17-655 Architectures for Software Systems

Software Architecture Document

A1: Pipe-and Filter

David Qiu, Roy Hsiao, Sashank Pandem , Yahui Chu

February 15, 2016

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| 02/06/2016 | 1.0 | Initial version of SAD for comments by team | Roy Hsiao |
| 02/09/2016 | 1.1 | Changed based on Team Discussion |  |

Contents

[1. Introduction 4](#_Toc442521993)

[1.1. Purpose 4](#_Toc442521994)

[1.2. Scope 4](#_Toc442521995)

[1.3. Definitions, Acronyms, and Abbreviations 4](#_Toc442521996)

[1.4. References 4](#_Toc442521997)

[1.5. Overview 4](#_Toc442521998)

[2. Architectural Representation 4](#_Toc442521999)

[3. Architectural Goals and Constraints 5](#_Toc442522000)

[3.1. Security 5](#_Toc442522001)

[3.2. Persistence 5](#_Toc442522002)

[3.3. Reliability/Availability 6](#_Toc442522003)

[3.4. Performance 6](#_Toc442522004)

[4. Use-Case View 6](#_Toc442522005)

[4.1. Actors 6](#_Toc442522006)

[4.2. Use-Case Realizations 6](#_Toc442522007)

[5. Logical View 6](#_Toc442522008)

[5.1. Overview 6](#_Toc442522009)

[6. Process View 6](#_Toc442522010)

[7. Module Decomposition View 6](#_Toc442522011)

[8. Data View 6](#_Toc442522012)

[9. Deployment View 6](#_Toc442522013)

[10. Size and Performance 7](#_Toc442522014)

[11. Issues and concerns 7](#_Toc442522015)

[12. References 8](#_Toc442522016)

Software Architecture Document

# Introduction

This document provides a high level architecture overview and explains the process of Assignment one Pipe-and-Filter

## Purpose

## Scope

## Definitions, Acronyms, and Abbreviations

## References

## Overview

# Architectural Representation

**Use Case view**

**Audience**: all the stakeholders of the system, including the end-users.

**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

**Related Artifacts** : Use-Case Model, Use-Case documents

**Logical view**

**Audience**: Designers.

**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.

**Related Artifacts**: Design model

**Process view**

**Audience**: Integrators.

**Area**: Non-functional requirements: describes the design's concurrency and synchronization aspects.

**Related Artifacts**: (no specific artifact).

**Module Decomposition view**

**Audience**: Programmers.

**Area**: Software components: describes the modules and subsystems of the application.

**Related Artifacts**: Implementation model, components

**Data view**

**Audience**: Data specialists, Database administrators

**Area**: Persistence: describes the architecturally significant persistent elements in the data model

**Related Artifacts**: Data model.

**Deployment view**

**Audience**: Deployment managers.

**Area**: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.

**Related Artifacts**: Deployment model.

# Architectural Goals and Constraints

## Security

## Persistence

## Reliability/Availability

## Performance

# Use-Case View

## Actors

## Use-Case Realizations

# Logical View

## Overview

# Process View

# Module Decomposition View

# Data View

# Deployment View

# Size and Performance

# Issues and concerns

# References

Bass, L., Clements, P., & Kazman, R. (2013). *Software architecture in practice.* Upper Saddle River, NJ: Addison-Wesley.

Clements, P., Bachmann, F., & Bass, L. (2003). *Documenting software architectures: views and beyond.* Boston: Addison-Wesley.

Eeles, P. (2006, May 15). *The benefits of software architecting*. Retrieved from IBM The Rational Edge: http://www.ibm.com/developerworks/rational/library/may06/eeles/

Garlan, D., & Shaw, M. (1994). *An Introduction to Software Architecture.* New Jersey: World Scientific Publishing Company.

Intel. (2001). *N-tier Architecture Improves Scalability, Availability and Ease of Integration.* Santa Clara: Intel.

Lattanze, A. (2008). *Architecting Software Intensive Systems.* New York, NY: Auerbach.