K J Somaiya College of Engineering
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A1	**Roll No.: 16010121051**
Experiment / assignment / tutorial No 5	

TITLE : ATAM - Architecture Trade-off Analysis Method for B E Project.

AIM : To apply Human-centric analysis process ATAM for identifying risks early on in software design for the B E Project.

Expected OUTCOME of Experiment:

Books/ Journals/ Websites referred:

1 “Software Architecture, Richard N Taylor etl, Wiley

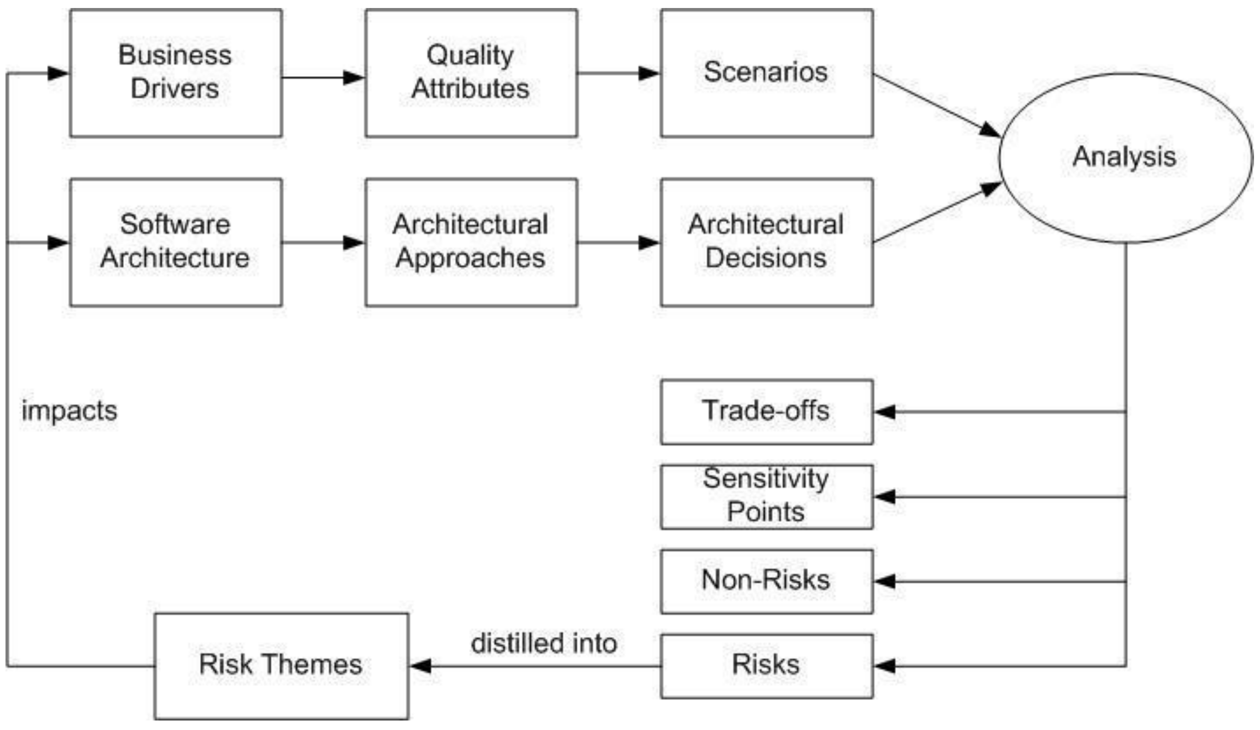
Theory:

ATAM stands for architectural trade-off analysis method. This method Focuses specifically on four quality attributes (NFPs)

1. Modifiability
2. Security
3. Performance
4. Reliability

ATAM reveals how well an architecture satisfies quality goals and how those goals trade-off.

ATAM Process:



The following scenarios can be used:

- Use-case scenarios
    - Describe how the system is envisioned by the stakeholders to be used
- Growth scenarios
    - Describe planned and envisioned modifications to the architecture
- Exploratory scenarios
    - Try to establish the limits of architecture's adaptability with respect to system's functionality, operational profiles ,underlying execution platforms

    - Scenarios are prioritized based on importance to Stakeholders

ATAM in a nutshell:

Goals	**Completeness**	
	Consistency	
	Compatibility	
	Correctness`	
Scope	**Subsystem- and system-level**	
	Data exchange	
Concern	**Non-functional**	
Models	**Informal**	
	Semi-formal	
Type	**Scenario-driven**	
Automation Level	**Manual**	
Stakeholders	**Architects**	
	Developers	
	Managers	
	Customers	

<u>ATAM Analysis of B E Project</u>:

Title: Fund Trail Analysis Tool

User Interfaces

 Web-Based Dashboard:
- A responsive web interface that allows users to interact with the tool, visualize data, and analyze transaction patterns.
- Users can upload financial documents, review analysis results, and generate reports.

 Data Visualization Module:
- Provides visual representations of the fund flow, balance sheet analysis, and money trail, including graphs, charts, and tables.
- Supports zoom, filter, and drill-down capabilities for in-depth analysis.

 Interactive Query Interface:
- A form-based or search-based interface where users can input queries for custom analysis of transactions and fund flows.

Hardware Interfaces

- **Server Infrastructure:**
  - High-performance servers are required for data processing, including OCR, machine learning models, and real-time analytics.
  - May utilize cloud-based infrastructure (AWS, Azure, or Google Cloud) for scalability.
- **Storage Systems:**
  - Interfaces with databases (e.g., SQL, NoSQL) for storing financial data, transaction records, and user-generated reports.
- **Network Equipment:**
  - Network hardware (routers, switches) to ensure secure and fast communication between servers and clients.

Software Interfaces

- **Operating System Compatibility:**
  - Supports major operating systems, including Windows, macOS, and Linux, for server-side and user-side applications.
- **Database Management Systems:**
  - Interfaces with databases like PostgreSQL, Neo4j (graph database for fund tracing), and MongoDB for various data needs.
- **APIs and SDKs:**
  - RESTful APIs and SDKs for third-party integrations, enabling easy interfacing with external systems such as bank APIs, AML (Anti-Money Laundering) systems, and accounting software.

Communication Interfaces

- **Secure Data Transmission:**
  - Uses HTTPS/SSL/TLS protocols for secure data exchange between users and servers to protect sensitive financial information.
- **Inter-Component Communication:**
  - Microservices communicate using REST or gRPC protocols, allowing for scalable and maintainable architecture.
- **Data Ingestion Channels:**
  - Supports data ingestion through multiple channels, such as file uploads (CSV, PDF), APIs, and direct database connections.

Quality Attributes

The quality attributes that comprise the current system are as follows:

1. **Security:**
   - Ensures the confidentiality, integrity, and availability of financial data through encryption, authentication, and authorization mechanisms.
2. **Performance:**
   - Optimizes for low-latency data processing and real-time transaction analysis, especially during peak loads.
3. **Scalability:**
   - Designed to handle increasing data volumes and concurrent users by leveraging cloud-native architectures.
4. **Maintainability:**
   - Code is modular and well-documented, facilitating easy updates and maintenance.
5. **Usability:**
   - User interfaces are intuitive, with minimal training required for effective usage by non-technical users.
6. **Reliability:**
   - The system is robust against failures, with proper error handling, failover mechanisms, and data backups in place.

Architectural Approaches

- **Microservices Architecture:**
  - Breaks down the system into loosely coupled services for scalability and maintainability.
- **Event-Driven Architecture:**
  - Utilizes message queues (e.g., Kafka, RabbitMQ) for real-time data processing and notifications.
- **Graph Database Integration:**
  - Incorporates Neo4j for efficient storage and querying of complex relationships in money trails.
- **Machine Learning Models:**
  - Deploys custom models for anomaly detection and predictive analysis of fraudulent transactions.

Scenarios

1. **Data Breach Attempt:**
   - System responds with automated alerts, data encryption, and user session termination.
2. **High-Volume Data Ingestion:**
   - System scales horizontally by deploying more instances and balancing the load effectively.
3. **Real-Time Fraud Detection:**
   - System processes transactions in real-time and flags suspicious patterns based on predefined rules.

4. **Integration with External Systems:**
   - Seamless integration with AML systems via REST APIs for enhanced fraud detection capabilities.

Analysis

 Performance vs. Security:
- Achieving a balance between processing speed and encryption overhead for sensitive data.

 Scalability vs. Maintainability:
- Managing the complexity of microservices while ensuring the ease of updates and debugging.

 Usability vs. Functionality:
- Ensuring the user interface remains simple without compromising the tool's analytical capabilities.

Trade-Off

 Security vs. Performance:
- High security may affect the system's performance due to encryption and decryption overheads.

 Scalability vs. Consistency:
- Scalable solutions, such as eventual consistency models in distributed databases, may compromise data consistency.

 Cost vs. Availability:
- Utilizing cloud-based infrastructure for high availability can lead to increased operational costs.

Sensitivity Points

- **Database Choice:**
  - Choosing between SQL, NoSQL, and graph databases impacts performance, scalability, and complexity.
- **API Design:**
  - The structure and endpoints of APIs significantly affect data transfer efficiency and security.
- **Model Accuracy:**
  - The accuracy of machine learning models influences the effectiveness of fraud detection.

Risks

 Data Privacy Breach:
- Sensitive financial data might be exposed if security measures fail.

 System Downtime:
- High availability might be compromised due to network failures or server crashes.

 False Positives in Fraud Detection:
- Incorrectly identifying legitimate transactions as fraudulent can cause trust issues with users.

Non-Risks

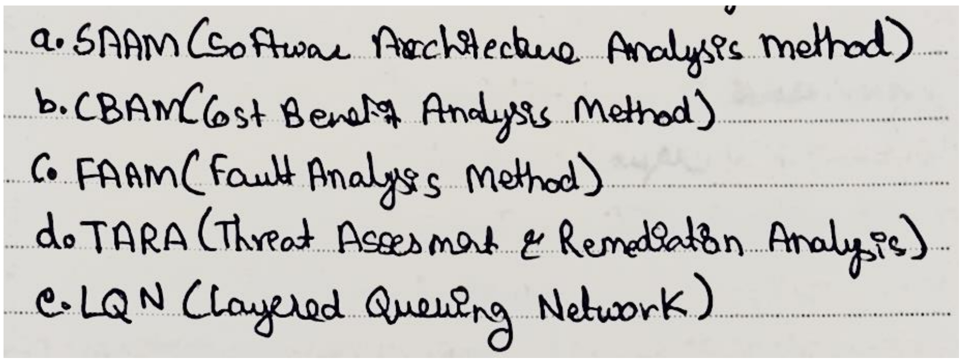
 Compliance with Regulatory Standards:
- The system adheres to financial regulations and data privacy laws, reducing the risk of legal repercussions.

 Modular Architecture:
- The microservices-based architecture reduces the risk of a single point of failure.

 Scalable Cloud Infrastructure:
- Leveraging cloud platforms ensures that scaling the application to accommodate more users is not a risk.

<u>Post Lab Descriptive Questions</u>

1 . What are the other methods of NFP analysis



a. SAAM (Software Architecture Analysis method)
b. CBAM (Cost Benefit Analysis Method)
c. FAAM (Fault Analysis Method)
d. TARA (Threat Assessment & Remediation Analysis)
e. LQN (Layered Queuing Network)