TensorFlow UI

Try Catch Degree

August 19, 2021

Name	Surname	Student Number
Felipe	Kosin Jorge	u17291195
Werner	van Rensburg	u15118046
David	Walker	u19055252
Siviwe	Lechelele	u18221409
Wessel	Kruger	u18014934

1 Introduction

TensorFlow is a powerful and well-rounded machine learning system, used by many different groups for an immense number of tasks around the globe. With the advent of AI and ML becoming so commonplace, it offers one of the most compelling entry points into such a world. However, due to the code-driven nature of the framework, it tends to be relatively unfriendly to new users.

To remedy this situation, a UI should be developed. This UI should be intuitive and good-looking, presenting a slick and effortless means of creating and training ML models for anybody with a basic understanding of the principles behind TensorFlow. This UI is what we are developing. It uses the idea of nodes as code segments, which can be linked together in a similar style to Scratch programs, to visually illustrate a model being created.

Our aim with this project is to make ML more accessible, and therefore allow the many benefits of machine learning to be used by more people, through lowering the barrier to entry of writing and training a model.

2 Functional Requirements

2.1 Glossary

A brief explanation of terms related to our requirements

- **Node**
  A node is defined as a singular "object" within the workspace. It can be functionally compared to functions or variables from traditional programming languages. It is comprised of either a "base" (making it a constant) or a set of sub-nodes, which combine to give it its total functionality.
- **Project**
  A project is a combination of many nodes in a workspace. It is the total set of functionality which is achieved by combining a particular set of nodes in a specific way.
- **Community Library**
  The Community Library is a faux-database/repository which contains uploaded Nodes created by the various users of the system, available for re-use and re-download to any user who wishes to replicate that functionality.

2.2 Must Have

- R1 - User must be able to create Nodes.
- R2 - User must be able to delete Nodes.
- R3 - User must be able to merge Nodes.
- R4 - User must be able to update Nodes.
- R5 - User must be able to map Nodes to TensorFlow functionality.
- R6 - User must be able to import Projects.
- R7 - User must be able to generate code from the existing Project or Node structure.

2.3 Should Have

- R8 - The system should allow the user to access a community built library of public nodes/projects to be used in a project

2.4 Could Have

- R9 - Export to TFHub.
- R10 - Export/Open in Jupyter Notebook/Colab
- R11 - Run in external Python environment and display output in web app

3 User Characteristics

3.1 Basic User

3.1.1 Characteristics

- No experience with code or machine learning
- Curious about the tool presented to them
- In need of clear, concise, and intuitive UI

3.1.2 User story

As a computer enthusiast with little to no experience, I would like a place to go and experience the various programming tools I can find online. I would like a way of using powerful tools like TensorFlow, without spending hours on learning a new language or how to use this specific tool.

3.2 New Machine Learning User

3.2.1 Characteristics

- Understanding of machine learning but little to no code or TensorFlow experience
- Wants to develop understanding of machine learning and build useful models without learning to code in TensorFlow
- In need of ability to view mapping between UI and code, and clear explanations of what's happening behind-the-scenes

3.2.2 User story

As a person who has the basic programming knowledge, but not advanced machine learning experience, I want an Interface that will allow me to visualize what TensorFlow can do so that I don't require advanced machine learning knowledge to use TensorFlow. This will give me the opportunity to use this tool, without the need to spend hours of research on how to use TensorFlow.

3.3 Advanced Machine Learning User

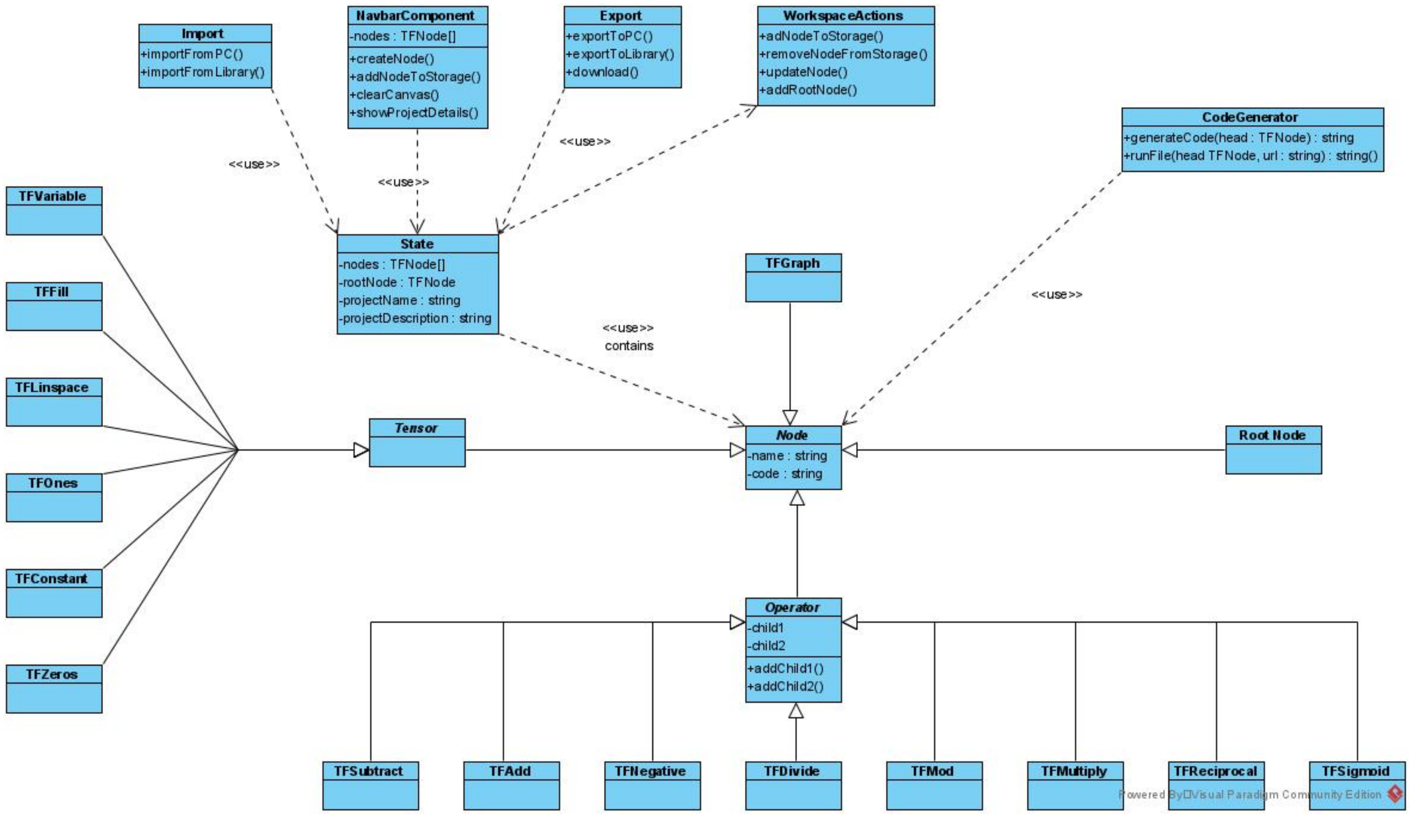
3.3.1 Characteristics

- Experienced with machine learning and TensorFlow
- Wants to streamline workflow
- In need of full-featured, powerful tool which produces code compatible with past work

3.3.2 User story

As a Machine Learning engineer I would like to use TensorFlow without the need to write code myself, so that the process of producing TensorFlow code can be easier and faster.

4 Class diagram



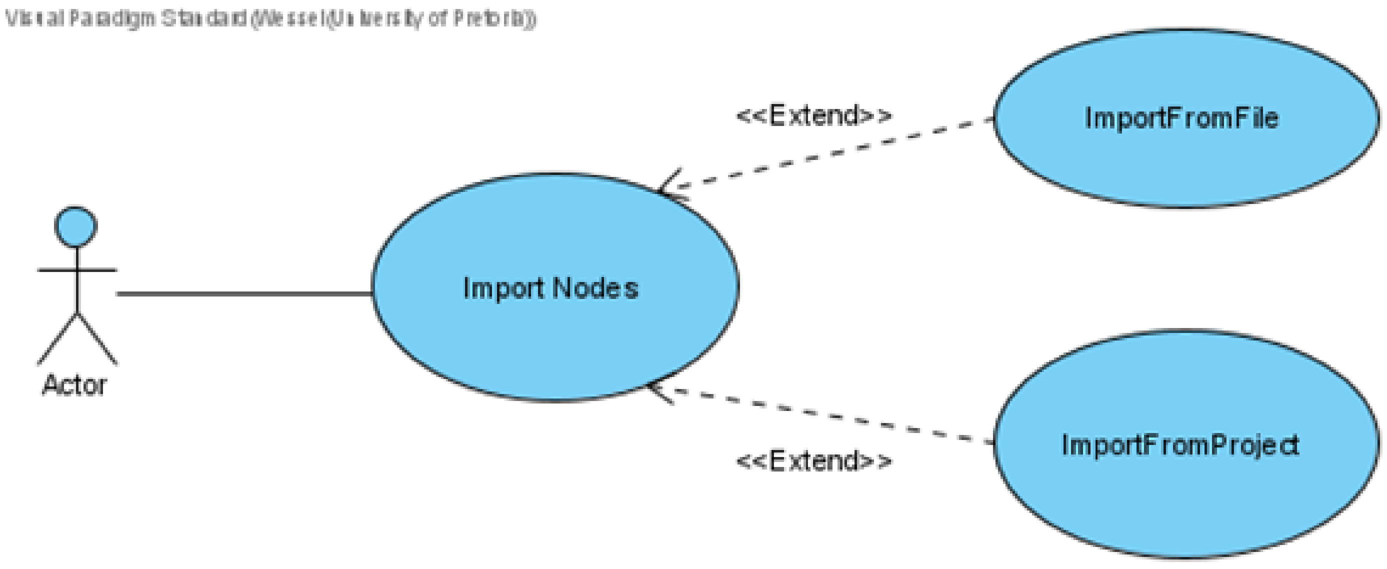
5 Subsystems

5.1 Disambiguation

In cases where "controller" is referenced by diagrams, the "controller" component represents the active component implementing the subsystem under which the diagram falls. Due to the nature of the import system used, there is no one central "controller", but many smaller sub-sections which encapsulate the idea of controller logic.

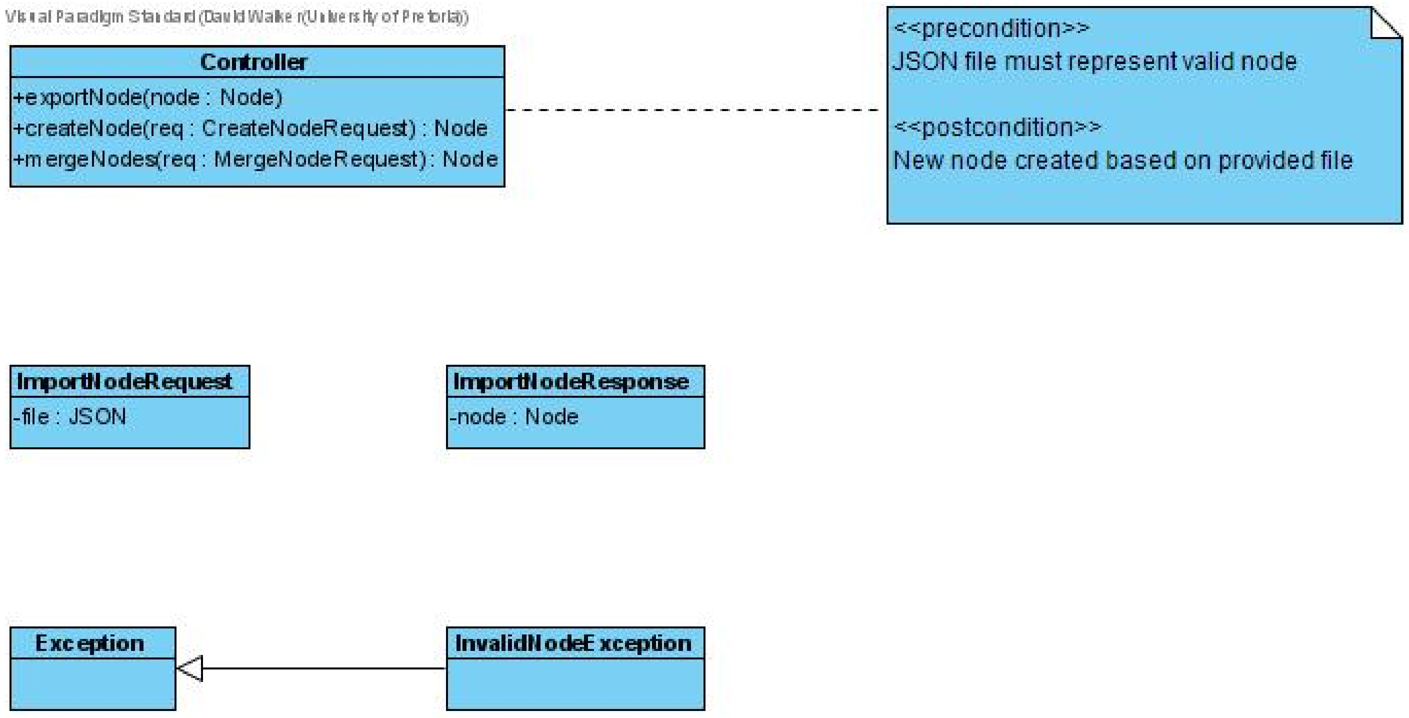
5.2 Import Subsystem

5.2.1 Use cases



- Import from Community Library

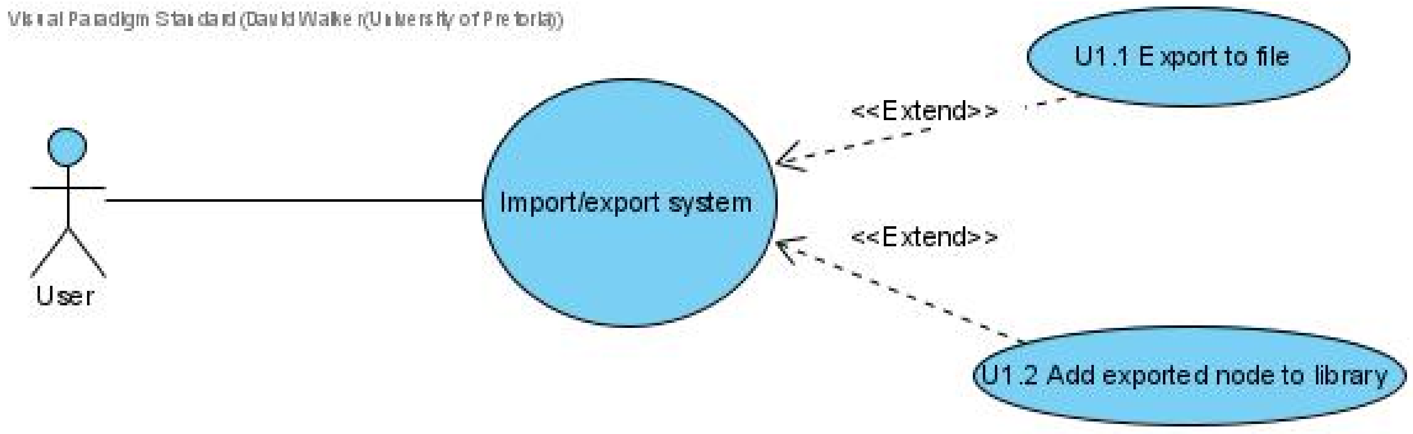
5.2.2 Service contracts


Visual Paradigm Standard (David Walker(University of Pretoria))
Controller
+exportNode(node : Node)
+createNode(req : CreateNodeRequest) : Node
+mergeNodes(req : MergeNodeRequest) : Node
<<precondition>>
JSON file must represent valid node
<<postcondition>>
New node created based on provided file
ImportNodeRequest
-file : JSON
ImportNodeResponse
-node : Node
Exception
InvalidNodeException

Precondition: The JSON file must be a valid node	
Actor: User	**System: TensorFlow UI**
1. The user clicks on the import button. 2. The user browses a file for selection	0. The main page of the WebApp is loaded for the user. 3. The system loads the file chosen into a working prototype.
Postconditions: -A new project based on the file imported is created, consisting of the file's contents	

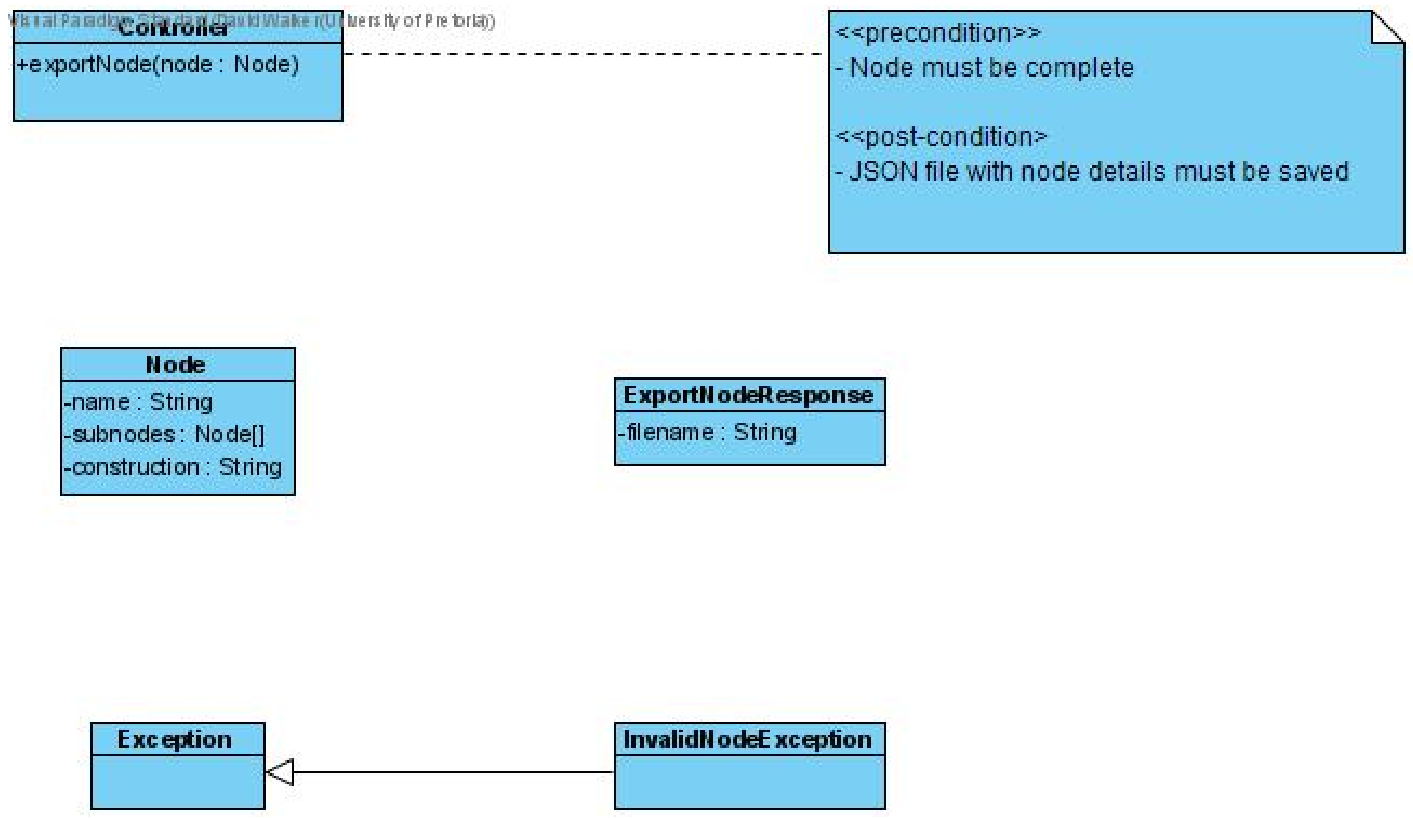
5.3 Export Subsystem

5.3.1 Use cases



Visual Paradigm Standard (David Walker(University of Pretoria))
User
Import/export system
<<Extend>>
U1.1 Export to file
<<Extend>>
U1.2 Add exported node to library

5.3.2 Service contracts



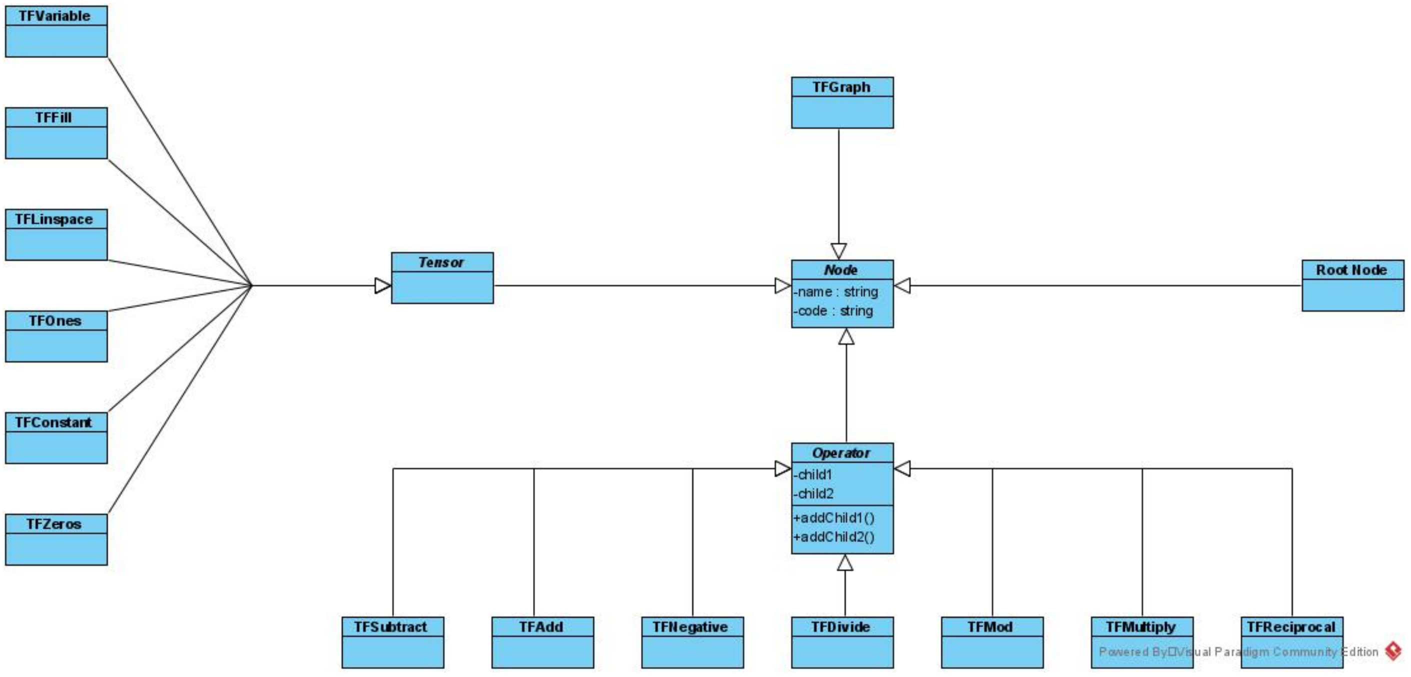
Precondition: There must be at least one Node to be exported.	
Actor: User	**System: TensorFlow UI**
0. The user clicks on the Export button when done working.	
1. The user selects the location of the export.	
2. The user chooses to export to the community library as an option	
	3. The system creates a downloadable JSON file for the user to Download.
	4. The system exports this created file to the community Library.
5. The user can continue working.	
Postconditions: -A JSON file with node details must be created and available for download.	

5.4 Node Subsystem

5.4.1 Use cases

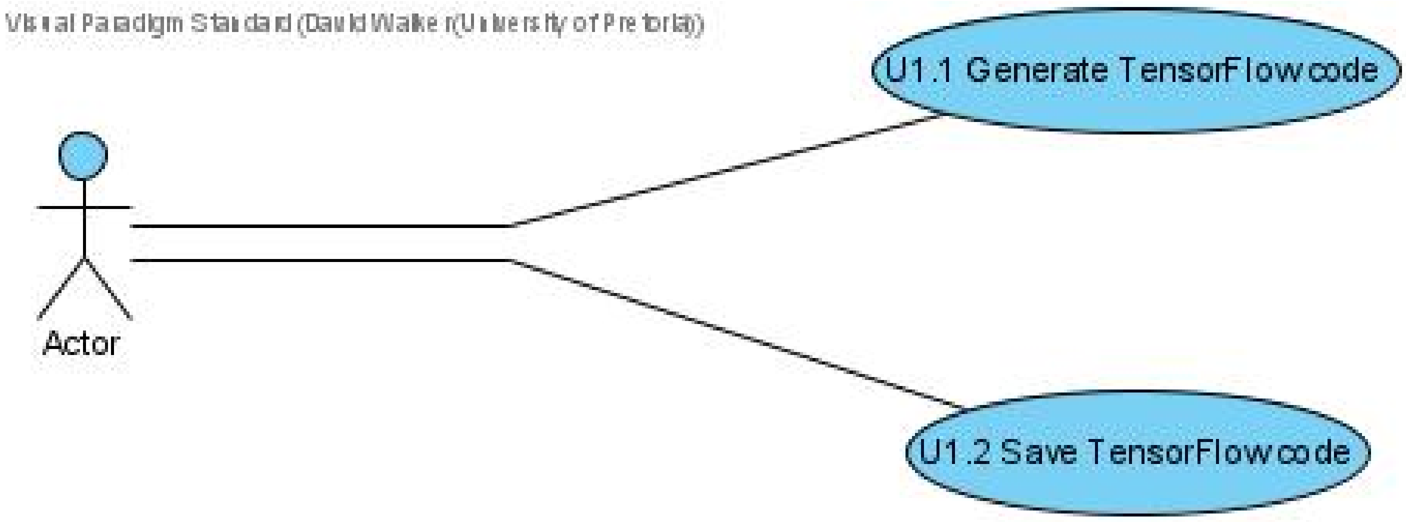
- Get node information
- Use node functionality
- Store child nodes for input and output
- Represent the culmination and combination of prior nodes used in node creation

5.4.2 Service contract



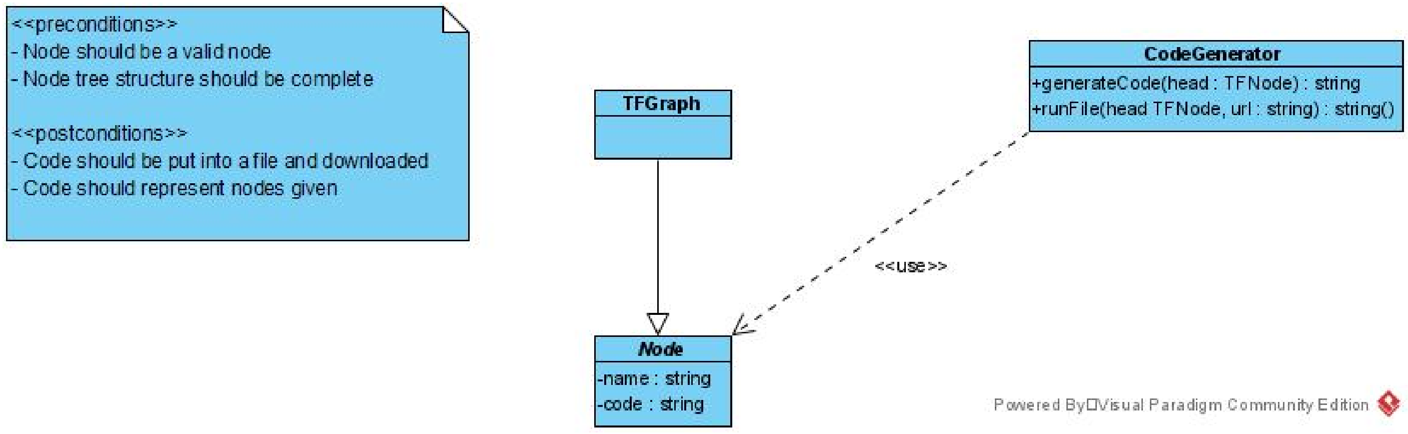
5.5 Code Generation Subsystem

5.5.1 Use cases


Visual Paradigm Standard (David Walker (University of Pretoria))
U1.1 Generate TensorFlow code
U1.2 Save TensorFlow code
Actor

- Edit TensorFlow code
- Show direct relationship between code segments and node objects

5.5.2 Service contracts


<<preconditions>>
- Node should be a valid node
- Node tree structure should be complete
<<postconditions>>
- Code should be put into a file and downloaded
- Code should represent nodes given
TFGraph
CodeGenerator
+generateCode(head : TFNode) : string
+runFile(head TFNode, url : string) : string()
<<use>>
Node
-name : string
-code : string
Powered By Visual Paradigm Community Edition

Precondition: A node is selected.	
Actor: User	**System: TensorFlow UI**
0. The user clicks on a Node. 1. The user selects 'Map functionality' 3. The user selects a functionality.	2. The system shows a list of TensorFlow functionality to choose from. 4. The system links the specified functionality with the selected Node.
Postconditions: -The new functionality is part of the node.	

6 Architectural Quality Requirements

6.1 Testability

6.1.1 Project Testing

The user must be able to test the entire system as a whole and any code generated by the system at any given time. In this vein, the project needs to have built-in testing functions wherever possible, which can be run by automated testing software to ensure maximum functionality with minimal testing effort.

6.1.2 Node Testing

The user must be able to individually test created functionality and individual nodes at any time, including any node's interaction with given child nodes. The user must be able to test generated code for a single node.

6.1.3 Multiple Node Testing

The user must be able to test the functionality of a group of nodes or merged nodes at any given time.

6.2 Performance

6.2.1 Load times

The program should open in no more than 125 percent of the time it takes to open only a web browser. For example, on a computer which takes 2 seconds to open a web browser, the TensorFlow UI should take no more than 2.5 seconds to open.
Furthermore, the code generation algorithm shouldn't require additional sub-routines above and beyond a simple generation step for each node, so in the worst case code generation from our node tree structure should occur in O(n) time.

6.2.2 Interaction performance

Interactions with the program's menus and controls should be practically instantaneous, meaning that all menus should appear within half a second of being clicked, and all new screens should do the same. Furthermore, animations should be fluid, smooth enough to be perceived as movement to the human eye.

6.2.3 Resource usage

The program should not consume more than 500MB of RAM when open.

6.3 Usability

6.3.1 Learnability

They system should be intuitive to the user, the user should at no time not know what to do next. To this end, each interface component should in some way be labelled descriptively.

6.3.2 Efficiency

The system needs to be structured in a way that the logical steps after a process has been completed is obvious to the user. That is, there should be visual feedback for all actions taken.

6.3.3 Memorability

The system needs to be memorable to the user. The user does not need to relearn how the system works. The system needs to remember the user and their current working state.

6.3.4 Satisfaction

The system needs to be designed in a way that exploring the rest of the system is enjoyable to the user.

6.4 Scalability

We do not expect multiple users to work on the same system since users will be downloading the UI and working from their own computer. Therefore scalability for this project will be examined in terms of project size and concurrent saves to the community library:

6.4.1 Project Size

Each individual project can have a different amount of nodes. This means that one project may have 10 nodes and another may have up to 100 nodes.
As such it is important that the chosen architecture behind the project building aspect of the TensorFlow UI can allow for scaling to keep up with the different sizes of projects. Any user should be able to perform equal levels of computation to an equivalent TensorFlow Python application. this end, no hard caps should be imposed on a maximum number of users.

6.4.2 Community Library

This will be a single library that will store projects from multiple users. We expect 50 initial users, but as the project popularity grows, it will increase from 10 to 100 and eventually 1000+.
As such it is important that the chosen architecture can allow for vertically scale-out to ensure the the demand for saving and accessing the community library can be met. To this end, no hard caps should be imposed on a maximum number of users.

6.5 Maintainability

Maintainability is centered around how easy or difficult it is for the system to comply with standards or conventions regarding maintainability, There are four main sections which are looked at to determine the Maintainability, namely: analysability, changeability, stability and testability. Some of these have already been discussed.

6.5.1 Maintainability of the System

It is difficult to quantify or be able to easily verify that a system is actually maintainable, one measure that can used to test for it is 'The complete fitting function', the higher this number is, the more maintainable a system is deemed to be. This is measured using the formula:

6.5.2 Code Duplication

Excessive amounts of duplication are detrimental to the systems maintainability, the duplication of code affects the analysability and changeability of the system. For our system, we will aim to have the overall code duplication in the code to be less than 8%, as this would be considered as well designed according to the maintainability standards. and changeability.

6.5.3 Unit Testing

Having good and extensive unit testing has a significant impact on the maintainability of the system. Unit tests raise testability, stability (as they ensure the system does not break at any point) and they also improve the analysability of the system. Thus for each use case, we expect the system to have unit test coverage of 80% or more of the use case.

7 Use cases list

- U1.1 Import from file
- U1.2 Export to file
- U2.1 Export to file
- U2.2 Export to library
- U3.1 Get node information
- U3.2 Use node functionality
- U3.3 Store child nodes inside given nodes
- U3.4 Represent culmination and combination of prior nodes used in node creation
- U4.1 Generate TensorFlow code
- U4.2 Save TensorFlow Code

8 Traceability Matrix

		Requirements										
		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
Use cases	U1.1						x		x			
	U1.2						x		x			
	U2.1						x					
	U2.2						x		x			
	U3.1	x			x							
	U3.2	x	x	x	x	x		x				
	U3.3			x	x							
	U3.4			x	x							
	U4.1							x	x	x	x	x
	U4.2							x	x	x	x	x

Table 1: Traceability Matrix