efficiency
- Multi-Zone And Multi-Region Topologies — Recovery Objectives And Their Cost
- Infrastructure As Code Blueprints — Repeatable Environments For Each Stage
- Landing Zone And Account Structure — Isolation Of Workloads And Teams

Day 6: Integration and Messaging Patterns

- API Design Styles — REST, gRPC And Asynchronous Contracts Compared
- Event-Driven Integration — Publish And Subscribe Versus Request And Reply
- Enterprise Integration Patterns — Routers, Translators And Idempotent Receivers
- Saga And Outbox Patterns — Consistency Across Distributed Transactions
- API Gateway And Service Mesh — Where Cross-Cutting Policies Live

Day 7: Data Architecture and Resilience

- Polyglot Persistence Choices — Relational, Document And Time-Series Stores
- CQRS And Read Models — Separating Query Load From Writes
- Caching Strategy — Freshness Versus Response Time Trade-offs
- Circuit Breakers, Retries And Bulkheads — Containing Failure Between Components
- Observability Design — Logs, Metrics And Traces For Diagnosis

Day 8: Recording and Governing Architecture Decisions

- Architecture Decision Records — Context, Options And Accepted Consequences
- Option Scoring Against Drivers — Transparent Selection Between Candidate Designs
- Fitness Functions In Pipelines — Automated Guarding Of Key Characteristics
- Architecture Review Board Submissions — Evidence Reviewers Expect To See
- Technical Debt Register — Deliberate Shortcuts And Their Repayment Path

Day 9: Communicating Architecture to Stakeholders

- C4 Model Diagrams — Context, Container, Component And Code Levels
- Viewpoints For Executives, Engineers And Operators — Matching Detail To Audience
- Architecture Runway Roadmap — Sequencing Foundations Ahead Of Features
- Risk And Assumption Log — Open Questions Raised With Sponsors
- Design Walkthrough Rehearsal — Defending Choices Under Challenge

Day 10: Practising the Solution Architecture Document

- Driver And Scenario Set Applied — Requirements Section For The Chosen System
- Service Boundary And Integration View Applied — Structure Of The Chosen System
- Cloud Deployment And Resilience View Applied — Runtime Topology And Recovery
- Peer Trade-off Review — Challenging Decision Records Before Sign-off
- Solution Architecture Document — Completed Views, Decisions And Roadmap

Applying Solution Architecture Design After the Course

- Run a utility tree session with product and operations leads for a live system, then turn the top five quality attribute scenarios into fitness functions in its delivery pipeline.
- Introduce architecture decision records for the next significant structural choice in your area, scoring at least two options against the ranked drivers and sharing the record with the review board.

Solution Architecture Design Questions FAQ

What is Solution Architecture Design and why does it matter now?

Solution Architecture Design is the discipline of turning business needs into a buildable structure of components, integrations and deployments. It matters now because cloud services and event-driven integration multiply the options, and each structural choice fixes cost, resilience and the speed of later change.

Which roles gain most from Solution Architecture Design?

Senior developers, technical leads and platform engineers moving into solution architect roles gain most from Solution Architecture Design. You benefit if you already build systems and now need to justify structural options, quality attribute trade-offs and integration contracts to product owners and review boards.

How does Solution Architecture Design differ from an enterprise architecture course?

Solution Architecture Design works on one business system at a time, from drivers to deployment. An enterprise architecture course looks across the whole portfolio of capabilities and platforms, while a software design course stays inside a single codebase rather than spanning services, data stores and cloud topology.

What do you bring back from Solution Architecture Design?

You bring back a solution architecture document for a real system from your organisation, with ranked drivers, quality attribute scenarios, service and deployment views and a set of decision records. You can present it to your review board and reuse its structure on the next system.

Solution Architecture Design: The Way Forward

The solution architecture document is the first artefact a review board and a delivery team will use, informing which hosting and integration option to fund. Engineers build from its views, operators plan from its resilience section and sponsors read its decision records. After its first release, revisit the quality attribute scenarios against measured behaviour and update the decisions whose consequences turned out differently.