SOFTWARE ARCHITECTURE

Modern development of products on scalable cloud platforms.

Aligned with ITABoK Version 3.0

Table of Contents

Course Sketch

Welcome to IASA's master class on software architecture. The software architecture role is complex, fast-moving and not something you can learn from a few podcasts or marketing blogs – being a good software architect is learned through experience. The SA has higher expectations from a broader group of stakeholders than the lead developer, must be a leader of the agile transformation, and have a broad view of patterns and best practice to coach the team to success… not to mention keeping up with the technology.

This course will take you from good architecture on familiar types of projects to being a truly great practitioner, able to take on new digital transformations and create products for cloud-scale customers. You will learn new patterns, tools, and terminology to grow your personal practice of leading the software enterprise. Learning from a core case study and successful examples from industry, and instructors who have been there – in the CTO or chief architect role for global software product developments – you will enter a broader world of software architects at a point where you can take your organization on this transformational journey.

Target Audience (who the course is for)

This course is specifically for individuals looking to grow into a senior software architect or product-oriented CTO position. Recommended for:

- Software developers and new software architects having completed large feature sets on 1-5 projects in at least one modern development environment and having participated in design, agile, and DevOps processes. This course does not teach how to code but requires general development tools familiarity.
- Also, non-coding software practitioners having performed solution or enterprise architecture and one technical specialty (infrastructure, software, information, security) across 1-5 projects and familiar with at least one modern development environment are invited.
- Technology leaders (chief technology officers, chief security officers, directors of operations/DevOps, program and project managers) are welcome to audit the course and are not required to take the final certification.

THERE IS NO PREREQUISITE TO HAVE TAKEN CORE OR SOLUTION ARCHITECTURE, NOR FAMILIARITY WITH THE NEW ITABOK 3.0

IF THE PARTICIPANT IS INTERESTED IN ACHIEVING THE CERTIFIED IT PROFESSIONAL IN SOFTWARE ARCHITECTURE, WE RECOMMEND TAKING THIS COURSE AT LEAST 3 MONTHS PRIOR TO THE BOARD DATE.

Course outcomes

You have shown the ability to take a set of requirements and deliver a solution to an operations team for your customer. This course focuses on the other aspects of a solution, and on maximizing your value and your company value in enabling your customers to reach the full potential for business value that your company provides. Most organizations involved in digital transformation are focused on efforts around three core processes: idea-to-market, market-to-order, and order-to-cash. They are concerned with changes needed to people, process, and technology. To provide the greatest value, you must be able to move beyond creating solutions to satisfy a number of requirements to advising the customer in technology use for the processes, and how people and process will be impacted and need to change as a result of the changes your solution proposes.

• understanding current agile/DevOps processes, principles and patterns broadened, increased ability to make decisions (tradeoffs) based on other projects and data, improve team efficiency
• be able to design multiple types of applications
• be able to develop and communicate principles to the team
• be able to identify where patterns would improve application design, where to find patterns, and how to evaluate a pattern for its usefulness in meeting project goals
• value-based backlog management
• grow knowledge of modern implementation techniques including cloud, microservices and containers
• work with other roles on the team effectively - especially agile and devops roles -- , improving SDLC efficiency (staying ahead of development and framing big goals), working with non-team roles to achieve QA goals
• develop tracks for development around security, quality attributes and present an architecture runway

Logistics

The course is delivered in an immersive, hands-on format with participant-driven labs followed by group analysis then followed by theory and best practice seminar. The labs require the participant to team with others in the course to accomplish particular software challenges, on their workstations, and experience patterns and tools that they may not have been familiar with previously. A case study is provided with both background details and completed solution, so no particular languages or tool familiarity is required. The instructor facilitates these experiences, coaching along the way, and then leads the group into discussion of learnings and the architecture trends exposed through the effort.

Additional self-paced learning materials and templates are provided to help you complete defined tasks. The extra time required to complete the self-paced activities varies with the initial hands-on-lab requiring approximately two hours down to “paper-based” analyses at 45 minutes work between sessions (or as reserved time in class).

This course is offered in two lengths and two modalities.

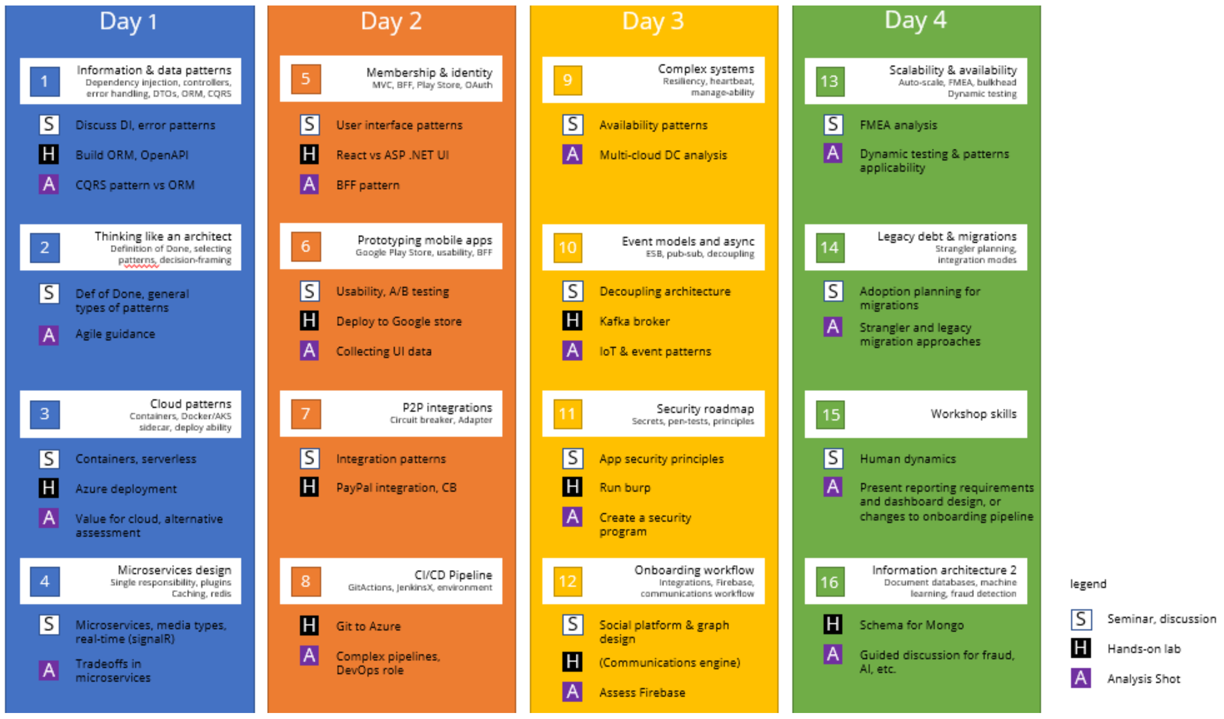
	ONLINE	IN PERSON
SHORT COURSE	• 24 hours of instructor-led training (typically 2x weekly for 2 hours each, for 6 weeks) • All analysis discussions with instructor and peers with supporting online digital classroom tools • 3 of 8 hands-on labs performed between sessions; remainder of HOLs	• 3 days of instructor-led training onsite or at training center with optional half day of certification prep • All analysis discussions with instructor and peers with supporting in-person digital classroom tools • 3 of 8 hands-on labs performed; remainder of HOLs provided for self-directed work after course

	provided for self-directed work after course • Full access to self-study materials, completed labs, additional readings	• Full access to self-study materials, completed labs, additional readings
FULL COURSE	• 40 hours of instructor-led training (typically 2x weekly for 2 hours each, for 10 weeks) • All analysis discussions with instructor and peers with supporting online digital classroom tools • All 8 hands-on labs performed between sessions • Full access to self-study materials, completed labs, additional readings	• 4.5 days of instructor-led training onsite or at training center with optional half day of certification prep • All analysis discussions with instructor and peers with supporting in-person digital classroom tools • All 8 hands-on labs performed • Full access to self-study materials, completed labs, additional readings

Executive Summary

This course will cover aspects of modern software architecture and includes discussions with the course instructor/practicing architect on any aspect of the topics below:

- Information Patterns
- Thinking like an Architect
- Cloud Patterns
- Microservices
- Membership and Identity
- Mobile Apps and Prototyping
- Integrations and Adapters
- DevOps and the CICD Pipeline
- Complex Systems
- Event-based Models
- Security Roadmap
- Workflow and Social Graphs
- Quality Attributes, Scalability
- Legacy Debt and Migrations
- Workshop Skills
- Information Architecture



Day 1

1 Information & data patterns
Dependency injection, controllers, error handling, DTOs, ORM, CQRS
- S Discuss DI, error patterns
- H Build ORM, OpenAPI
- A CQRS pattern vs ORM

2 Thinking like an architect
Definition of Done, selecting patterns, decision-framing
- S Def of Done, general types of patterns
- A Agile guidance

3 Cloud patterns
Containers, Docker/AKS sidecar, deploy ability
- S Containers, serverless
- H Azure deployment
- A Value for cloud, alternative assessment

4 Microservices design
Single responsibility, plugins Caching, redis
- S Microservices, media types, real-time (signalR)
- A Tradeoffs in microservices

Day 2

5 Membership & identity
MVC, BFF, Play Store, OAuth
- S User interface patterns
- H React vs ASP .NET UI
- A BFF pattern

6 Prototyping mobile apps
Google Play Store, usability, BFF
- S Usability, A/B testing
- H Deploy to Google store
- A Collecting UI data

7 P2P integrations
Circuit breaker, Adapter
- S Integration patterns
- H PayPal integration, CB

8 CI/CD Pipeline
GitActions, JenkinsX, environment
- H Git to Azure
- A Complex pipelines, DevOps role

Day 3

9 Complex systems
Resiliency, heartbeat, manage-ability
- S Availability patterns
- A Multi-cloud DC analysis

10 Event models and async
ESB, pub-sub, decoupling
- S Decoupling architecture
- H Kafka broker
- A IoT & event patterns

11 Security roadmap
Secrets, pen-tests, principles
- S App security principles
- H Run burp
- A Create a security program

12 Onboarding workflow
Integrations, Firebase, communications workflow
- S Social platform & graph design
- H (Communications engine)
- A Assess Firebase

Day 4

13 Scalability & availability
Auto-scale, FMEA, bulkhead Dynamic testing
- S FMEA analysis
- A Dynamic testing & patterns applicability

14 Legacy debt & migrations
Strangler planning, integration modes
- S Adoption planning for migrations
- A Strangler and legacy migration approaches

15 Workshop skills
- S Human dynamics
- A Present reporting requirements and dashboard design, or changes to onboarding pipeline

16 Information architecture 2
Document databases, machine learning, fraud detection
- H Schema for Mongo
- A Guided discussion for fraud, AI, etc.

legend

S	Seminar, discussion
H	Hands-on lab
A	Analysis Shot

Figure 1 Day estimates and topics differ per course length 3-5 days

Pictorial view of the modules in software architecture.

Syllabus by Module

Module 1 - Information and data patterns

Module 1

Lecture topics:

- **Review syllabus, draft of best practices, discussion of architecture value, the T-Coffee case study, and setting up your workstation.**
- **Introduction to state management in applications through information management and data storage; baseline agile terminology. Patterns include: ORM, CQRS.**

Hands-on-lab: ORM and OpenAPI

- **Hands-on designing and building the core of a loyalty subsystem for a retail organization (T-Coffee) to gain experience and review as a group, lessons learned related to object-relation manager components and API patterns.**

Facilitated peer discussions:

- **Architecture use of patterns, the IASA repository, comparison of ORM and CQRS patterns**

Module 2 - Thinking like an architect

Module 2

Lecture topics:

- **Describing the domain, service blueprint (for case study) and requirements analysis; architecture skills and key concerns. Thinking in layers, patterns, principles from schools of design, and the ITABOK. Software development lifecycle, evolutionary architecture techniques, and views.**

Facilitated peer discussions:

- **Discuss what goes into a definition of done (guidance to the team).**

Module 3 – Cloud patterns

Module 3

Lecture topics:

- **Understanding modern deployment platforms like containers and public cloud. Architecture technique of alternative assessments ad defining the architecture runway. Patterns include containers (infra), serverless, microservices.**

Hands-on labs:

- **Hands-on design and build the core application in a Docker container and migrating to Azure, discussion of different types of containers**

Facilitated peer discussions:

- **Technical, business and process efficiency tradeoffs of major public clouds vice on-premises datacenters; and the view from the CTO chair.**

Module 4 – Microservices design

Module 4
Lecture topics: • **Understanding design principles for microservices. Architectural value in the context of the project drivers, specific feature sets/value streams, benefits realization, and MVP prioritization. Demo of tools used in different lifecycle phases.**
Facilitated peer discussions: tradeoffs • **Compare and contrast our case study and other architectures both using and not using microservices**

Module 5 – Membership, identity and user interface

Module 5
Lecture topics: • **Presentation of user interface design patterns with specific discussion of how to guide the team for UX work. Considerations when writing a platform with membership/subscription and identity management options. Patterns include BFF, MVC/MVVC/MVP.**
Hands-on labs: user interface • **Hands-on design and implementation of a customer-facing interface in React or ASP .NET with authentication to the loyalty application.**
Facilitated peer discussions: decoupled UX • **General systems engineering concerns when the front-end doesn't match the back-end technology.**

Module 6 – Mobile apps and prototyping

Module 6
Lecture topics: • **Understanding user experience design and feedback-driven usability; the role of mockups and prototyping. Architecturally significant requirements, utility trees, and quality attributes in balance with functional requirements.**
Hands-on labs: Play Store • **Building a mobile application (comparing the UX and model) targeted for an application store online.**
Facilitated peer discussions:

- **How to collect actionable customer/user feedback including A/B testing in the mobile lifecycle.**

Module 7 – Integrations and adapter patterns

Module 7

Lecture topics:

- **Understanding the difference between integration development and functional application development starting with the business value proposition, API and contract development, prototyping and technical approaches. Differences between polling and event-based decoupling of state-based systems; reconciliation of data and sequence diagrams. Patterns include the adapter, circuit breaker, store-and-forward.**

Hands-on labs: user interface

- **Building an adapter to an external system like PayPal with resilience techniques like circuit breaker**

Facilitated peer discussions: decoupled UX

- **General systems engineering concerns when the front-end doesn't match the back-end technology.**

Module 8 – DevOps and CI/CD pipeline

Module 8

Lecture topics:

- **Understanding the role of a DevOps engineer and typical stages in a CI/CD pipeline. How the software architect can practically coordinate the agile cycle with the DevOps cycle and find efficiencies. Discussion of source code control and versioning in the context of multiple, simultaneous projects; the deployment and operations views. Common tools used in the DevOps process.**

Hands-on labs:

- **Develop a simple CI/CD pipeline from Git to a cloud deployment platform**

Module 9 – Complex systems

Module 9

Lecture topics:

- **Understanding the difference between complicated and complex systems (external variability) and techniques to build resilience in dynamic environments. Residual theory and techniques to bound non-linear and unexpected inputs (finding feedback points). Chaos and dynamic testing compared to traditional QA approaches. Discussion of CAP theorem and storage concerns in distributed systems.**

Facilitated peer discussions:

- **Analysis of multi-datacenter deployments**

Module 10 – Events and asynchronous processing

Module 10
Lecture topics: • **A deeper dive into the patterns of asynchronous message passing and events. Technologies include gRPC, webhooks, framework events, Kafka through to ESB/orchestration brokers. Patterns include pub-sub, queues, hub-and-spoke.**
Hands-on labs: events and message-passing • **Kafka or custom event system for communicating between distributed or scaled processes.**
Facilitated peer discussions: IoT patterns • **Compare techniques seen in real-time/stream/IoT solutions with general-purpose message passing considering the impact of reliable delivery/transactions and performance.**

Module 11 – Security roadmap

Module 11
Lecture topics: • **Following identity, adding discussion of authorization and roles, infrastructure security planning, regulatory constraints implemented in software. When to call in specialists, common tools used, and relationship with security team (is it the land-of-no)? Common enterprise and software architecture principles discussed. Patterns include saga, event sourcing/journaling.**
Hands-on labs: security tools • **Burp, Qualys, Snyk**
Facilitated peer discussions: • **Exercise to draft a security roadmap/track for the case study.**

Module 12 – Workflow and social graphs

Module 12
Lecture topics: • **Common techniques for implementing human-to-human workflow and graph data structures for social applications. Pros and cons of different workflow approaches from roll-your-own to service bus orchestration. How to integrate with social platforms.**
Hands-on labs: communications • **Integration of loyalty with a social platform (a la foodspotting)**
Facilitated peer discussions: Firebase

- **Understanding of a large-scale practical example of social application and how concerns like scalability, API decoupling and microservices are implemented. Analysis of how to migrate a relational schema to a graph schema.**

Module 13 – Scalability and availability

Module 13

Lecture topics:

- **Discussion of scalability and availability/reliability as two primary quality attributes of cloud-based and commercial applications: module-level patterns, testability, and FMEA. Common infrastructure solutions applicable to the software architecture. Patterns include retry.**

Facilitated peer discussions: dynamic testing

- **Exercise to develop a scalability approach for distributed workflow (eCommerce payments).**

Module 14 – Legacy debt and migrations

Module 14

Lecture topics:

- **Topics in identifying and planning for legacy debt remediation, developing more sustainable solutions, sunk cost, and user change management when migrating to a new solution. Platform thinking and building commercial products. Digital transformation in business model change and novel customer experience.**

Facilitated peer discussions: strangler pattern

- **Analysis of a practical legacy debt/coexistence/migration scenario in loan processing; building a roadmap for implementation**

Module 15 – Workshop skills

Module 15

Lecture topics:

- **Assessing personal human dynamics skills, cultural change for the software architect, how to introduce change through a planned engagement model.**

Hands-on labs: user interface

- **Hands-on design and implementation of a customer-facing interface in React or ASP .NET with authentication to the loyalty application.**

Facilitated peer discussions: participant presentation

- **Participants present a topic to the group to gain consensus on an architectural decision.**

Module 16 – Information architecture

Module 16
Lecture topics: • **Extending previous ORM and data analysis, presenting non-relational and long-term warehouse storage, the information view, frameworks for data governance and working with data stewards. Patterns include sharding/data splitting.**
Hands-on labs: • **Conversion of loyalty case study relational data storage to a document-based schema.**
Facilitated peer discussions: **Special topics in data science including applicability of statistical methods on application data (fraud detection) and customer engagement (AI/RPA/bots).**