Final Report

DXC Technology Managed Services For Workloads On Public Cloud

SE 4485.001
Dr. Weichen Wong

Group 2:
Ihfaz Tajwar
Zunayed Siddiqui
Po-Yu Liu
Yeswanth Bogireddy
Zachary Tarell

Executive Summary

The following is our teams Senior Design Project to aid DXC Technology with CloudWatch monitoring and ServiceNow integration by creating a monitoring service that will provide for their company. Service providers are constantly challenged to accept customers' workloads as-is and manage it on behalf of the customer. The management can range from monitoring, patching, and security aspects of the workload. The security could shut off certain ports or setup security scans for Denial of Service attack, or over-utilization, among others. The monitoring could leverage the cloud provider monitoring but would alert based on certain rules. The patching could use cloud provider specific to patch from vulnerabilities, routine service packs, customer service, etc.

This document will provide a detailed rundown of all necessary actions, models, and resources needed to setup a basic monitoring tool for DXC Technology. Starting with management planning that includes risk analysis, requirements and deliverables, software needed, and scheduling of events to make it better for the company to implement a working system.

The architectural style and design models consist of the method at which our project can be deployed as well as class and sequence diagrams to elaborate on the work breakdown structure of its operation, utilization, and total consumption to run the application successfully. Rounding out with testing and traceability matrices connecting use cases to requirements and deliverables, our cloud monitoring provisioning allows for a solid and efficient system.

In conclusion, whether the company decides on AWS, GCP, Azure, or any other cloud provider, this project will meet those needs with the proper foundation of adding future projects with more extensive provisioning and will surely save money and time on much needed resources to allocate elsewhere.

Contents

List of Figures 4
List of Tables 4
1. Introduction 5
1.1 Purpose and Scope 5
1.2 Product Overview 5
1.3 Structure of the Document 6
1.4 Terms, Acronyms, and Abbreviations 6
2. Project Management Plan 7
2.1 Project Organization 7
2.2 Life Cycle Model Used 7
2.3 Risk Analysis 8
2.3.1 - Risk Management Plan 8
2.4 Hardware and Software Resource Requirements 8
2.4.1 - Hardware 8
2.4.2 - Software 8
2.5 Deliverables and Schedule 9
2.6 Monitoring, Reporting and Controlling Mechanisms 9
2.6.1 - Work Breakdown Structure (WBS) 9
2.6.2 - Requirements Traceability Matrix (RTM) 9
2.6.3 - Control Chart 9
2.6.4 - Review and Status Meetings 9
2.7 Professional Standards 9
2.8 Evidence the Document has been placed under Configuration Management 10
2.9 Impact of the Project on Individuals and Organization 10
3. Requirement Specifications 11
3.1 Stakeholders for the System 11
3.2 Use Case Model 11
3.2.1 – Graphic Use Case Model 11
3.2.2 – Textual Description for Each Use Case 12
3.3 Rationale for Use Case Model 12
3.4 Non-Functional Requirements 12
4. Architecture 13
4.1 Architectural Style(s) Used 13
4.2 Architectural Model 13
4.3 Technology, Software and Hardware Used 13
4.4 Rationale for Architectural Style and Model 13

5. Design 14
5.1 GUI (Graphical User Interface) Design 14
5.2 Static Model – Class Diagram 14
5.3 Dynamic Model – Sequence Diagrams 15
5.4 Rationale for Detailed Design Model 16
5.5 Traceability Matrix from Requirements to Detailed Design Model 16
6. Test Plan 17
6.1 Requirements/Specifications-Based System Level Test Cases 17
6.2 Traceability of Test Cases to Use Cases 17
6.3 Techniques Used for Test Generation 17
6.4 Assessment of the Goodness of Test Suite 17
Acknowledgements 18
References 19

List of Figures

Figure 1 – MakeManaged to Cloud Services 5
Figure 2 – WaterFall Methodology 7
Figure 3 – GitHub 10
Figure 4 - Use Case Model 11
Figure 5 - Client Server Architecture 13
Figure 6 - Class Diagram of Facade Style 14
Figure 7 - Sequence Diagram of Endpoint Flow 15
Figure 8 - Sequence Diagram of Example AWS Flow 15

List of Tables

Table 1 - Acronyms 6
Table 2 - RMP 8
Table 3 - Traceability Matrix from Requirements to Design 16
Table 4 - Traceability Matrix of Test Cases to Use Cases 17

1. Introduction

Our team has been tasked with implementing a makeManaged (Fig. 1) script for DXC Technology to monitor, make patchwork, and manage security aspects of their workload. This project management plan will include all of the aspects from beginning to end on how our team will create and implement DXC Technology's goal from our script. Even though the requirements list out three different phases of this project, our sponsor has made clear that the minimum viable product will only need to contain the monitoring aspect of our project and if time permits move onto patchwork as well as security, load balancing, and firewall monitoring.

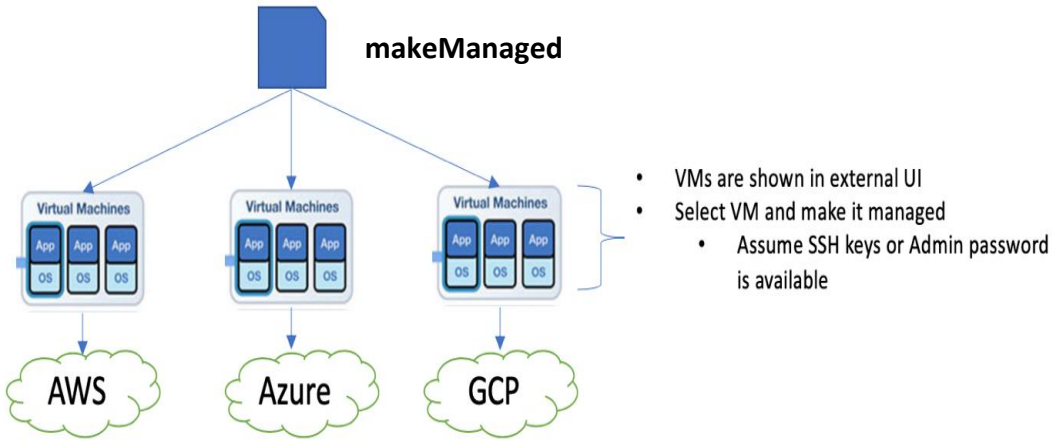

Figure 1 – MakeManaged to Cloud Services

1.1 Purpose and Scope

The purpose of monitoring software through Infrastructure as a System (IaaS) is to benefit many different companies that need help desk support, patchwork, and security. Since SaaS and PaaS are already managed by cloud providers, we will only be using IaaS. The scope will range from a working monitoring script able to communicate with a cloud service and relay information, notifications, and patchwork to DXC Technology. This will lighten the workload of their employees by automating a lot of tasks and opening up free time to allocate resources elsewhere.

1.2 Product Overview

A brief overview of the makeManaged program will be to use a virtual machine (VM) to communicate to AWS, Azure, and/or GCP cloud services over IaaS to monitor and the help desk and send notifications and email alerts or patch any necessary fixes.

1.3 Structure of the Document

- Create AWS or Google account to get our VM
- The monitoring tool will use open source - Nagios/Zabbix
- Install the monitoring tool in our cloud which will be Agentless Monitoring
- Work in API which will be barebone and then add dumb UI (time permitting)
    - OS monitoring - use case (runbook remediation - what to do with it, i.e. call an action or any abstract thing like write notification)
    - Software monitoring - using JAVA and the VM will use name only keys and no user passwords
    - Most likely no User Interface (everything will run through the VM to push automation to it)
- Make notification and send through email
- Create an incident and use serviceNow or Remedy Software
- Patching and Security (time permitting)

1.4 Terms, Acronyms, and Abbreviations

API	Application Programming Interface
AWS	Amazon Web Services
Azure	Microsoft Cloud Provider
D2C	Detect to Correct
F/NF	Functional or Non-Functional Requirement
GCP	Google Cloud Provider
GUI	Graphical User Interface
IDE	Integrated Development Environment
IT4IT	Reference Architecture for flow of data between IT managing systems
JDK	Java Development Kit
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
LAMP Stack	Linux operating system, the Apache HTTP Server, the MySQL relational database management system, and the PHP programming language - Stacks
Micro-Service	An architectural approach to building applications where pieces of an app work independently, but together
PMP	Project Management Plan
R2F	Request to Fulfill
RMP	Risk Management Plan
RTM	Requirements Traceability Matrix
SDLC	Software Development Lifecycle
UI/UX	User Interface / User Experience
VM	Virtual Machine
WBS	Work Breakdown Structure

Table 1 - Acronyms

2. Project Management Plan

2.1 Project Organization

The team consists of five members.

Po-Yu Liu: Group Leader/Project Manager
Zunayed Siddiqui: Front-end Developer
Yeswanth Bogireddy: Back-end developer
Ihfaz Tajwar: Testing and Quality Assurance
Zachary Tarell: Technical Writer

These are only the official roles. While they are defined, we are expecting to collaborate on different parts of the project together. We determined these roles for the main responsibilities based on strengths determined in our introduction meeting.

2.2 Life Cycle Model Used

Our team will implement a Waterfall Software Development Lifecycle (SDLC) to accommodate the DXC Technology, as well as the UTD requirements, for this project. At some point there might be a need to switch to a more agile approach which would then change into a hybrid model, but that depends on the process as it goes and measures up to our scheduling and guidelines put forth in this document.

The rationale for the Waterfall (Fig. 2) approach was taken into great consideration, and after two meetings with our team and DXC Technology, we concluded that the UTD requirements out-weighed any other plan in order to comply with meeting the SE 4485 standards given to us first. Also, learning the new monitoring technology was given great consideration and we all concluded that it would be best to attack this project one step at a time seeing as none of us are too familiar with the inner workings yet.

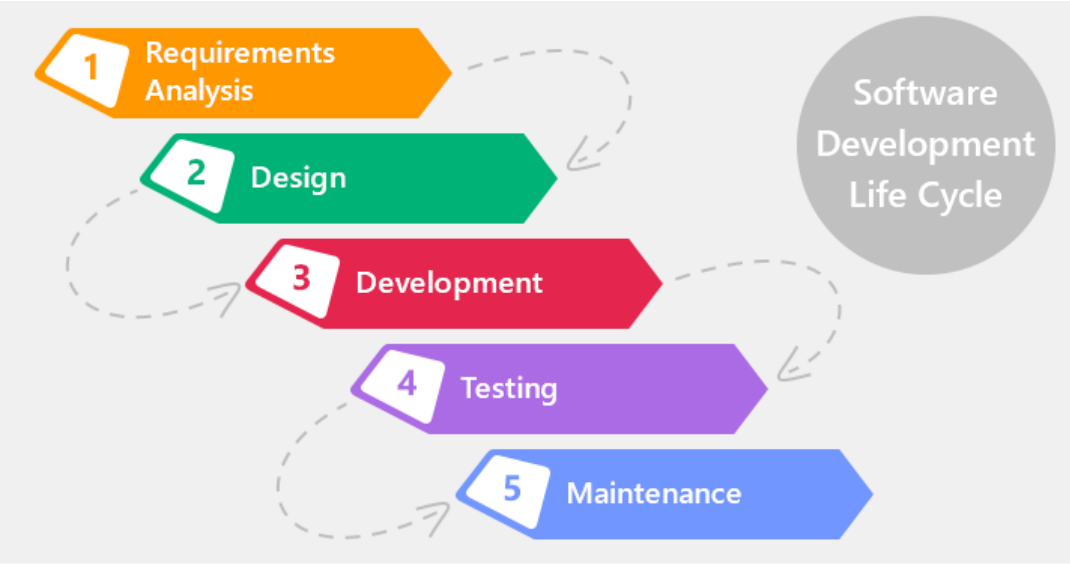

Figure 2 – WaterFall Methodology

2.3 Risk Analysis

2.3.1 - Risk Management Plan

Requirement Change	Change Request Form
Delay in requirement completion.	Group meeting and split the work.
Monitoring mechanism failure	Create script to email everyone when the failure happens
Unit testing delay	Group meeting and every member test different component

Table 2 - RMP

2.4 Hardware and Software Resource Requirements

Our team's main goal is to produce an action called make-managed that would instrument monitoring, patching (if time permits) across cloud providers and use late binding so that the implementation is one code base that works across multiple cloud providers.

2.4.1 - Hardware

Our project is fully cloud based which means we would not be requiring any hardware other than a computer and an internet connection for the programmer and the user to use our services.

2.4.2 - Software

For this project we will assume the Workload to be LAMP stack (Apache HTTP Server and MySQL server running on a single node or multiple nodes). Since this project completely exists on public cloud, both the client and the developer will require a set of software to use the services. The list of software needed for using and developing the cloud service:

- IntelliJ - IDE for backend development using Java
    - Including JDK (Java Development Kit) and JRE (Java Runtime Environment) to compile and run the app
- AWS - Virtual Machine to host the service on Cloud
    - Along with GCP as a secondary cloud provider
- Spring Boot - to build stand-alone and production ready application
- Jenkins - Open source pipeline to build, test and deploy our application
- Nagios / Zabbix - Open source software to monitor data and networks of the application
- Swagger - for API calling that is connected to spring boot and we will also have a UI
- GitHub - we will use our own repository for the project so that each of the team members can contribute and combine their code for production
- JIRA testing for bug tracking, issue tracking, and project management
- Maven - for dependency management and build automation of our Java project

User Interface (UI) is a stretch goal for us but for now we will be using the default UI by Swagger in order to make API calls and get results.

2.5 Deliverables and Schedule

We will have an estimated twelve weeks of time to work in. These will be divided up into deliverables in two week segments. While this is the initial plan, we may need to edit to stay consistent with our workflow.

- Deliverable 1 (due 9/11) - Finished plans (epic/user story creation)
- Deliverable 2 (due 9/25) - First iteration
- Deliverable 3 (due 10/9) - First iteration (continued)
- Deliverable 4 (due 10/23) - Finish first iteration & review with sponsor
- Deliverable 5 (due 11/06) - Second iteration, bug fixes
- Deliverable 6 (due 11/20) - Testing and quality assurance

2.6 Monitoring, Reporting and Controlling Mechanisms

Monitoring and control keep projects on track. The right controls can play a major part in completing projects on time. The data gathered also lets project managers make informed decisions. They can take advantage of opportunities, make changes, and avoid crisis management issues. Some tools that can be used are:

2.6.1 - Work Breakdown Structure (WBS) to break a project down into small units of work, or sub-tasks.

2.6.2 - Requirements Traceability Matrix (RTM) This will map, or trace, the project's requirements to the deliverables. This makes the project's tasks more visible. It also prevents new tasks or requirements being added to the project without approval.

2.6.3 - Control Chart monitors the project's quality. There are two basic forms of control chart – a univariate control chart displays one project characteristic, while a multivariate chart displays more than one.

2.6.4 - Review and Status Meetings further analysis problems, finding out why something happened. They can also highlight any issues that might happen later.

2.7 Professional Standards

Our standards are developed and approved through a consensus-based process that ensures that all interested stakeholders can participate. PMI publishes 13 globally recognized project management standards which will be followed by our team for this project:

- A Guide to the Project Management Body of Knowledge (PMBOK® Guide)
- The Standard for Program Management
- The Standard for Portfolio Management
- Organizational Project Management Maturity Model (OPM3®)
- Practice Standard for Project Risk Management
- Practice Standard for Earned Value Management
- Practice Standard for Project Configuration Management

- Practice Standard for Work Breakdown Structures
- Practice Standard for Scheduling
- Practice Standard for Project Estimating
- Project Manager Competency Development Framework
- Construction Extension to the PMBOK® Guide Third Edition
- Government Extension to the PMBOK® Guide Third Edition

2.8 Evidence the Document has been placed under Configuration Management

Our team is using GitHub for the Configuration Management (Fig.3) as shown below.

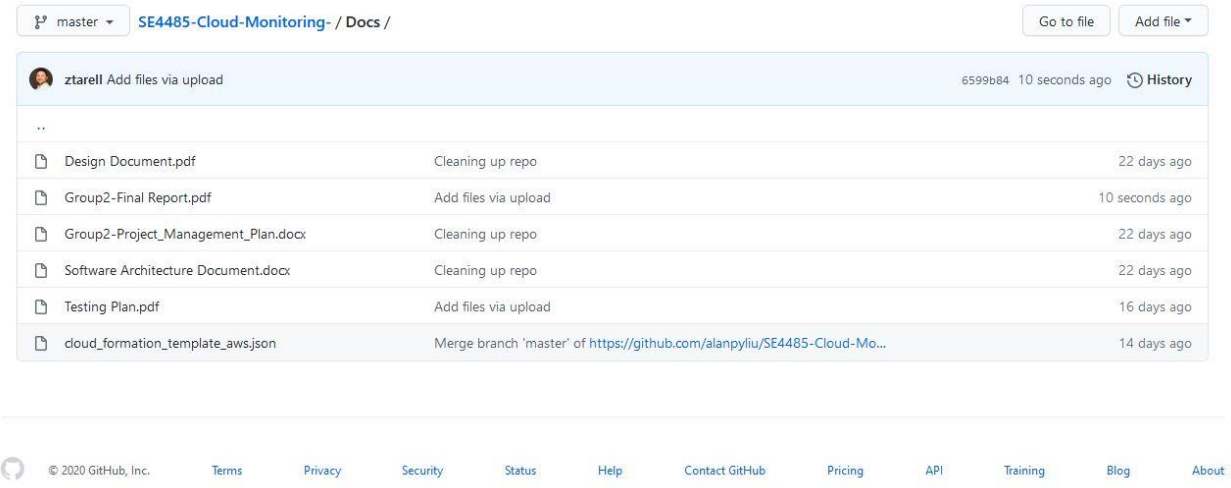

Figure 3 – GitHub

2.9 Impact of the Project on Individuals and Organization

Our project had an enormous impact on us as a group from gaining knowledge on cloud provider interfaces, ServiceNow interfaces and integration, and microservices has a way to monitor stacks and write correct programming code in our development as software engineers.

Chandra and DXC Technology also had the benefit of a working program that helped monitor LAMP Stacks and provisioning of monitoring tools. They now have a solid foundation to build off of as a company that would like to expand their monitor services to help allocate resources while saving time and money.

Lastly, UTD now has a successful program and guide to follow for future software engineering students to create and build their own final projects in the future. Overall, this was a success and the impact it will have is to provide the proper foundation for successful and useful tools for students, companies, sponsors, professors, and the university as a whole.

3. Requirement Specifications

3.1 Stakeholders for the System

- **DXC Technology** – Benefit from a workable program to build off of with better monitoring and provision their company
- **Chandra** – As our sponsor and an employee of DXC Technology profits off bettering company provisioning and helping UTD in showing students and faculty of success
- **Members of Group 2** – We benefit from gaining knowledge, getting good grades, and showing cloud provisioning as a tool to future employers
- **Dr. Wong** – As our professor this is important for teaching students and showing faculty that the company, sponsor, and project were important and successful
- **UTD** – Can show to other students, companies, faculty, and that this is a program worth taking, keeping, and teaching

3.2 Use Case Model

In our Use Case Model, the Client has 2 different cases:

1. To monitor a cloud provider where it receives requests and sends to ServiceNow
2. And controls created LAMP Stacks which receives incidents and sends to ServiceNow

And all Use Cases are resolved from ServiceNow as per each incident or request from the cloud provider being used.

3.2.1 – Graphic Use Case Model

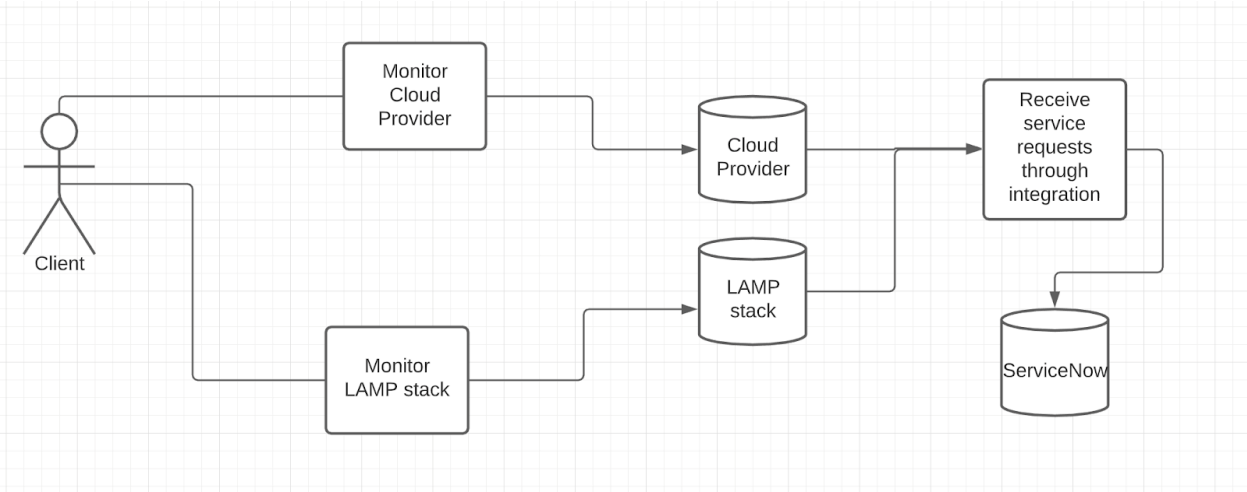

Figure 4 - Use Case Model

3.2.2 – Textual Description for Each Use Case

Case Name: Client uses monitoring tool to monitor their cloud provider
Participating Actors: Client
Entry Conditions: Client wants to monitor their systems on the cloud
Normal Flow of Events: [Log in, Look for incident reports, check platform, process ticket]
Exit Conditions: Sendoff ticket for verification
Exceptions: Wrong key, priority of incidents, duplicates
Special Requirements: Key Verification, Approval of Ticket Request

Case Name: Client uses monitoring tool to monitor LAMP stack
Participating Actors: Client
Entry Conditions: Client decides to use our tool
Normal Flow of Events: [Log in, Look for incident reports, Look for service failures, check platform, process ticket]
Exit Conditions: No more tickets to process, End of working shift
Exceptions: Wrong key, priority of incidents, duplicates, intentional bug fix disruption
Special Requirements: User should be admin, Key Verification

Case Name: Alert customer support with notifications or (patches) with a ServiceNow integration
Participating Actors: Client Customer Service, ServiceNow
Entry Conditions: Trigger of failures from ServiceNow
Normal Flow of Events: [Trigger, extract failure, process failure, ticket created, send ticket to client]
Exit Conditions: Ticket is processed then closed
Exceptions: Delays, False triggers, Duplicates
Special Requirements: UI between customer support to monitoring agent

3.3 Rationale for Use Case Model

Our client has specified through our initial meetings of certain requirements that we would need to meet with our final product for our project to be considered successful. We have identified key criteria through our use case model as well as identify key attributes like special requirements through these discussions with our client. Overall, we've created a thorough use case model through a collaborative approach with DXC Technology's representative, Chandra Kamalakantha.

3.4 Non-Functional Requirements

- The system should not have a response delay of more than 500ms
- The system should be up for 99.5% of the time
- The system should be able to handle up to 5 triggers per minute
- The system should notify users within 100ms of the trigger
- The system should create up to at least 5 tickets per second
- The system will be robust to fault tolerance and have a backup if there are crashes
- The system should make sure not to compromise the security of the users
- The system should run within a certain operational cost threshold

4. Architecture

4.1 Architectural Style(s) Used

We will be using a client-server architecture because of the event listener feature the monitoring tool will feature. Once events occur within the cloud provider, it will be received by our monitoring tool which will then send appropriate responses to ServiceNow.

4.2 Architectural Model

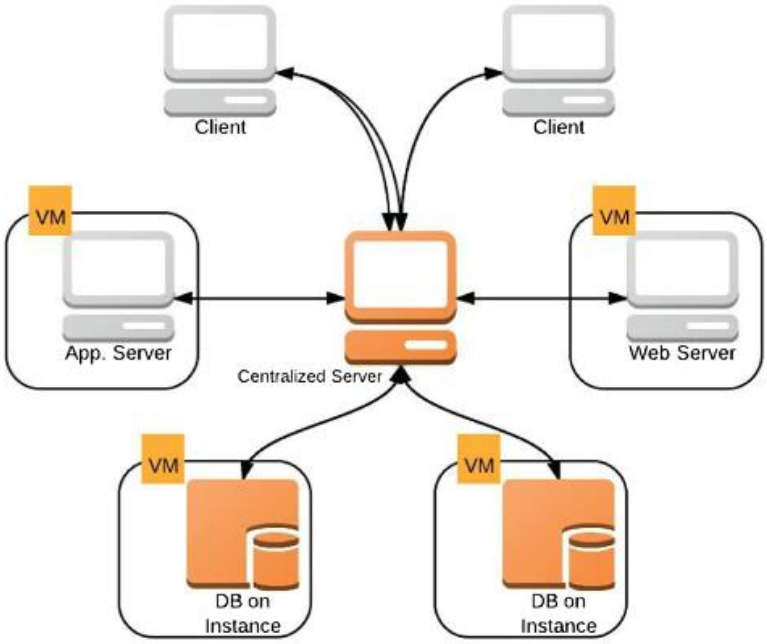

Figure 5 - Client Server Architecture

4.3 Technology, Software and Hardware Used

Our project is fully cloud based which means we would not be requiring any hardware other than a computer and an internet connection for the programmer and the user to use our services.

4.4 Rationale for Architectural Style and Model

Based on our conversation with our sponsor, we determined that this model would best fit our needs in monitoring the cloud providers. The client can use this core architecture to add more features like reporting and patching as needed. This model exemplifies usability, extensibility, and availability, therefore was the best one we could choose.

5. Design

5.1 GUI (Graphical User Interface) Design

When talking to our sponsor, Chandra, and as we monitored our process and established a better understanding of our goals in this project; we and Chandra decided a GUI was not needed for this project.

5.2 Static Model – Class Diagram

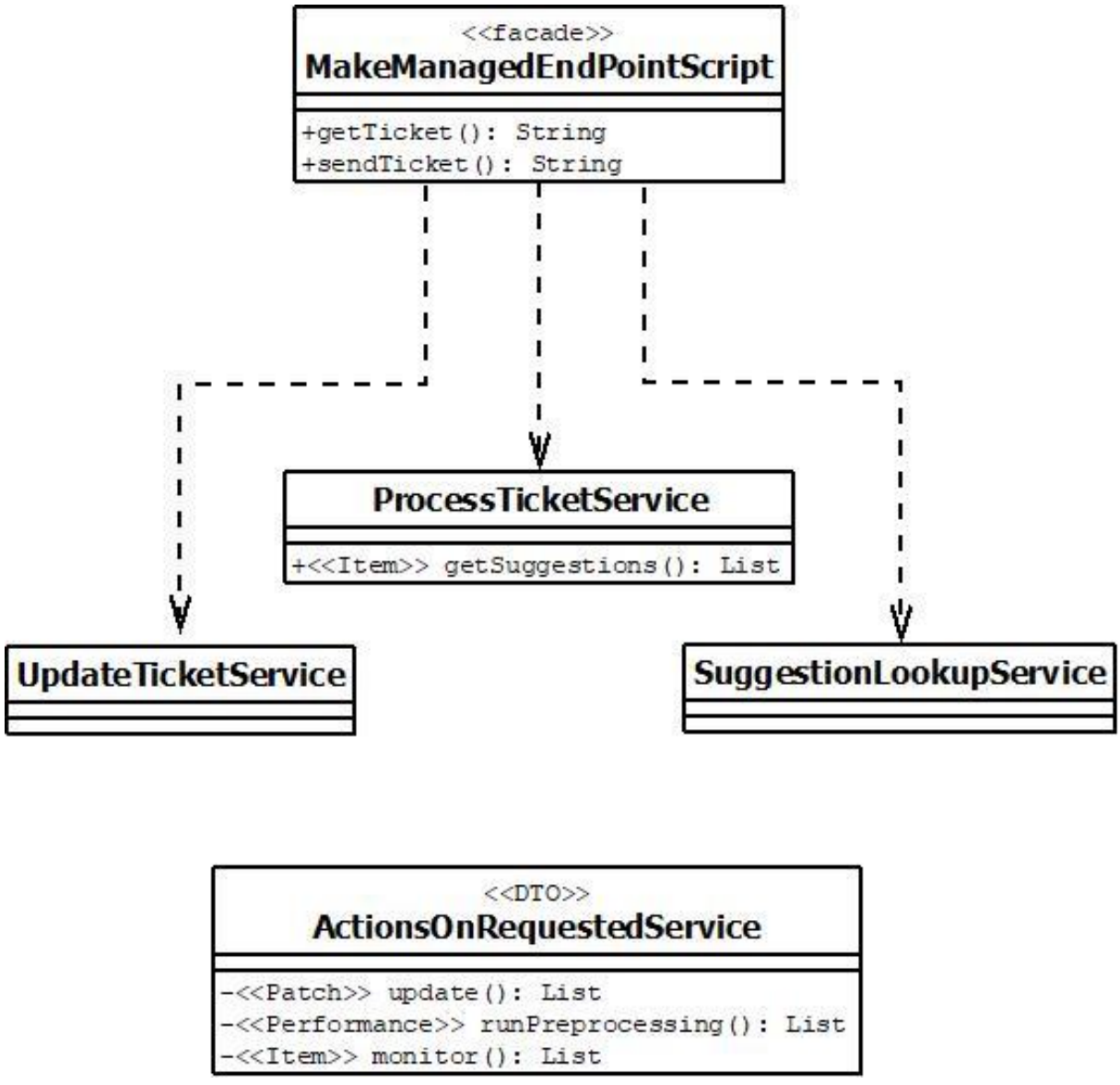

Figure 6 - Class Diagram of Facade Style

5.3 Dynamic Model – Sequence Diagrams

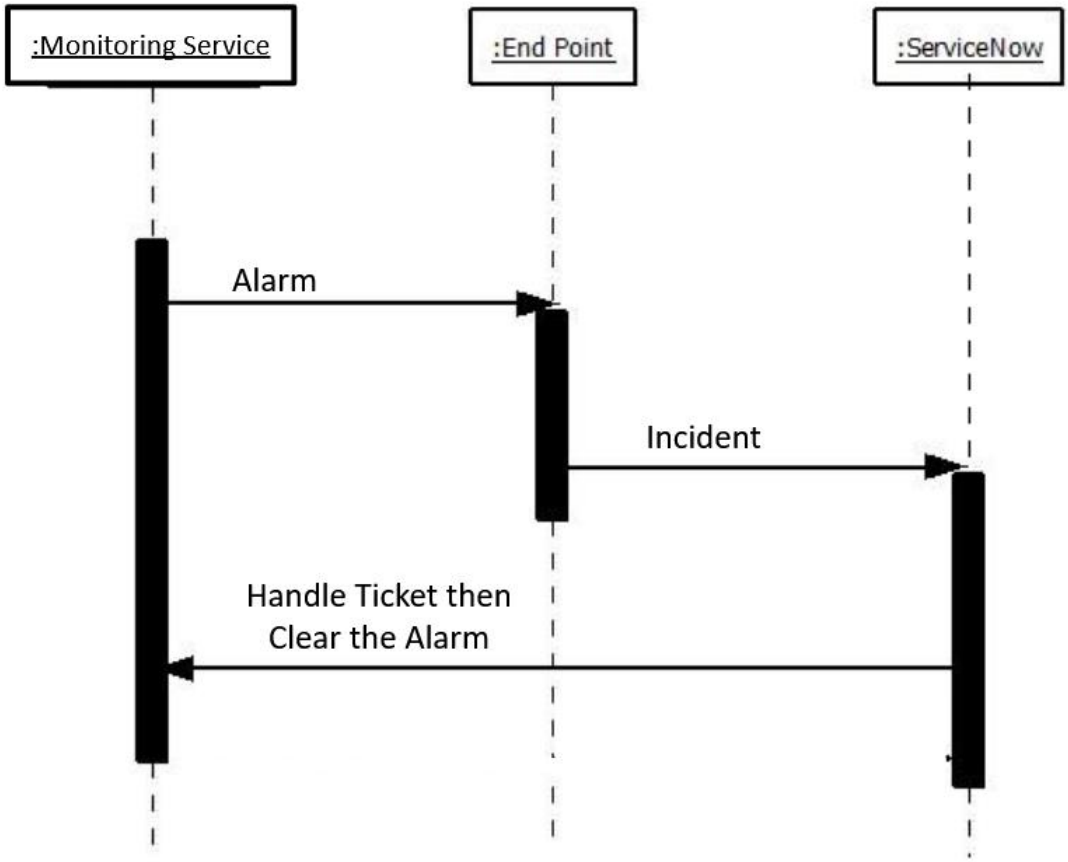

Figure 7 - Sequence Diagram of Endpoint Flow

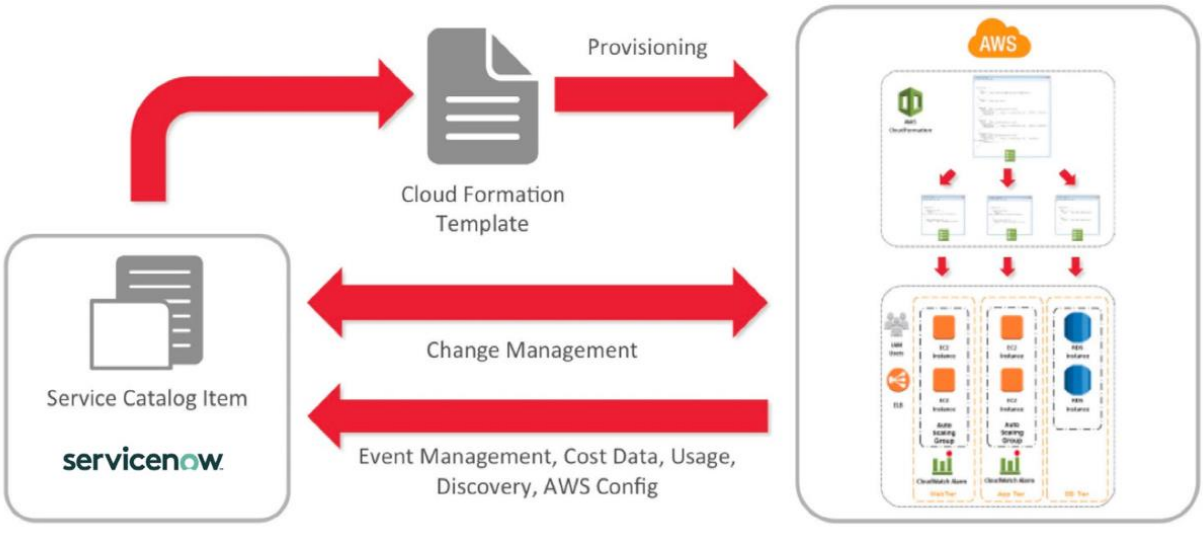

Figure 8 - Sequence Diagram of Example AWS Flow

5.4 Rationale for Detailed Design Model

Based on our conversation with our sponsor, we determined that this model would best fit our needs in monitoring the cloud providers. The client needs to be able to access the monitoring system which comes from our end point back to the cloud provider which sends out a call for a problem or issue that needs to be addressed. After more understanding and programming, we'll be able to update as needed to comply with our final design and architecture.

5.5 Traceability Matrix from Requirements to Detailed Design Model

Req ID	Req Description	Rationale	Type
1	The system should not have a response delay of more than 500 ms	Performance	NF
2	The system should be up for 99.5% of the time	Performance	NF
3	The system should be able to handle up to 5 triggers per minute	Performance	NF
4	The system should notify users within 100 ms of the trigger	Performance	NF
5	The system should create up to at least 5 tickets per second	Performance	NF
6	The system will be robust to fault tolerance and have a backup if there are crashes	Maintainability	NF
7	The system should make sure not to compromise the security of the users	Security	NF
8	The system should run within a certain operational cost threshold	Usability	NF
9	Client can use monitoring tool to monitor cloud provider	Client doesn't need to waste time on monitoring the cloud	F
10	Alert customer service with notifications (or patches) with a servicenow integration	Easier to free up other services instead of help desk	F
11	Client can use monitoring tool to monitor LAMP stack	Can write a script to send back to client	F

Table 3 - Traceability Matrix from Requirements to Design

6. Test Plan

6.1 Requirements/Specifications-Based System Level Test Cases

- Monitoring tool is able to receive events from the cloud provider
- Monitoring tool uses a secure connection to connect to cloud provider and LAMP stack
- Monitoring tool is able to receive events from LAMP stack
- Monitoring tool is able to load balance multiple events from cloud provider or LAMP stack
- Monitoring tool does not reveal data from ServiceNow to developers or non-admin staff.
- Monitoring tool is able to send an incident to ServiceNow when it receives an event
- Monitoring tool is able to send an incident within 1 minutes

6.2 Traceability of Test Cases to Use Cases

Test No.	Test Case	Use Case
1	Monitoring tool is able to receive events from the cloud provider	Client uses monitoring tool to monitor their cloud provider
2	Monitoring tool uses a secure connection to connect to cloud provider and LAMP stack	Client uses monitoring tool to monitor LAMP stack
3	Monitoring tool does not reveal data from ServiceNow to developers or non-admin staff	Alert customer support with notifications or (patches) with a servicenow integration
4	Monitoring tool is able to send an incident to ServiceNow when it receives an event	Client uses monitoring tool to monitor LAMP stack
5	Monitoring tool does not reveal data from ServiceNow to developers or non-admin staff.	Alert customer support with notifications or (patches) with a servicenow integration
6	Monitoring tool is able to send an incident to ServiceNow when it receives an event	Alert customer support with notifications or (patches) with a servicenow integration
7	Monitoring tool is able to send an incident within 1 minutes	Alert customer support with notifications or (patches) with a servicenow integration

Table 4 - Traceability Matrix of Test Cases to Use Cases

6.3 Techniques Used for Test Generation

We would want to do black box testing to ensure that the functional requirements are met. These are pretty straightforward to test the functionality of the product. But we also need white box testing as some of the test cases like hiding data means the code needs to be tested.

6.4 Assessment of the Goodness of Test Suite

After starting our stress test to trigger an alarm from AWS and GCP, the suite was successful because it passed by sending a request to our endpoint code, where we were able to create an incident and send it to ServiceNow which handled the incident and cleared the alarm.

Our sponsor, Chandra was pleased with its effectiveness and from this test suite we were able to confirm that our provisioning of the cloud provider worked. Our code was successful in receiving a request and turned it into an incident. From there ServiceNow was able to handle our incident and clear the CPU-Alarm Stress Test.

Acknowledgements

We cannot express enough thanks and gratitude to our sponsor from DXC Technology, Dr. Chandra Kamalakantha. Not only was he an inspiring presence throughout the entirety of the project, but he was also with us for every scheduled meeting, every newly planned meeting, and answered emails right away if we had any questions or problems that needed resolved.

Our completion of the project was one of a every team member working together and Chandra taking extra time to teach and listen to us. He was invaluable and dedicated to see us succeed and we will never take that for granted and never forget how much that meant to us. So, from all of us, we say thank you.

Finally, we would also like to mention Professor Weichen E. Wong and Teaching Assistant Linghuan Hu for their valuable feedback and taking the time to provide us with the tools and instructions necessary to make this project a success. So, from all of us in Group 2, thank you all very much.

References

"An overview of the Commercial Cloud Monitoring Tools: Research Dimensions, Design Issues, State-of-the-Art." https://my.ece.msstate.edu/faculty/skhan/pub/A_K_2014_COMP.pdf

"AWS Overview" https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc

"Controlling the Cloud: Requirements for Cloud Computing." F5, 2010. www.f5.com/services/resources/white-papers/controlling-the-cloud-requirements-for-cloud-computing.

"Detect to Correct (D2C) Value" https://pubs.opengroup.org/it4it/refarch20/chap08.html

"Getting Started | Applications on the ServiceNow Platform." YouTube. Sept 27, 2018. https://www.youtube.com/watch?v=ofKrvQ32AXo

"Google Compute Engine documentation" https://cloud.google.com/compute/docs

"IT4IT - Reference Architecture Standard" https://www.goodelearning.com/courses/digital-transformation/it4it-foundation/what-is-it4it

"Project Monitoring and Control Techniques." PMP® Blog - Project Management Professional Training, 21 Oct. 2019, www.projectmanagementqualification.com/blog/2019/10/21/project-monitoring-control/

"Request to Fulfill (R2F) Value" https://pubs.opengroup.org/it4it/refarch20/chap07.html

Ritter, Danielle. "SDO: Project Management Institute." PMI - Project Management Institute, www.standardsportal.org/usa_en/sdo/pmi.aspx.

"Service Portal" https://www.servicenow.com/products/service-portal.html

"ServiceNow Cloud Management" https://www.servicenow.com/products/cloud-management.html

Sharma, Nishant. "Software Architectural Structures and Design Patterns". Medium. Web. July 25, 2019. https://medium.com/ios-expert-series-or-interview-series/software-architectural-patterns-design-structures-c5692fe8affc

"Software Architecture and Design" Tutorials Point. Web. 2020 https://www.tutorialspoint.com/software_architecture_design/introduction.html

"What are microservices really all about? - Microservices Basics Tutorial." YouTube. Dec 31, 2018. https://www.youtube.com/watch?v=j1gU2oGFayY

"What Is Non-Functional Requirement? Types and Examples." Guru99, www.guru99.com/non-functional-requirement-type-example.html.