|  |  |  |
| --- | --- | --- |
| **FA 3613** | | |
| Adversarial Multi-Agent Transfer Learning | | |
| **Amal Chulliyat Jose** | | |
|  | | |
| **System Requirements Specification** | | |
|  | Examiner: | Jun.-Prof. Dr.-Ing. Andrey Morozov |
|  | Supervisor: | Joachim Grimstad, M.Sc.Ing |
| Begin: 16.10.2023 | | End: 27.10.2023 |
|  | |  |

**Document Version Management**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Author | QA | Date | Status | Changes |
| 0.1 | ACJ |  | 09.10 | in progress | Creation |
|  |  |  |  |  |  |

# Table of Contents

0 Table of Contents 2

1 General Purpose 4

1.1 Mandatory Criteria 4

1.2 Optional Criteria 4

1.3 Demarcation Criteria 4

2 Operational Area 4

2.1 Areas of Application 4

2.2 User Group 4

3 Requirements to the Conception 4

4 Quality Requirements 4

5 Execution 5

6 Prototype 5

6.1 Environment 6

6.1.1 Software 6

6.1.2 Hardware 6

6.1.3 System Interfaces 6

6.2 Quality Requirements for the Prototype 6

6.3 Development Environment 7

6.3.1 Software 7

6.3.2 Hardware 7

7 Global Evaluation Methods 7

8 Additions 7

# General Purpose

Model-based systems engineering (MBSE) is a method used to model and analyze a system for its capabilities and limitations before its actual implementation in the real world. Often, the performance features of the system compete with each other to take control over its state and affect its safety. The Deep Learning-based Adversarial Multi-Agent Reinforcement Learning (Adv-MARL) can be used to decide on the optimum model in this scenario. Here, the Dynamic Fault Trees (DFT) are made into a zero-sum game and it is evolved over time by taking in the lessons learned from the previous versions.

## Mandatory Criteria

* The methodology must not compromise the safety of the systems involved.
* The methodology must be scalable to handle a large number of agents and complex systems.
* It should achieve a minimum level of performance improvement over an existing approach.

## Optional Criteria

* It should try to achieve the desired results with minimum computational resources.
* It should be compatible with different software environments.

## Demarcation Criteria

* The project must be completed within 6 months of its commencing.

# Operational Area

## Areas of Application

The primary area of application of this project is education. However, it may also be used in the current research work of my supervisor.

## User Group

Students and researchers of IAS.

# Requirements to the Conception

|  |  |
| --- | --- |
| /CR10/ | It must include a literature survey on topics like MBSE, DFT, Game theory, Adv-MARL, Transfer Learning |
| /CR20/  /CR21/  /CR22/ | Implementation to be done in Python  Must utilise the PettingZoo library for the implementation of the learning environment  The code must follow OOP concepts |

(User Requirements Specifications 0.1)

# Quality Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product Quality** | **very high** | **high** | **normal** | **not relevant** |
| **Theory** |  | **x** |  |  |
| Universality |  | x |  |  |
| Consistency |  | x |  |  |
| **Functionality** |  | **x** |  |  |
| Correctness |  | x |  |  |
| Applicability |  |  | x |  |
| **Safety** |  |  |  | **x** |
| **Usability** |  | **x** |  |  |
| Comprehensibility |  | x |  |  |
| Learnability |  | x |  |  |
| **Adaptability** |  |  | **x** |  |
| Analyzability |  |  | x |  |
| Modifiability |  |  | x |  |

# Execution

The thesis has to be executed according to the “IAS Process Model” (Model for Conceptional Projects).

The current state of the thesis and results have to be discussed with the tutor every two weeks.

The IAS guidelines have to be respected.

# Prototype

A Python implementation of the concept is intended.

## Environment

### Software

* Python
* TensorFlow or PyTorch for machine learning
* OpenAI Gym for simulation

### Hardware

Laptop computer

### System Interfaces

none

## Quality Requirements for the Prototype

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Product Quality** | **very high** | **high** | **normal** | **not relevant** |
| **Functionality** |  | **x** |  |  |
| Correctness |  | x |  |  |
| Security |  | x |  |  |
| Interoperability |  |  | x |  |
| **Reliability** |  | **x** |  |  |
| Maturity |  | x |  |  |
| Fault Tolerance |  | x |  |  |
| Recoverability |  | x |  |  |
| **Safety** |  |  |  | **x** |
| **Usability** |  | **x** |  |  |
| Comprehensibility |  | x |  |  |
| Learnability |  | x |  |  |
| Handling |  | x |  |  |
| **Efficiency** |  |  | **x** |  |
| Temporal Efficiency |  |  | x |  |
| Consumption Efficiency |  |  | x |  |
| **Adaptability** |  | **x** |  |  |
| Analyzability |  | x |  |  |
| Modifiability |  | x |  |  |
| **Portability** |  |  | **x** |  |

## Development Environment

### Software

* Python
* TensorFlow or PyTorch for machine learning
* OpenAI Gym for simulation

### Hardware

Laptop computer

# Global Evaluation Methods

* Use unit testing and integration testing in case of multiple components and verify that the Python script produces the intended results.
* Measure the runtime performance of the prototype by evaluating factors like processing time, memory utilization, etc.
* Ensure object-oriented programming to allow easy adaptations and modifications.

# Additions

None.