SW Architecture Syda[1468]

Table of Contents

1 Introduction[1469]

1.1 Purpose[1470]

Specify the purpose of this document and its intended audience.

The purpose of this document is to define major structural components of the software, their externally visible properties, and

the relationship among them. This document transforms the specific software system requirements into the a documented

architecture for the product. This information shall be the basis for unit implementation and testing and shall facilitate

software maintenance.

The intended audiences of this document are the device development team.

The Software Project Lead or designee is responsible for creating and updating this document.

Enter additional information if applicable.

1.2 Scope[1471]

The scope of this document is to describe the software architecture for SYDA (Synthetic Data generator). This document

will present a functional overview for each of the major software subsystem; whereas more detailed information is included in

the appropriate software design document for each software subsystem/component mentioned in this document. This

document describes the software architecture and high level abstraction that includes but is not limited to the following topics:

- A high-level abstraction and decomposition of logical components into software subsystems, including decomposition of each subsystem into major components, and description of important concepts.
- Decomposition of the major software components within each subsystem.
- Description of class structure/hierarchy and associations
- Description of the message/data flow between objects
- Description of the interface between the subsystems

2 Definitions, Acronyms and Abbreviations[1472]

a few Examples:

- Software Item – Any identifiable part of a computer program, ie, source code, object code, control code, control data or a collection of these.
- Software System –An integrated collection of “Software Items” organized to accomplish a specific function or set of functions.
- Software Unit – A “Software Item” that is not subdivided into other items.
- SOUP – Software of unknown provenance: Software item that is already developed and generally available and that has not been developed for the purpose of being incorporated into a medical device (also known as off the shelf software “OTS”) or a software item that was previously developed for which adequate records of the development process are not available.

3 Applicable and Reference Documents[1473]

- Product Requirements Document
- Software Specification Document
- Data Management Plan

4 Overview of Software Architecture[1474]

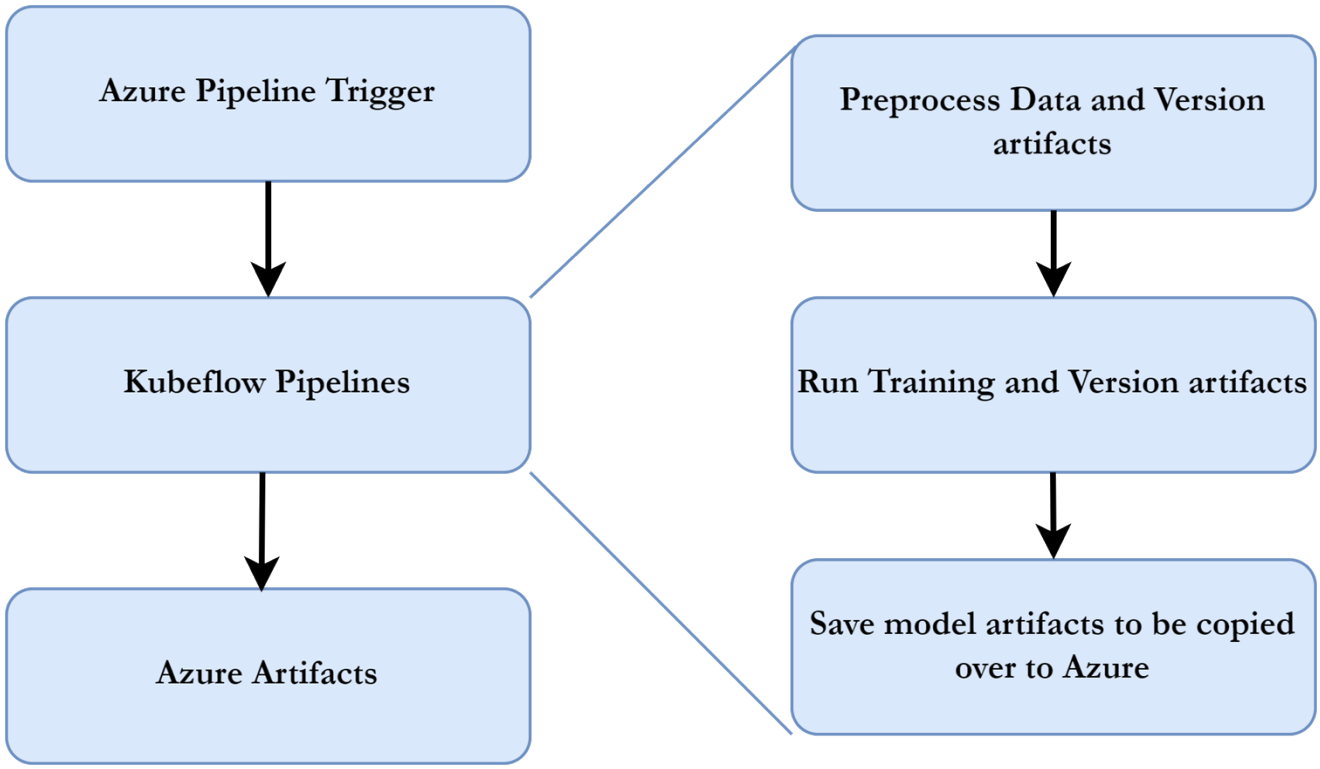
Give an overview of major software structural components, their key responsibilities, their external visible properties, and the relationship among them.

Enter here.

4.1 Software Operating Environment[1475]

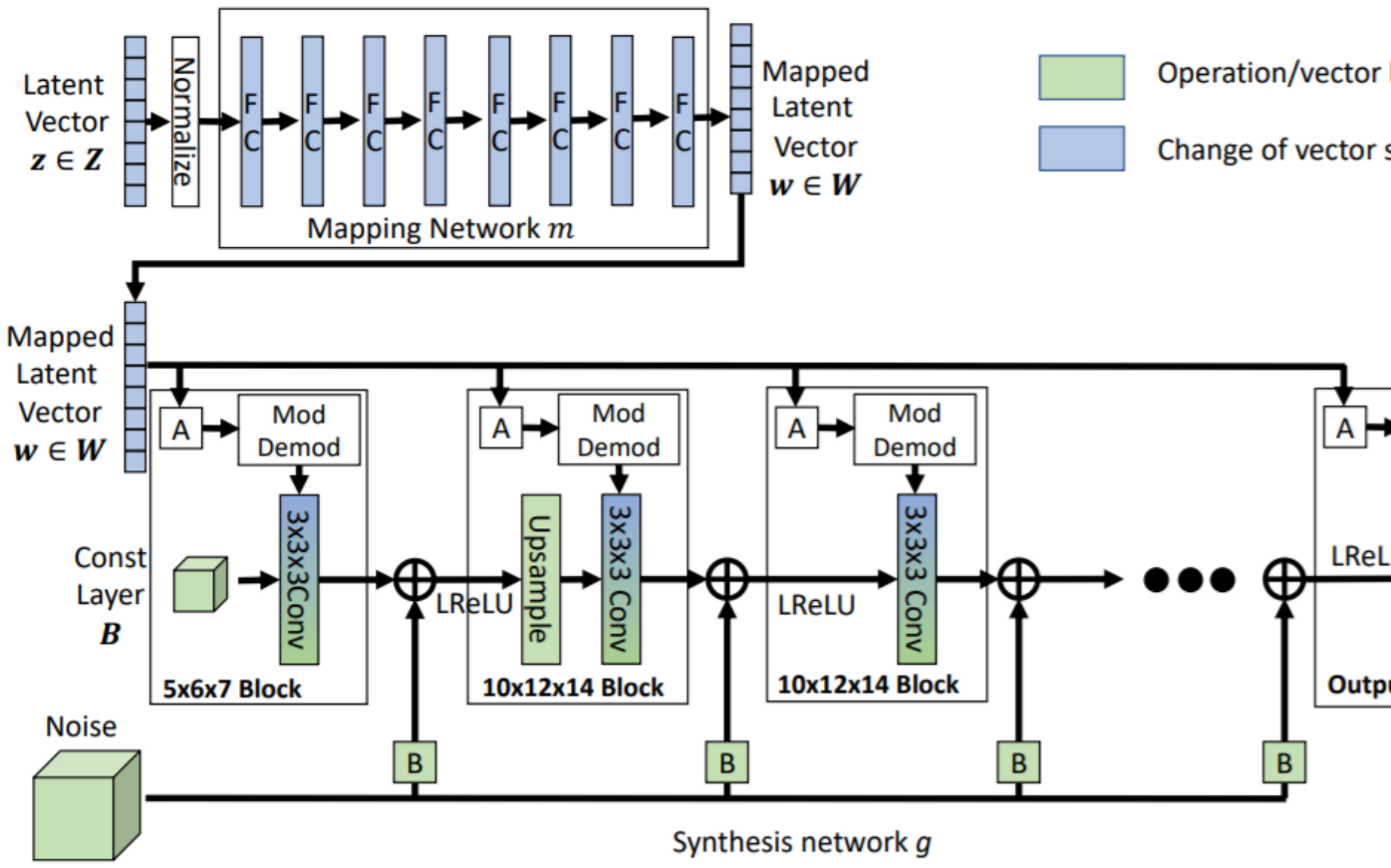
- Programming Language: All Software Components are written in Python, Automatic pipelines are triggered using Azure environments (Shell, Bash, Python).
- Hardware platforms: A Combination of using Azure Devops platform for triggering automatic pipelines and Google Cloud for running our developed code
- Operating system (if applicable): Linux.
- If Off-the-Shelf software is used: No.

5 Detailed Architecture Design for Software Items[1476]



Shown above is a high level Software Architecture (currently) for developing the Model artifacts that can generate 3d imagaes of Abdominal CT scans. The Basic Flow shows we use Azure Pipeline to Trigger our training pipeline and store th eresult artifacts in Azure's Artifacts section.

5.1 Model Architecture and Training[1477]



The Model is a Stylegan based architecture that takes in a Noise vector and a Latent vector to generate CT scan data of resolution 128x128x128. The model is trained on the Data Using 8 GPUs. We calculate a Softplus based Generator Loss and Discriminator Loss with R1 regularization. once th e model has reach its saturation point we stop the training and save the best model. SSIM and perceptual difference is used to evaluate how the images can have realistic features and structural similarity.

5.1.1 Behaviour[1478]

The Software Component runs the training of the described architecture above.

5.1.2 Interfaces[1479]

this Software interacts with Google cloud platform to run training and hosting the curated Data for more details on Data refer the Data management Plan.

5.2 Enter Software item title here - Add more items as required[1480]

5.2.1 Behaviour[1481]

5.2.2 Interfaces[1482]

6 Detailed Architecture Design for SOUP [B, C][1483]

6.1 Python packages[1484]

Tensorflow
Pytorch

pytorch-lightning

monai

Tensorboard

google cloud SDK

6.1.1 Functional and Performance Requirements[1485]

Tensorflow: Used for handling TF record datasets and communication between the data and the training cluster. This is used since converting the versioned data into a TFrecord for parallel IO operations

Pytorch: Model training code is built on this framework

Pytorch-Lightning: this is a simplified wrapper over Pytorch to code model training in a n organized way and also handle hardware communications with minimal coding.

Monai: library for applying image transforms on Medical images

Tensorboard: Used for logging model metrics during training.

google-cloud-sdk: necessary library to install components needed to communicate with google cloud services.

6.1.2 Hardware and Software Support[1486]

base support listed out in our Software Specification Document. More details will be added further down the Development Cycle

7 Segregation of Software items [C][1487]

Identify the segregation between software items that is essential to risk control, and describe the methods used to ensure effective segregation of those software items.

Enter here.

An example of segregation is to have software items execute on different processors. The effectiveness of the segregation can be ensured by having no shared resources between the processors.