**<PRODUCT>**

**Architecture Description**

Date: <date>  
Date: <date>

Version

Author: <author name>, Architect

|  |  |  |
| --- | --- | --- |
| **List of reviewers:** | <add role if required> | <Name> |
|  | <add role if required> | <Name> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **Distribution List:** | Project Team | |

Contents

[1. Introduction 3](#_Toc311704874)

[1.1. Purpose of the Document 3](#_Toc311704875)

[1.2. Audience 3](#_Toc311704876)

[1.3. Scope 3](#_Toc311704877)

[1.4. Related Documents 3](#_Toc311704878)

[1.5. Revision History 3](#_Toc311704879)

[2. Glossary 4](#_Toc311704880)

[3. Problem Background 5](#_Toc311704881)

[3.1. System Overview 5](#_Toc311704882)

[3.2. Goals and Context 5](#_Toc311704883)

[3.3. Stakeholders 5](#_Toc311704884)

[3.4. Significant Driving Requirements 5](#_Toc311704885)

[4. Solution Background 6](#_Toc311704886)

[4.1. Solution Overview 6](#_Toc311704887)

[4.2. Assumptions 6](#_Toc311704888)

[4.3. Approach Summary 6](#_Toc311704889)

[4.4. Analysis Results 6](#_Toc311704890)

[4.5. Tradeoffs 6](#_Toc311704891)

[4.6. Made Tradeoffs and Known Non-Risks 6](#_Toc311704892)

[4.7. Recognized Risks and Sensitivity Points 6](#_Toc311704893)

[4.8. Design Decisions Backlog 7](#_Toc311704894)

[5. Quality Requirements and Architectural Approaches 8](#_Toc311704895)

[5.1. Availability and Resilience Perspective 8](#_Toc311704896)

[5.2. Performance and Scalability Perspective 8](#_Toc311704897)

[5.3. Evolution and Maintainability Perspective 8](#_Toc311704898)

[5.4. Security Perspective 8](#_Toc311704899)

[5.4.1. Threat Model 8](#_Toc311704900)

[5.5. Regulation Perspective 9](#_Toc311704901)

[5.6. Usability Perspective 9](#_Toc311704902)

[5.7. Internationalization Perspective 10](#_Toc311704903)

[5.8. Development Resource Perspective 10](#_Toc311704904)

[5.8.1. Development Process Approach 10](#_Toc311704905)

[5.8.2. Organization 10](#_Toc311704906)

[5.8.3. Skills 10](#_Toc311704907)

[6. Logical View 11](#_Toc311704908)

[7. Information View 12](#_Toc311704909)

[8. Concurrency View 13](#_Toc311704910)

[9. Development View 14](#_Toc311704911)

[9.1.1. Implementation 14](#_Toc311704912)

[9.1.2. Standards 14](#_Toc311704913)

[9.1.3. Build 14](#_Toc311704914)

[9.1.4. Testing 14](#_Toc311704915)

[10. Deployment View 15](#_Toc311704916)

[10.1.1. Installation 15](#_Toc311704917)

[10.1.2. Configuration 15](#_Toc311704918)

[11. Operational View 16](#_Toc311704919)

[11.1.1. Monitoring and Troubleshooting 16](#_Toc311704920)

[11.1.2. Upgrade 16](#_Toc311704921)

# Introduction

## Purpose of the Document

## Audience

## Scope

## Related Documents

List here any related documents

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Number** | **Modified sections** | **Description of changes** |
| **Specify revision date** | Specify new document revision number | Specify modified sections | Describe what changes have been made and why |

# Glossary

|  |  |
| --- | --- |
| **Term/Acronym** | **Meaning** |
|  |  |

# Problem Background

The sub-parts explain the constraints that provided the significant influence over the architecture

## System Overview

This section describes the general function and purpose for the system or subsystem whose architecture is described in this AD

## Goals and Context

This section describes the goals and major contextual factors for the software architecture. The section includes a description of the role software architecture plays in the life cycle, the relationship to system engineering results and artifacts, and any other relevant factors.

## Stakeholders

## Significant Driving Requirements

This section describes behavioral and quality attribute requirements (original or derived) that shaped the software architecture. Included are any scenarios that express driving behavioral and quality attribute goals

# Solution Background

The sub-parts of this section provide a description of why the architecture is the way that it is, and a convincing argument that the architecture is the right one to satisfy the behavioral and quality attribute goals levied upon it

## Solution Overview

One or two diagrams, which could be pasted in nearly any presentation about the system and short description of key solution features

## Assumptions

List any assumptions made while making design of the architecture

## Approach Summary

This section provides a rationale for the major design decisions embodied by the software architecture. It describes any design approaches applied to the software architecture, including the use of architectural styles or design patterns, when the scope of those approaches transcends any single architectural view. The section also provides a rationale for the selection of those approaches. It also describes any significant alternatives that were seriously considered and why they were ultimately rejected. The section describes any relevant COTS issues, including any associated trade studies

## Analysis Results

This section describes the results of any quantitative or qualitative analyses that have been performed that provide evidence that the software architecture is fit for purpose. If an architecture analysis evaluation has been performed, it is included in the analysis sections of its final report. This section refers to the results of any other relevant trade studies, quantitative modeling, or other analysis results

## Tradeoffs

This section highlights all trade-offs between different requirements that have been made during architecture design. E.g. “we had to relax max response time requirement from 1 sec to 5 sec, because that would be impossible to reach over HTTPS protocol, which we introduced into current architecture instead of HTTP, because of the security requirement on the data confidentiality, which has more priority, rather than performance.”

## Made Tradeoffs and Known Non-Risks

## Recognized Risks and Sensitivity Points

This section describes all existing risks to satisfy functional and non-functional requirements within current architecture design.

|  |  |  |
| --- | --- | --- |
| **Risk** | **Probability/Impact** | **Mitigation Approach** |
| In evolution perspective - Market Values can grow a much in memory consumption.  Design for financial availability has not been properly elaborated | High/High | Analyze data sizing and elaborate the design |
| By distributing processing (Inventory Enquirer and Flight Connection Enquirer) we might run into troubles, if eventually will be requested to do some logic, which spans current responsibility borders between these two components | Low/Med | To be validated within IAS Domain Team (AC). |

## Design Decisions Backlog

# Quality Requirements and Architectural Approaches

## Availability and Resilience Perspective

Desired quality: the ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability

Concerns: сlasses of service, planned downtime, unplanned downtime, time to repair, and disaster recovery

## Performance and Scalability Perspective

This perspective helps you to address the two related quality properties of performance and scalability. These properties are important because, in large systems, they can cause more unexpected, complex, and expensive problems late in the system lifecycle than most of the other properties combined.

Concerns: Response time, throughput, scalability, predictability, hardware resource requirements, and peak load behavior

## Evolution and Maintainability Perspective

Describes the concerns related to dealing with evolution during the lifetime of a system and thus is relevant to most large-scale information systems because of the amount of change that most systems need to handle.

Concerns: Magnitude of change, dimensions of change, likelihood of change, timescale for change, when to pay for change, development complexity, preservation of knowledge, and reliability of change

## Security Perspective

The security perspective guides you as you consider the set of processes and technologies that allow the owners of resources in the system to reliably control who can perform what actions on particular resources.

Concerns: Policies, threats, mechanisms, accountability, availability, and detection and recovery.

### Threat Model

As you identify a threat give it a DREAD rating. After you ask the above questions, count the values (1–3) for a given threat. The result can fall in the range of 5–15. Then you can treat threats with overall ratings of 12–15 as High risk, 8–11 as Medium risk, and 5–7 as Low risk.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Rating** | **High (3)** | **Medium (2)** | **Low (1)** |
| **D** | Damage potential | The attacker can subvert the security system; get full trust authorization; run as administrator; upload content. | Leaking sensitive information | Leaking trivial information |
| **R** | Reproducibility | The attack can be reproduced every time and does not require a timing window. | The attack can be reproduced, but only with a timing window and a particular race situation. | The attack is very difficult to reproduce, even with knowledge of the security hole. |
| **E** | Exploitability | A novice programmer could make the attack in a short time. | A skilled programmer could make the attack, then repeat the steps. | The attack requires an extremely skilled person and in-depth knowledge every time to exploit. |
| **A** | Affected users | All users, default configuration, key customers | Some users, non-default configuration | Very small percentage of users, obscure feature; affects anonymous users |
| **D** | Discoverability | Published information explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable. | The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious use. | The bug is obscure, and it is unlikely that users will work out damage potential |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Threat** | **D** | **R** | **E** | **A** | **D** | **Total** | **Rating** |
|  |  |  |  |  |  |  |  |
| Mitigation | | | | | | | |

## Regulation Perspective

Unlike other system qualities, compliance with the law is an area where you cannot make compromises. Although you may be able to live with a system that is slow, occasionally unreliable, or potentially insecure, a system that does not comply with legal regulations may be prevented from going into production or may expose the organization to risk of prosecution.

Concerns: statutory industry regulation, privacy and data protection, cross-border legal restrictions, data retention and accountability, and organizational policy compliance

## Usability Perspective

Applying the Usability perspective ensures that the system allows those who interact with it to do so effectively. This perspective tends to focus on the end users of the system but should also address the concerns of any others who interact with it directly or indirectly, such as maintainers and support personnel.

Concerns: User interface usability, business process flow, information quality, alignment of the human–computer interface (HCI) with working practices, alignment of the HCI with users’ skills, maximization of the perceived usability, and ease of changing user interfaces

## Internationalization Perspective

The Internationalization perspective is important for any system that will have users who speak different languages or come from different countries. If systems are aimed at a specific locale with no plans to move it into a wider area, this perspective has limited relevance.

Concerns: Character sets, text presentation and orientation, specific language needs, cultural norms, automatic translation, and cultural neutrality.

## Development Resource Perspective

All software projects are primarily constrained by time and cost. IT budgets are never unlimited, and although technology capabilities improve from year to year, so do the costs of building, deployment, and support. This perspective allows you to consider whether your architecture can be created, given development resource constraints.

Concerns: time constraints, cost constraints, required skill sets, available resources, budgets, and external dependencies

### Development Process Approach

Describe the development approach: incremental and iterative development, agile practices, expectation management, descoping, prototyping and piloting, teams interaction, communication with customer’s stakeholders, etc.

### Organization

Present here rough organizational structure. In some cases, it directly depends on the functional decomposition of the system (a subsystem may be given to one team, another large subsystem – to another team)

### Skills

What skills the project team should fit?

# Logical View

Describes the system’s runtime functional elements and their responsibilities, interfaces, and primary interactions

Concerns: functional capabilities, external interfaces, internal structure, and design philosophy

[INSERT HERE LOGICAL COMPONENT DIAGRAM]

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Description** | **Provided Interfaces** | **Required Interfaces** |
|  |  |  |  |

# Information View

Describes the way that the architecture stores, manipulates, manages, and distributes information

Concerns: information structure and content; information flow; data ownership; timeliness, latency, and age; references and mappings; transaction management and recovery; data quality; data volumes; archives and data retention; and regulation

[INSERT HERE DATA MODEL DIAGRAM]

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Description** | **Life-cycle** | **Ownership/Access** |
|  |  |  |  |

# Concurrency View

Describes the concurrency structure of the system, mapping functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently, and shows how this is coordinated and controlled

Concerns: task structure, mapping of functional elements to tasks, inter-process communication, state management, synchronization and integrity, startup and shutdown, task failure, and reentrancy

# Development View

Describes the architecture that supports the software development process

Concerns: module organization, common processing, standardization of design, standardization of testing, instrumentation, and codeline organization

### Implementation

For each software component from the logical view define which technologies it should be built upon, which design patterns should be applied to it. And finally, in which form it will be packaged for deployment into the target and test environments.

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Implementation Technologies** | **Design Approaches** | **Target build artifacts** |
| Component from the logical view | e.g. Java 6, JSF, Hibernate 3.2 | Design patterns: GoF, P of EAA, P of EAI, P of SOA  e.g. multi-tier JEE application, web MVC, DDD | Applications archives (WAR, JAR), SQL sripts, etc |

### Standards

Describe here which industrial or project-specific standards, company’s blueprints that are mandated to comply with.

|  |  |
| --- | --- |
| **Scope** | **Standard** |
| Any Java code | Sun’s Java coding conventions, project-specific standards |
| Web-services: WSDL and XSD artifacts | WS-Interoperability Basic Profile 1.1, Exigen blue-print standards on naming and versioning |
| Data model, subsystem interfaces | TMF SID, OSSJ |

### Build

Build process and build artifacts. Codeline structure.

Defined regularly build schedule and actions included into these builds (e.g. unit-testing or interface contract validation against some policy). Base-lines, versioning, branching.

### Testing

This is the right place to call out special testing concerns or to detail the ways in which testers might most effectively exercise the functionality of the component. Does system provide instrumentation to let tester view what’s happening inside the system, not only the user UI (special logging, monitors, etc), especially for testing exceptional cases, e.g. connection failure.

# Deployment View

Describes the environment into which the system will be deployed, including the dependencies the system has on its runtime environment

Concerns: types of hardware required, specification and quantity of hardware required, thirdparty software requirements, technology compatibility, network requirements, network capacity required, and physical constraints

[INSERT HERE Deployment diagram: nodes, networks]

|  |  |  |
| --- | --- | --- |
| **Node** | **HW/SW Stack** | **Deployed Units** |
|  | Hardware platform, OS, VM, AS, ESB, MQ,… – with specific version and service pack details | Deployed units, that are normally the build artifacts described in the Development view |

### Installation

How will the product be deployed? Are there any special deployment considerations?

### Configuration

How will this component be configured? Is it a master configuration or master configuration service? Is it component specific configuration file?

Describe data stored in each configuration section

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Default** |
|  |  |  |

# Operational View

Describes how the system will be operated, administered, and supported when it is running in its production environment

Concerns: installation and upgrade, functional migration, data migration, operational monitoring and control, configuration management, performance monitoring, support, and backup and restore

### Monitoring and Troubleshooting

What are the monitoring and maintenance requirements? How does this design address the monitoring and maintenance needs?

### Upgrade

What are the upgrade requirements? How does this design address the upgrade needs?