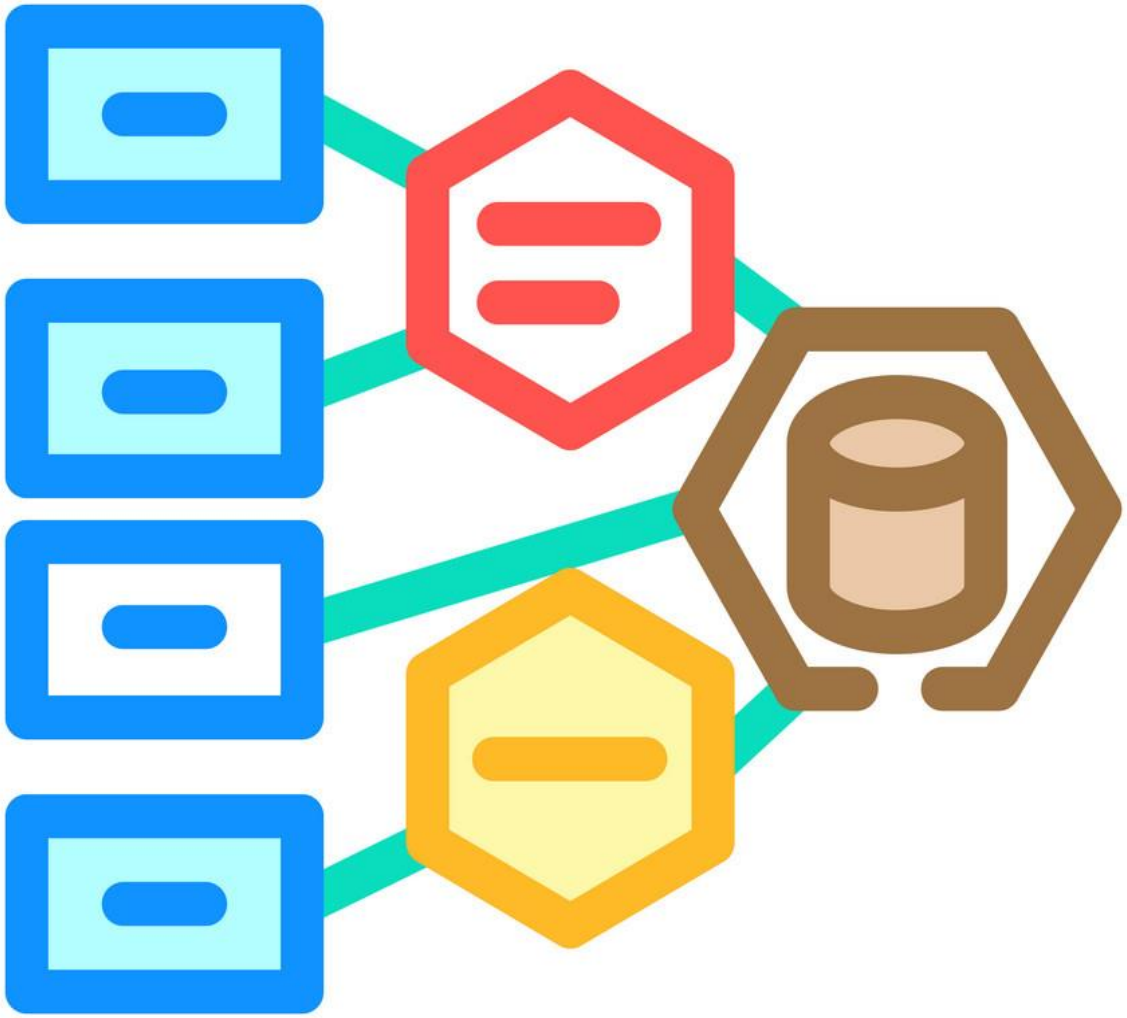
Software Design and architecture

Fall 2024

Dr. Natalia Chaudhry



Outline

- Architectural Evaluation

- An architectural evaluation is a structured process for assessing the quality, performance, and appropriateness of a system's architecture to ensure it meets functional and non-functional requirements.
- This evaluation typically involves examining factors like system scalability, maintainability, performance, and security, among other quality attributes.

Common Architectural Evaluation Methods

- **Architecture Tradeoff Analysis Method (ATAM):** Focuses on evaluating trade-offs between different architectural choices based on quality attributes. It's widely used to assess risks and benefits.
- **Cost-Benefit Analysis Method (CBAM):** Expands on ATAM by incorporating a cost-benefit perspective, evaluating the costs and expected benefits of architectural decisions.
- **Scenario-Based Evaluation:** Uses specific scenarios (use cases) to evaluate how well the architecture performs under expected conditions. Scenario-based evaluation methods such as **ATAM (Architecture Tradeoff Analysis Method)** and **CBAM (Cost-Benefit Analysis Method)** are particularly useful in this context.
- **SAAM (Software Architecture Analysis Method):** Focuses on modifiability, analyzing the ease with which the architecture can be changed to meet new requirements.
- **Performance Modeling and Simulation:** Uses simulations to predict how the architecture will perform under different conditions, identifying potential bottlenecks

Architecture Tradeoff Analysis Method (ATAM):

Phase 1: Preparation

- Identify the stakeholders (architects, developers, business leaders, end-users) who have an interest in the architecture.
- Define Quality Attributes
- Align architectural decisions with the system's business goals to ensure the architecture supports overall objectives.

Phase 2: Initial Evaluation

- Present the Architecture
- **Generate Utility Tree:** Stakeholders identify scenarios and organize them in a "utility tree" that prioritizes quality attributes by their importance and architectural impact.
- **Analyze Scenarios:** For each scenario, assess its impact on the architecture and determine potential risks, trade-offs, or sensitivities.

Phase 3: Detailed Analysis

- **Identify Sensitivities and Trade-offs:** Evaluate how the architecture's decisions impact various quality attributes, noting which decisions are sensitive to change or involve trade-offs.
- **Assess Risks and Non-Risks:** Identify and document both risks (areas where quality attributes may not be met) and non-risks (areas where the architecture has demonstrated robustness).
- **Rate Scenarios:** Assign priorities to scenarios to focus on the most impactful ones, emphasizing areas that need further refinement.

Phase 4: Final Report

- **Compile Findings:** Document all identified trade-offs, risks, sensitivities, and key architectural decisions.
- **Provide Recommendations:** Suggest risk mitigation strategies, alternative design options, or adjustments to the architecture to better balance trade-offs.
- **Review with Stakeholders:** Share findings with stakeholders, ensuring alignment on architectural decisions and strategies for improvement.

Using the **Architecture Tradeoff Analysis Method (ATAM)** to evaluate **Netflix's architecture** provides insight into how Netflix balances competing quality attributes like scalability, performance, availability, and modifiability in its complex system. Netflix, as a high-traffic streaming platform with a global user base, faces unique architectural challenges.

Define Quality Attributes (Goals for Netflix)

Performance
Scalability
Availability
Modifiability
Security

Identify Architectural Approaches

- Netflix uses a highly modular microservices architecture, where each microservice is responsible for a specific function (e.g., recommendations, user profile, streaming). This modularity allows Netflix to scale individual services independently and facilitates frequent updates.
- **Cloud-Based Infrastructure:** Netflix relies on AWS (Amazon Web Services) for cloud infrastructure, allowing it to dynamically scale its resources based on demand.

Generate Utility Tree and Scenarios

The utility tree helps Netflix prioritize key quality attributes and scenarios, such as:

- **High-performance streaming** during peak hours without degradation in quality.
- **Seamless global scalability** to support millions of concurrent users.
- **Automatic failover and recovery** to ensure high availability if a data center goes down.
- **Frequent feature deployment** with minimal service disruption.
- **Data security compliance** with global standards like GDPR for handling user information.

Analyze Scenarios for Trade-Offs

Example Scenario 1: High Availability and Scalability for Streaming Services

•**Trade-Offs:** To maximize availability and scalability, Netflix sacrifices some centralization by using distributed microservices across multiple AWS regions. This decentralization improves availability and resilience but increases the complexity of system monitoring and error handling.

Example Scenario 2: Low-Latency, High-Quality Streaming

•**Trade-Offs:** To provide high-quality, low-latency streaming, Netflix heavily invests in its Open Connect CDN, caching content closer to users. However, this approach incurs significant costs in CDN infrastructure and data replication, which could affect cost-efficiency compared to a centralized model.

Example Scenario 3: Modifiability for Rapid Feature Updates

- **Trade-Offs:** The microservices architecture allows rapid development and deployment of new features, as individual services can be updated independently. However, this level of modifiability may lead to data consistency challenges and debugging complexities due to multiple services interacting asynchronously.

Identify Sensitivity and Trade-Off Points

- **Sensitivity Point:** Scaling streaming services is sensitive to network bandwidth and latency, especially during peak times. Network congestion could impact quality, so Netflix carefully manages server locations and data flows in its CDN.
- **Trade-Off Point:** The balance between scalability and modifiability is achieved by using microservices. While this enhances modifiability, it introduces complexity in maintaining data consistency across services, especially with user preferences and recommendations.

Recommendations

- **Enhance Monitoring and Logging:** Invest in advanced monitoring and logging tools for real-time insights and anomaly detection, especially for latency-sensitive services.
- **Optimize Open Connect Nodes:** Continue optimizing CDN placement and load management to minimize latency, especially in regions with high user concentration.
- **Implement Fine-Grained Security Controls:** Adopt granular access and data protection measures to ensure compliance with regional data regulations and protect user privacy.

Cost-Benefit Analysis Method (CBAM)

Start with quality attributes and scenarios:

- **S1:** Scaling streaming services for peak viewership (e.g., new show releases).
- **S2:** Improving content delivery speed in new geographic markets.
- **S3:** Ensuring rapid deployment of new features without affecting service availability.
- **S4:** Enhancing data security to meet global regulatory requirements.

benefit scoring for Netflix:

- **S1 (Scalability for peak viewership):** Score: 9 (critical to maintaining user satisfaction during high-traffic events).
- **S2 (Content delivery speed in new regions):** Score: 8 (important for expansion and user acquisition in new markets).
- **S3 (Rapid feature deployment):** Score: 7 (essential for maintaining a competitive edge).
- **S4 (Enhanced data security):** Score: 9 (highly important to ensure compliance and avoid costly penalties).

Identify Architectural Strategies

Netflix would then propose various architectural strategies or improvements that could achieve these prioritized goals. Some strategies might include:

- **Strategy A: Expand Open Connect CDN:** Expand the Open Connect Content Delivery Network to optimize streaming quality and speed in underserved regions.
- **Strategy B: Introduce Advanced Auto-Scaling with Predictive Analytics:** Implement predictive scaling to manage peak demands without over-provisioning.
- **Strategy C: Microservices Enhancements:** Improve Netflix's microservices infrastructure for more efficient feature updates and lower inter-service latency.
- **Strategy D: Enhanced Data Encryption and Security Compliance Tools:** Upgrade security controls and tools for data encryption and regulatory compliance.

Estimate Costs and Benefits of Each Strategy

Strategy	Estimated Cost ($M)	Benefit Score (Weighted)	Benefit-Cost Ratio
Strategy A: Expand Open Connect CDN	$50M	8	0.16
Strategy B: Predictive Auto-Scaling	$20M	9	0.45
Strategy C: Microservices Enhancements	$30M	7	0.23
Strategy D: Enhanced Data Security Compliance	$40M	9	0.225

Select Optimal Strategies

Based on the cost-benefit analysis and risk assessment, Netflix can prioritize strategies with the highest benefit-cost ratio while factoring in risk. In this example, **Strategy B: Predictive Auto-Scaling** has the highest benefit-cost ratio, making it a top candidate for investment due to its lower cost and high impact on scalability and cost-efficiency.

Scenario-Based Evaluation

These scenarios represent common and critical operations for Netflix:

- **High Demand for Streaming:** A scenario where a popular show is released, causing a spike in users. Netflix's architecture must handle increased load, maintain performance, and ensure no service disruption.
- **New Feature Deployment:** Releasing new features (like recommendations or playback speed options) without disrupting the service.
- **Regional Expansion:** Expanding streaming availability in new regions, which involves handling language localization, new regulatory requirements, and optimizing CDN performance.
- **Data Privacy and Security Compliance:** Addressing scenarios where Netflix must comply with global privacy laws (e.g., GDPR) while maintaining data security and access control.
- **Disaster Recovery:** A scenario where a data center fails. Netflix must reroute services to other centers seamlessly.

SAAM (Software Architecture Analysis Method):

Applying SAAM to **Netflix's architecture** offers insight into how well its system supports change, new feature additions, and specific functional needs—especially given Netflix’s reliance on microservices and cloud-based infrastructure.

assess how well the architecture supports necessary changes and functions

•**Scenario 1: Scaling for High Demand**
•**Modifiability:** Netflix's microservices architecture and AWS cloud infrastructure support rapid scaling by allowing independent service replication and distribution. Modifications are limited to scaling individual microservices without affecting the entire system.
•**Functionality:** AWS's autoscaling and Netflix's Open Connect CDN enhance functionality by delivering high-quality, low-latency streaming to users during peak loads.

•**Scenario 2: Deploying New Features**
•**Modifiability:** Microservices allow Netflix to deploy new features independently. For example, a new recommendation algorithm can be deployed to a specific microservice without needing a full system update. This architecture is highly modifiable.
•**Functionality:** Modifying individual services without affecting the core system ensures continuity in user experience, meeting functionality goals.

- **Scenario 3: Geographic Expansion**
- **Modifiability:** The architecture's modular structure supports easy integration of new language modules, localization features, and regional legal requirements without major system-wide changes.
- **Functionality:** Open Connect's distributed CDN structure provides high-quality streaming regardless of user location, enhancing global functionality.

- **Scenario 4: Enhancing Data Privacy and Security**
- **Modifiability:** Security upgrades, such as improved encryption or access control, can be applied to relevant microservices independently.
- **Functionality:** Enhanced security measures meet both user expectations and regulatory requirements, ensuring Netflix can operate in regions with strict data protection laws.

- **Scenario 5: Disaster Recovery and Failover**
- **Modifiability:** AWS’s infrastructure enables Netflix to adapt to service interruptions by dynamically rerouting traffic, even if a data center goes down.
- **Functionality:** This supports Netflix’s high availability requirement, ensuring that users experience minimal to no service interruption during failures.

Performance Modeling and Simulation

Simulation:
Models can simulate different types of **user behaviors**, such as browsing, streaming, or interaction with content recommendations.
Simulating peak usage times (like weekends or during new releases) can help predict server load and plan for scaling.

Modeling how content is cached and served from different edge servers across the globe. This can help identify potential bottlenecks in data transmission.
Simulating different network conditions (e.g., congested or slow networks) can help optimize content delivery strategies

Simulating cloud resource utilization (e.g., computing power, memory, and storage) during varying load scenarios, such as high demand periods or sudden traffic spikes.
Predicting the scaling requirements of microservices, databases, and video transcoding during a new show release.

That’s it