On Achieving Acceptable Levels of Safety Risk in a Reinforcement Learning Environment

Christian Davila
Washington University – McKelvey School of Engineering

Follow this and additional works at: https://openscholarship.wustl.edu/eng_etds

Recommended Citation

This Dissertation is brought to you for free and open access by the McKelvey School of Engineering at Washington University Open Scholarship. It has been accepted for inclusion in McKelvey School of Engineering Theses & Dissertations by an authorized administrator of Washington University Open Scholarship. For more information, please contact digital@wumail.wustl.edu.
On Achieving Acceptable Levels of Safety Risk in a Reinforcement Learning Environment
by
Christian Davila

A dissertation presented to
the McKelvey School of Engineering
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Science

August 2023
St. Louis, Missouri
# Table of Contents

List of Figures .......................................................................................................................... vii
List of Tables ............................................................................................................................. ix
Acknowledgments ................................................................................................................... x

Abstract ................................................................................................................................... xii

Chapter 1: Introduction ............................................................................................................. 1
  1.1 Background ..................................................................................................................... 1
  1.2 Motivation ...................................................................................................................... 1
  1.3 Objective ..................................................................................................................... 2

Chapter 2: Literature Review ................................................................................................... 3
  2.1 Overview of Machine Learning .................................................................................... 3
    2.1.1 Training Data ........................................................................................................ 3
    2.1.2 Training Process .................................................................................................... 4
  2.2 Different Types of Machine Learning ......................................................................... 4
    2.2.1 Supervised Learning ............................................................................................. 4
      2.2.1.1 Challenges in Supervised Learning .............................................................. 5
    2.2.2 Unsupervised Learning ......................................................................................... 6
      2.2.2.1 Challenges in Unsupervised Learning .......................................................... 6
    2.2.3 Reinforcement Learning ....................................................................................... 7
      2.2.3.1 Challenges in Reinforcement Learning ....................................................... 8
  2.3 System Safety ............................................................................................................... 10
    2.3.1 Review of Traditional Safety and Design Assurance Standards ...................... 11
      2.3.1.1 Relevant Reference Documents .................................................................... 11
        2.3.1.1.1 Road Vehicles – Functional Safety (ISO 26262) .................................. 11
        2.3.1.1.2 Safety of the Intended Functionality (SOTIF ISO/PAS 21448) .......... 11
        2.3.1.1.3 NASA Software Safety Guidebook (NASA-GB-8719.13) ............... 12
        2.3.1.1.4 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (ARP4761) .................. 12
2.3.1.5 Joint Software Systems Safety Engineering Handbook (JSSSEH Version 1.0) ................................................................................................................... 12
2.3.1.6 JSSSEH Implementation Guide (JS-SSA-IG Rev. B) ............................ 13
2.3.1.7 Department of Defense Standard for System Safety (MIL-STD-882E) 13
2.3.1.8 Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178) ................................................................. 13
2.3.1.9 Standard for Software Safety Plans (IEEE Std 1228-1994) ............. 14

2.3.2 Safety in Machine Learning ................................................................... 14

2.3.3.1 Gaps with Traditional Safety Standards as They Pertain to Machine Learning ................................................................. 16
2.3.3.2 Safe Reinforcement Learning Strategies .................................. 17
2.3.3.3 Safety Best Practices Applicable to Reinforcement Learning .... 18

Chapter 3: Methods ....................................................................................... 20

3.1 Overview of Methods ............................................................................. 20

3.1.1 Platform ............................................................................................... 23
3.1.2 Data ..................................................................................................... 24

3.2 Process ...................................................................................................... 24

3.2.1 System Definition and Safety Planning – System Development ........ 27

3.2.1.1 Guidelines for System Definition and Safety Planning .......... 28

3.2.2 System Requirements – System Development ............................... 32

3.2.2.1 Guidelines for System Requirements ........................................ 34

3.2.3 System/Subsystem Design – System Development ...................... 37

3.2.3.1 Guidelines for System/Subsystem Design ................................ 39

3.2.4 Conceptual Model Design – RL Model Development ................. 39

3.2.4.1 Guidelines for Conceptual Model Design ................................. 41

3.2.4.1.1 ODD Characterization .......................................................... 42

3.2.4.1.2 RL Requirements ................................................................. 43

3.2.4.1.3 RL Environment ................................................................... 44

3.2.5 Model Data Management – RL Model Development ................ 44

3.2.5.1 Guidelines for Model Data Management ................................ 47

3.2.6 Detailed Model Design – RL Model Development ........................ 49
4.1.2  System Requirements – System Development .................................................. 89
     4.1.2.1 High-Level Requirements ..................................................................... 89
     4.1.2.2 Process Requirements ......................................................................... 90
     4.1.2.3 Identification of Hazards and Failure Conditions ............................... 91
4.1.3  System/Subsystem Design – System Development ........................................ 93
     4.1.3.1 System/Subsystem Architecture ......................................................... 93
     4.1.3.2 Derived Low-Level Requirements ...................................................... 94
4.1.4  Conceptual Model Design – RL Model Development ................................... 98
     4.1.4.1 ODD Characterization ......................................................................... 98
     4.1.4.2 RL Model Requirements .................................................................... 100
     4.1.4.3 RL Environment ................................................................................ 104
4.1.5  Model Data Management – RL Model Development .................................. 105
     4.1.5.1 Training and Cross-Validation Datasets ............................................. 105
     4.1.5.2 Verification Dataset ............................................................................ 106
4.1.6  Detailed Model Design – RL Model Development ....................................... 106
     4.1.6.1 Model Logical Architecture ................................................................. 106
     4.1.6.2 Objective Function ............................................................................. 107
     4.1.6.3 Learning Algorithm ........................................................................... 108
     4.1.6.4 Policy Type ....................................................................................... 108
4.1.7  Model Coding/Building – RL Model Development ...................................... 108
     4.1.7.1 Model Coding .................................................................................... 108
         4.1.7.1.1 Environment ............................................................................... 108
         4.1.7.1.2 Agents ....................................................................................... 110
         4.1.7.1.3 Training ..................................................................................... 112
         4.1.7.1.4 Testing ....................................................................................... 113
     4.1.7.2 Model Building .................................................................................. 114
         4.1.7.2.2 Agent Vehicle and Environment ................................................. 115
         4.1.7.2.3 Signal Processing for Longitudinal Control ................................. 119
         4.1.7.2.4 Signal Processing for Lateral Control ......................................... 121
4.1.8  Model Training and Evaluation – RL Model Development ......................... 122
     4.1.8.1 Optimized Policy .............................................................................. 123
4.1.9   Model Verification – RL Model Development.................................................. 124
         4.1.9.1 RL Model Requirements on Data Verification.......................................... 124
         4.1.9.2 RL Model Requirements on Design Verification ...................................... 128
4.1.10  System/Subsystem Implementation – System Development.......................... 153
4.1.11  System Verification & Risk Assessment – System Development....................... 153

Chapter 5: Discussion and Conclusions........................................................................ 155
Chapter 6: Future Work ................................................................................................. 159
References..................................................................................................................... 161
List of Figures

Figure 3.1 Reinforcement learning model development ............................................................ 23
Figure 3.2 System safety assurance and integrity process .......................................................... 31
Figure 4.1 Safety control structure path-following control ......................................................... 95
Figure 4.2 Path-following control using reinforcement learning ................................................ 114
Figure 4.3 Agent vehicle and environment .............................................................................. 115
Figure 4.4 Agent vehicle ........................................................................................................... 116
Figure 4.5 Agent vehicle 3DOF model ..................................................................................... 116
Figure 4.6 Agent sensor ............................................................................................................. 117
Figure 4.7 Lead vehicle ............................................................................................................. 117
Figure 4.8 Right vehicle ............................................................................................................ 118
Figure 4.9 Left vehicle .............................................................................................................. 118
Figure 4.10 Signal processing for longitudinal control ............................................................... 119
Figure 4.11 Velocity error ........................................................................................................ 119
Figure 4.12 Safe distance ......................................................................................................... 120
Figure 4.13 Simulation termination longitudinal control ............................................................ 120
Figure 4.14 Reward function .................................................................................................. 120
Figure 4.15 Signal processing for lateral control ..................................................................... 121
Figure 4.16 Reward function .................................................................................................. 121
Figure 4.17 Lateral control observations .................................................................................. 122
Figure 4.18 Reinforcement learning episode manager .............................................................. 123
Figure 4.19 Training simulation ............................................................................................... 125
Figure 4.20 Simulation termination longitudinal control ........................................................... 127
Figure 4.21 Signal processing for lateral control .................................................................... 128
Figure 4.22 Training simulation ............................................................................................... 129
Figure 4.23 Sample of not merging – safe and relative distance (DE4) ..................................... 132
Figure 4.24 Sample of not merging – longitudinal velocity (DE4) ......................................... 133
Figure 4.25 Sample of not merging – acceleration and steering (DE4) .................................... 133
Figure 4.26 Sample of not merging – yaw error (DE4) ............................................................ 134
Figure 4.27 Sample of not merging – lateral error (DE4) ........................................................ 134
Figure 4.28 Sample of not merging – lane position (DE4) ................................................................. 135
Figure 4.29 Sample of merging left – safe and relative distance (DE4) ............................................... 135
Figure 4.30 Sample of merging left – longitudinal velocity (DE4) ...................................................... 136
Figure 4.31 Sample of merging left – acceleration and steering (DE4) ............................................... 136
Figure 4.32 Sample of merging left – yaw error (DE4) ........................................................................ 137
Figure 4.33 Sample of merging left – lateral error (DE4) .................................................................. 137
Figure 4.34 Sample of merging left – lane position (DE4) ................................................................. 138
Figure 4.35 Sample of endurance testing – relative distance (DE4) ..................................................... 138
Figure 4.36 Sample of endurance testing – safe distance (DE4) .......................................................... 139
Figure 4.37 Sample of outlier case – safe and relative distance (DE5) .................................................. 141
Figure 4.38 Sample of outlier case – longitudinal velocity (DE5) ....................................................... 141
Figure 4.39 Sample of outlier case – acceleration and steering (DE5) ................................................ 142
Figure 4.40 Sample of outlier case – yaw error (DE5) ....................................................................... 142
Figure 4.41 Sample of outlier case – lateral error (DE5) ................................................................... 143
Figure 4.42 Sample of outlier case – lane position (DE5) ................................................................. 143
Figure 4.43 Sample of corner case – safe and relative distance (DE5) .................................................. 144
Figure 4.44 Sample of corner case – longitudinal velocity (DE5) ....................................................... 144
Figure 4.45 Sample of corner case – acceleration and steering (DE5) ................................................ 145
Figure 4.46 Sample of corner case – yaw error (DE5) ....................................................................... 145
Figure 4.47 Sample of corner case – lateral error (DE5) ................................................................... 146
Figure 4.48 Sample of corner case – lane position (DE5) ................................................................. 146
Figure 4.49 Sample of endurance testing – relative distance (DE5) ..................................................... 147
Figure 4.50 Sample of endurance testing – safe distance (DE5) ........................................................ 147
Figure 4.51 Sample of reliability testing – relative distance (DE6/DE7/DE8) ....................................... 149
Figure 4.52 Sample of reliability testing – safe distance (DE6/DE7/DE8) ......................................... 149
List of Tables

Table 3.1 Safety risk determination ................................................................. 26
Table 3.2 Reinforcement learning safety criticality matrix ........................................ 32
Table 3.3 Level of rigor activities for conceptual model design ................................... 41
Table 3.4 Level of rigor activities for model data management .................................... 46
Table 3.5 Level of rigor activities for detailed model design ..................................... 52
Table 3.6 Level of rigor activities for model coding/building ..................................... 62
Table 3.7 Level of rigor activities for model training and evaluation ............................ 64
Table 3.8 Level of rigor activities for model verification .......................................... 66
Table 3.9 Level of rigor verification activities for model development ......................... 67
Table 4.1 Severity categories ................................................................................. 86
Table 4.2 Control categories ................................................................................... 87
Table 4.3 Safety case reinforcement learning safety criticality matrix ............................. 88
Table 4.4 Safety case reinforcement learning safety criticality matrix results .................. 91
Table 4.5 Safety case level of rigor tasks results .................................................... 91
Table 4.6 Undesired control action ......................................................................... 96
Table 4.7 Model requirements ................................................................................. 102
Table 4.8 Model requirements safety tagging .......................................................... 104
Table 4.9 Stopping criteria ..................................................................................... 123
Table 4.10 Hyperparameter selection for DDPG - critic .......................................... 124
Table 4.11 Hyperparameter selection for DDPG - actor ........................................... 124
Table 4.12 Hyperparameter selection for DQN - critic ............................................. 124
Table 4.13 DE4 testing parameters ......................................................................... 131
Table 4.14 DE4/DE9/DE10/DE11/DE12 testing results ............................................ 132
Table 4.15 DE5 testing parameters ......................................................................... 140
Table 4.16 DE5/DE9/DE10/DE11/DE12 testing results ............................................ 140
Table 4.17 DE6 testing parameters and results .......................................................... 148
Table 4.18 DE7 testing parameters and results .......................................................... 150
Table 4.19 DE8 testing parameters and results .......................................................... 151
Table 4.20 Safety case safety risk determination ..................................................... 152
Acknowledgments

First, I am grateful to my advisor Prof. Ramesh Agarwal for his encouragement and advice for more than seven years, guiding me through challenging decisions to arrive at this stage. He has provided mentorship for my career and academic aspirations. Thank you for your unending patience and willingness to stick with me through a cross-country move and career transitions.

I want to thank the Washington University of St. Louis faculty committee members for agreeing to serve on the committee: Prof. Jakiela, Prof. Li, and Prof. Peters. I would also like to express my gratitude to Prof. Juba. His invaluable guidance and technical expertise have consistently pushed me to explore new perspectives and refine my ideas.

I also want to acknowledge the mentors that helped me develop as a professional: Mr. Plawecki of Northrop Grumman and Mr. Schmedake of Boeing. Thank you for taking me under your wing, for your countless hours of support, and for serving on this committee.

I am deeply appreciative of my amazing family, whose consistent affection, motivation, and unwavering backing have served as the foundation of my educational voyage. To my dear dad, Fredy Dávila, your guidance and wisdom have been a guiding light, showing me the importance of perseverance and hard work. To my beloved mom, Blanca Cantoral de Dávila, your nurturing care and boundless faith have been a constant inspiration. Your sacrifices have shaped me into the person I am today. To my incredible brothers, David, Paolo, and Ricardo Dávila, your camaraderie and encouragement have been invaluable.
I am also profoundly grateful to my cherished fiancée, Kelsey Douglas, whose unwavering support and boundless encouragement played an instrumental role in the completion of this thesis. Your dedication and patience were indispensable throughout this journey.

As I stand on the threshold of this academic achievement, I am acutely aware that it would not have been possible without your collective influence.

Christian Davila

Washington University in St. Louis

August 2023
ABSTRACT OF THE DISSERTATION

On Achieving Acceptable Levels of Safety Risk in a Reinforcement Learning Environment

by

Christian Davila

Doctor of Science in Mechanical Engineering

Washington University in St. Louis, 2023

Professor Ramesh Agarwal, Chair

It is well-known that model correctness does not ensure the safe operation of systems that perform safety-critical functions. However, to ensure that potential hazardous events cannot occur, the model must be verified against its safety requirements, many derived during a systematic safety analysis. Typically, this process utilizes a complementary approach of formal verification and testing during the development process of complex systems. As these complex systems expand with the use of Reinforcement Learning (RL), safety considerations must be taken in the development of this technology.

Although RL is becoming widely used across different industries, RL does pose some safety challenges. Few guidelines are available for developing a model for systems that demand a high level of safety confidence in an RL environment. The goal that I put forth is to increase safety confidence in systems that utilize RL by implementing an approach that incorporates industry-vetted safety guidelines and novel practices for developing safe RL models. The objective is to create a method that allows for the assessment of the safety risks associated with RL-based systems and determine their acceptability, providing a metric to measure this acceptability. The aim is to develop specific objectives and Level of Rigor (LOR) activities, which create guidelines for developers or project teams to increase the safety confidence in the model. These guidelines offer
recommendations for enhancing the development of RL subsystems, facilitating the identification, evaluation, and mitigation of safety risks. It is essential to emphasize that these guidelines are designed to complement, rather than replace, the LOR activities established by traditional system safety standards to enhance the overall system’s safety confidence.

The presented guidelines are demonstrated in a safety case to address both known unsafe and unknown unsafe hazards. For known unsafe scenarios, a deterministic analysis is utilized, while a rigorous development process is implemented to ensure safety in unknown unsafe scenarios, ensuring assurance and integrity in safety-critical functions. This approach involves a hazard analysis process to identify measures for risk mitigation (functional coverage) and LOR activities to ensure the high-quality model development (development coverage). The goal is to create guidelines that are comprehensive, contextually relevant, and easily understandable in addition to establishing a robust, accurate, reliable, and generalizable model.
Chapter 1: Introduction

1.1 Background
Machine learning (ML) is a subset of artificial intelligence that involves the use of datasets to train computer algorithms, allowing performance improvement on specific tasks without explicit programming. Today, ML is becoming increasingly important to the development of technologies and products in the automotive industry, aerospace industry, and defense, among others. While ML can be a useful tool, it may also pose safety risks as it takes over safety-critical functions. Historically, systems that implement safety-critical functions must operate deterministically to achieve highly predictable behavior. The problem with ML algorithms, especially reinforcement learning (RL), is that RL policies may not be perceived as deterministic due to the stochasticity in the environment which introduces safety risk. This, in turn, makes interpreting the agent’s behavior difficult and makes proving design correctness with testing under reasonable time constraints challenging.

1.2 Motivation
With the expansion of RL into functions that pose safety-critical implications, safety considerations must be taken in the development of this technology. RL is becoming widely used across different industries, and there is an increased demand for the answers to the safety challenges posed by RL. Few guidelines are available for developing a model for systems that require a high level of safety confidence in an RL environment. Because of the lack of guidance on system safety techniques for developers, this process will be a timely addition to the field of study.
1.3 Objective

The objective set forth is to increase safety confidence in systems utilizing RL by employing an approach that integrates industry-vetted safety guidelines and unique practices for developing safe RL models. The aim is to develop a method that enables the assessment of safety risks in RL-based systems and evaluates their acceptability, using a metric. Specific objectives and Level of Rigor (LOR) activities will be formulated to create guidelines for developers or project teams, facilitating the increase of safety confidence in the model. These guidelines will provide recommendations for the development of RL subsystems, aiding in the identification, evaluation, and mitigation of safety risks. Importantly, these guidelines are intended to complement, rather than replace, the LOR activities prescribed by traditional system safety standards, thus strengthening the overall safety confidence of the system.

The proposed guidelines will be demonstrated in a safety case to address both known and unknown unsafe hazards. For known unsafe scenarios, deterministic analysis will be utilized, while a rigorous development process will be employed to ensure safety in unknown unsafe scenarios, ensuring assurance and integrity in safety-critical functions. The approach will include a hazard analysis process to identify risk mitigation measures (functional coverage) and LOR activities to ensure high-quality model development (development coverage). The end product will be comprehensive, contextually relevant, and easily understandable guidelines, ultimately leading to the establishment of a robust, accurate, reliable, and generalizable RL model.
Chapter 2: Literature Review

2.1 Overview of Machine Learning

There are different definitions of ML, however, a modern definition is given by Tom Mitchel in his book *Machine Learning*, “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E” [1]. Although the application of ML does not necessarily constitute artificial intelligence within a system, it allows the system to execute tasking such as pattern detection, behavior generation based on patterns, decision-making based on rewards and penalties, and behavior modification based on interaction with the environment [2]. Therefore, ML uses data patterns to automatically improve without explicit programming. This function works by using a set of data or sample data, known as “training data,” which is either provided or collected, to make predictions for new data inputs, thus, learning from experience.

This learning may either occur during development or after deployment [3]. In some cases, the learner receives training before and during deployment which provides a framework or starting point and the opportunity for continued learning during deployment. The process of learning during development is referred to as *offline* learning, and the process of learning during deployment is referred to as *online* learning [3].

2.1.1 Training Data

For some ML algorithms, the initial data that helps develop an ML model is referred to as training data, helping the model to establish its algorithm’s properties. The model’s continued development is highly impacted by the quality of its training.
\section*{2.1.2 Training Process}
In ML, multiple iterations are required for training. The available data is divided for training purposes and testing purposes. First, the training dataset is used for training the model. Then the testing dataset is utilized to measure the model’s accuracy. This is done by having the model predict testing dataset values and comparing those to the values that are known to be true. A best practice is to incorporate a cross-validation dataset as well, cross-validating during the training process to prevent overfitting \cite{4}. Overfitting occurs “when a learning algorithm fits the training dataset so well that noise and the peculiarities of the training data are memorized” \cite{5}. Therefore, the goal during the training process is to develop an ML model capable to generalize based on the training data it has seen.

\section*{2.2 Different Types of Machine Learning}
ML utilizes different approaches to teach computers how to make predictions for new data inputs. These approaches consist of supervised learning, unsupervised learning, and RL. In supervised learning, a function is inferred from a labeled dataset. In unsupervised learning, there are no labeled datasets; the algorithm must draw inferences from data patterns. Seemingly, unsupervised learning appears to be an effective way to determine data correlations without previously knowing trends in the data. For this reason, there has been an increased amount of research on the subject and many researchers have turned their focus to RL. This algorithm is unique from supervised and unsupervised learning, in which the feedback is received from trial and error, allowing the machine to learn from experience using a system of rewards and penalties.

\subsection*{2.2.1 Supervised Learning}
Supervised learning, one of the earliest forms of ML, builds on statistical learning theory. In supervised learning, labeled datasets serve as training data, composed of inputs and outputs. The
algorithm is trained to correlate anticipated outputs to specific inputs. The goal of supervised learning is to utilize new data to predict outputs.

Different types of supervised learning algorithms include decision trees, multi-class classification, linear regression, and logistic regression, which are commonly used to solve problems like regression and classification according to [6].

2.2.1.1 Challenges in Supervised Learning
In supervised learning, Faria indicates the probability distribution \( P(X, Y) \) derived from the statistical learning theory is unknown, with only a finite set of samples. According to Faria, “…the expected loss on the training set is called \( \text{empirical risk of } h: R_{\text{emp}}(h) = \frac{1}{m} \sum_{i=1}^{m} L(x^{(i)}, h(x^{(i)}), y^{(i)}) \)” for hypothesis \( h \) and loss function \( L \). With ML algorithms, the performance of training data can be optimized, ideally to reduce the safety risk in real-world practice. However, if this assumption does not hold true, potential concerns arise [7]. These potential concerns include training data and risk optimization.

Two main factors to consider in collecting training data include the size of the dataset and the quality of the dataset [8]. The quantity of samples required for appropriate generalization is still debated [7]. However, “an order of magnitude more examples than trainable parameters” is a general guideline [8]. In terms of quality, it is also cautioned that “corrupt or badly curated data” may not provide full representation [7]. It is recommended to take label error rates, feature noise, and data filtering into account when curating datasets [8]. This type of data preparation and pre-processing is another challenge posed by Supervised Learning. In addition, it is also noted that feature representation is another quality concern, taking into account numeric value normalization, outliers, and how data is shown to the model [8].
Another concern category includes risk optimization, which may result in overfitting. A model lacking generalization may be cause for concern. To overcome this challenge, Faria states that researchers aim to minimize some regularized risk [7]. This is a technique that discourages learning to avoid overfitting.

### 2.2.2 Unsupervised Learning

In supervised learning, labeled datasets are used for training, while in unsupervised learning, unlabeled datasets are given to the algorithm to find hidden patterns or features. By clustering and analyzing this data, patterns that are not easily understood by humans can be utilized for data segmentation, image or speech recognition, and other applications. Common unsupervised learning approaches include clustering, association, and dimensionality reduction [9].

#### 2.2.2.1 Challenges in Unsupervised Learning

The two main challenges of supervised learning are the lack of transparency on how the data is grouped, seen in clustering applications and the lack of measurement of accuracy of results. In addition, other challenges include the “computational complexity due to high volume of training data” and the “human intervention to validate output variables” [10].

The lack of transparency on the data clustering concerns users, however, some researchers have posed the use of semi-supervised learning to overcome this challenge. In semi-supervised learning, labeled and unlabeled datasets are used. This allows for significant features to be identified and isolated while providing accurate and trustworthy results, which draws from the advantages of supervised learning [9].

Many researchers have circumvented the lack of measurement of accuracy by developing different evaluating techniques based on the approach. For example, the clustering approach draws from
evaluation techniques that may vary depending on factors such as identifying trends in the clustering data, the availability of external information, and the number of clusters to name a few.

### 2.2.3 Reinforcement Learning

Unlike supervised and unsupervised learning, the RL algorithms are unique in that agents, the entities that take decisions, actively gather new information about their environment. In terms of RL, an environment is a world in which the agent interacts, in a process known as exploration. As agents gather information, their actions are rewarded or penalized. Rewards for the agent include environmental feedback in the form of numerical values. The goal of the agent is to maximize the reward function through trial and error, in search of the optimum policy. The policy is a mapping of an agent’s state or current situation to actions. The optimum policy may be achieved through a future or delayed reward known as value. The agent acts in an attempt to attain the highest possible reward in a process known as exploitation.

The mathematical framework of this RL process may be described by the Markov Decision Process (MDP). According to “Assured Reinforcement Learning with Formally Verified Abstract Policies,” this process offers a map for depicting decision-making in situations with risk in that actions are taken by the agent in terms of probability. This process may be modeled mathematically by the transition function and the reward function with the parameters “<S,A,T,R>,” where S is a finite set of states; A is a finite set of actions; T: S × A × S → [0,1] is a state transition function … and R: S × A × S → ℝ is the reward function” [11]. In an MDP, the point of interest is to find a deterministic policy π: S → A to maximize the reward. According to the 2019 MATLAB E-book *Reinforcement Learning with MATLAB*, the Bellman optimality equation provides for “necessary and sufficient condition for an optimum” [12]. The Bellman optimality equation is defined as $V^*(s) = \max \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V^*(s'))$ [13].
2.2.3.1 Challenges in Reinforcement Learning

In RL, at a high level, some of the challenges perceived in the industry include high computational expense and approximation, exploration vs. exploitation dilemma, unspecified reward functions, and tasks that may be partially observable.

High-dimensional continuous state and action spaces make RL computationally expensive. In addition, these high dimensionality state and action spaces do not permit the precise computation of the value function. A common solution is to use artificial networks for approximating the Bellman optimality equation [7].

According to Coggan, “A common problem in reinforcement learning is finding a balance between exploration (attempting to discover new features about the world by selecting sub-optimal action) and exploitation (using what we already know about the world to get the best results we know of).” In essence, exploration requires time and chance which may pose a significant to many real-world problems. Coggan writes that there are many basic strategies to overcome this challenge such as state-action value updating strategies and action selection strategies. The state-action value updating strategies work with the principle of storing a value for each action from each state. Action selection strategies center on greedy selection in which the highest state action is chosen [14].

“Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis” states that the objective is to optimize a global reward function. However, this often proves challenging when the developers or the agent do not clearly understand the optimization goal. Additionally, it is often the case that in the optimization process, other areas of maintenance or
improvement are also found. For this reason, careful examination should be employed to understand and divide the features of the reward function that need to be optimized [15].

The fourth main challenge commonly found in the literature deals with partial observability. Typically, except for specific simple scenarios, it is assumed the agent does not have complete knowledge of the environment, but rather an incomplete perspective based on observations that the agent has collected during the exploration process. This partially observable problem is known as the Partially Observable Markov Decision Process (POMDP). There are two common methods for approaching this challenge. One approach is to integrate history into the agent’s observations, and the other approach uses agent networks to “enabling them to track and recover hidden state” [15].

2.2.4 Deep Learning
Common to supervised learning, unsupervised learning, and RL, deep learning is a type of ML that mimics how the brain works by utilizing artificial neural networks (ANN). “DL is a collection of techniques and methods for using neural networks to solve ML tasks, either Supervised Learning, Unsupervised Learning, or Reinforcement Learning” [16]. The most essential types of neural networks for deep learning include convolution neural networks (CNN) and recurrent neural networks (RNN). CNN and RNN are both types of ANN but are used to solve different kinds of problems, with CNN utilized for image processing due to its capability to extract spatial features and RNN utilized for problems related to time series data due to its ability to use temporal information [17].

2.2.4.1 Challenges in Deep Learning
With an exceptional ability to generalize correlations between input and output data, deep learning has proven to be useful in many applications. However, deep learning also poses some challenges.
One challenge of deep learning perceived black-box structure; while input and output data are visible, the structure of the internal network is not easily understood. According to Angelov and Sperduti, other challenges posed by deep learning are the constant structural evolution and computational efficiency [18].

To overcome the structural challenges of deep learning, researchers rely on different interpretability techniques based on the features. According to Montavon et al., some of the common ones are rooted in gradient-based methods such as sensitivity analysis and simple Taylor decomposition. Other popular methods are backward propagation techniques that provide practical benefits because of superior scaling to complex Deep Neural Networks (DNN) models [19].

2.3 System Safety
As defined by MIL-STD-882E, the standard utilized by the Department of Defense, safety can be defined as “Freedom from conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment” [20]. One may conclude from this definition that a safe system is one in which scenarios with unacceptable consequences such as death have an acceptably low probability of occurrence.

Therefore, system safety, a branch of systems engineering, aims to achieve acceptable safety risk of mishap within the environment throughout the lifetime of the system. System safety accomplishes this by systematically identifying potential hazards, assessing the hazard risk, and utilizing strategies to eliminate hazards. If a hazard cannot be eliminated, then the system safety analyst aims to identify mitigating measures to bring the hazard to an acceptably low level of occurrence. All of this is done with careful consideration for cost, certification, and performance by utilizing safety best practices.
2.3.1 Review of Traditional Safety and Design Assurance Standards
It is of paramount importance in many industries to ensure safety in complex systems to protect human lives, the environment, and the system itself. To achieve this, various system safety standards have been established to provide comprehensive guidelines and best practices for the design, development, and operation of safety-critical systems. By adhering to rigorous system safety standards, industries can instill confidence in their products and systems while safeguarding against potential hazards and risks, ultimately ensuring safety in complex systems.

2.3.1.1 Relevant Reference Documents
2.3.1.1.1 Road Vehicles – Functional Safety (ISO 26262)
One of the major safety standards in the automotive industry today is ISO 26262. This standard is utilized by engineers to eliminate hazards due to Electric and Electronic (E/E) functional failures. If hazards cannot be eliminated, the standard guides engineers in bringing these hazards to an acceptable level of safety risk. ISO 26262 prescribes performing a Hazard and Risk Assessment (HARA) to determine the initial level of safety risk in a system. This process then generates safety requirements to lower a system’s initial level of safety risk to an acceptable level. These safety requirements are then allocated to either hardware or software during the development phase. The systems engineering V model is used during this systematic process to validate and verify safety requirements, providing confidence that the safety requirements are met by the architecture and tested. These requirements and testing procedures are flowed down to the unit level [21].

2.3.1.1.2 Safety of the Intended Functionality (SOTIF ISO/PAS 21448)
As ISO 26262 did not include standards to address safety in autonomous vehicles, a supplemental safety standard, ISO/PAS 21448, was created. ISO/PAS 21448, also known as Safety of Intended Functionality (SOTIF), enhances ISO 26262 by acknowledging software constraints. This standard accounts for ML and its safety concerns, such as the associated epistemic uncertainty.
Additionally, this standard addresses the human-machine interface and its shortcomings in situational awareness and its misuse [22].

2.3.1.1.3 NASA Software Safety Guidebook (NASA-GB-8719.13)
This guidebook is a comprehensive document developed by the National Aeronautics and Space Administration (NASA) to provide guidelines and best practices for ensuring the safety of software used in aerospace systems and missions. The guidebook serves as a crucial reference for software developers and other personnel involved in NASA projects, outlining a systematic approach to identify, analyze, and mitigate potential software-related hazards. It covers various aspects of software safety and highlights the importance of rigorous software testing, verification, and validation techniques to achieve a high level of safety confidence in safety-critical systems [23].

2.3.1.1.4 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (ARP4761)
This document is widely recognized in the aerospace industry, developed by Society of Automotive Engineers (SAE) International, providing comprehensive guidelines and methodologies for conducting safety assessments of civil airborne systems and equipment. The document outlines a systematic approach to identify, analyze, and mitigate potential safety hazards and risks associated with safety-critical systems. ARP4761 is a crucial reference for developers and other personnel involved in the design, development, and certification of aviation systems, supporting the enhancement of safety measures and ensuring the airworthiness and reliability of civil airborne equipment [24].

2.3.1.1.5 Joint Software Systems Safety Engineering Handbook (JSSSEH Version 1.0)
The JSSSEH is a product of a collective effort between the U.S. Air Force, Army, Department of the Navy, and Coast Guard Safety Centers, with cooperation from the Federal Aviation Administration (FAA), NASA, academia, and defense industry contractors. This handbook
provides engineering and management guidelines to attain an appropriate level of assurance that the software will perform within the system context with an acceptable level of safety risk. In addition, the handbook addresses technical aspects of software function and design to aid with understanding software safety [25].

2.3.1.1.6 JSSSEH Implementation Guide (JS-SSA-IG Rev. B)
This document offers implementation guidance for the software system safety program requirements outlined in MIL-STD-882E and aligns with the details provided in the JSSSEH. The guide aims to support software system safety initiatives by providing developers and other personnel with comprehensive and direct guidance for ensuring safety and compliance within software development and engineering processes [26].

2.3.1.1.7 Department of Defense Standard for System Safety (MIL-STD-882E)
The Department of Defense (DoD) safety standard practice is MIL-STD-882. This safety standard identifies a systematic approach to eliminate hazards, when possible, and reducing safety risk where those hazards cannot be eliminated. The standard covers hazards associated with both hardware and software. This standard prescribes software safety considerations which include understanding and rationale behind the different Level of Rigor (LOR) tasks that are implemented as part of the software safety process [20].

2.3.1.1.8 Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178)
Widely recognized in the aviation industry, this document is developed jointly by the Radio Technical Commission for Aeronautics (RTCA) and the European Organization for Civil Aviation Equipment (EUROCAE). This document outlines the guidelines and objectives for the certification of airborne software used in safety-critical systems, defining the processes and requirements for software development, verification, validation, and integration to ensure
compliance with safety standards. DO-178 plays a crucial role in the certification and airworthiness of software used in aircraft, contributing to the overall safety and integrity of the aviation industry [27].

2.3.1.9 Standard for Software Safety Plans (IEEE Std 1228-1994)
Published by the Institute of Electrical and Electronics Engineers (IEEE), this document provides comprehensive guidelines and requirements for developing software safety plans in safety-critical systems and applications. The standard outlines the necessary methodologies for identifying, assessing, and mitigating potential safety hazards associated with software. It emphasizes the importance of integrating safety measures into the entire software development lifecycle and serves as a valuable reference for software developers and other personnel to create robust and reliable software safety plans to ensure the safe operation of the safety-critical system [27].

2.3.2 Safety in Machine Learning
As this technology has developed, its range of applicability and utility have also grown, leading to the increasing impact of ML on many industries. For example, ML is regularly used to estimate commute times and classify email, however, these functions are not classified as safety-critical.

Safety-critical is a term used to describe a function in which the loss of functionality or malfunction may result in a catastrophic event (such as death or loss of system) or as determined by the “immediate human costs of outcomes exceed[ed] [by] some severity threshold to be harmful” [29]. This loss of functionality or malfunction may be further categorized depending on when in the process it occurred. It is also worth mentioning that society defines the safety risk, in terms of severity and likelihood, in a safety-critical application. This is one of the reasons that “Employing ML in safety-critical systems that possess the potential to endanger human life or the environment is quite challenging” [3].
Risk minimization in ML is a key challenge to tackle. Reducing the likelihood of hazards does not address the concern associated with the epistemic uncertainty related to safety. Some factors of epistemic uncertainty relate to the training data. First, if it is wrongly assumed that the dataset accurately represents the likelihood distribution, there is potential for mishaps. Also, it may be the case that the training dataset only represents a small part of the distribution. Lastly, it should be noted that only a limited amount of data is available for training and may not accurately represent the true safety risk. This epistemic uncertainty may pose significant safety implications [29].

Another challenge posed by ML in safety-critical applications is the perceived black-box structure. Even when the developer identifies the right goal, undesirable behavior may occur. “Concrete Problems in AI Safety” outlines five specific failure modes. These failure modes can be illustrated through an example of an autonomous vehicle.

- **Avoiding Negative Side Effects**: The objective function may have been incorrectly defined. When this function is maximized, unintended results may occur. For example, an autonomous vehicle may disturb the environment by running over people while pursuing its goal of driving from point A to point B in the minimum amount of time possible.

- **Avoiding Reward Hacking**: The system solves problems by corrupting the objective function. In the autonomous vehicle example, the vehicle may ignore the speed limit constraint to drive from point A to point B in the minimum amount of time possible.

- **Scalable Oversight**: Limited training data leads to poor extrapolations. In an autonomous vehicle, this could be seen when a rare object is on the road. The vehicle needs to decide on whether to avoid the object at all costs even at the risk
of hitting another object while doing so or to run over the object. If an opossum was on the road, the autonomous vehicle should opt to run over it rather than potentially hitting a larger object while swerving. If instead of an opossum, the object was a human toddler, one would hope the autonomous vehicle would choose to swerve to avoid the child, even at the risk of hitting a larger object.

- Safe Exploration: Exploratory actions may lead to consequences that exceed the return value of the learning. For an autonomous vehicle, this could include experimenting with driving through molten lava. The learning the vehicle would derive is not worth the damages the exploration would cost.

- Robustness to Distributional Shift: When given input data that greatly differs from the training data, the ML system may make poor decisions. For example, an autonomous vehicle with training data on driving in snowy conditions may drive too quickly when operating in icy conditions.

Some of these failure modes may only be applicable in the context of the RL, while other failure modes become more complex in this environment. With the expansion of ML algorithms into functions that pose safety-critical implications, safety considerations must be taken in the development of this technology [30].

2.3.3.1 Gaps with Traditional Safety Standards as They Pertain to Machine Learning
Recent studies regarding automotive software safety assessed automotive standards, finding that about, “40% of software safety methods do not apply to ML models” [31]. Researchers have identified safety gaps that impact the development of ML models in the automotive industry. Mohseni et al. divide the gaps into categories such as design specification, implementation
transparency, testing and verification, performance and robustness, and run-time monitoring function [32].

In addition to these safety gaps, the safety standards lack a clear process for determining the approach to assessing the risk for a system that employs ML. Typically, different risk determination approaches establish confidence using a collection of assurance activities, therefore, there is a specific need to identify ML-specific assurance activities.

In sum, industries lack safety standards that address the safety gaps and challenges of ML. This need is even more dire in the realm of RL because of its capability of learning from trial and error. With the expansion of RL into functions that pose safety-critical implications, safety considerations must be taken in the development of this technology.

2.3.3.2 Safe Reinforcement Learning Strategies

Many scholars are attempting to identify strategies to develop an inherently safe design in an RL environment and ways to assess it. However, as the scientific community is far from reaching an inherently safe design in an RL environment with traditional methods, it is vital to explore safe reinforcement strategies to lower the safety risk. This is primarily because RL algorithms “discover policies that maximize reward, but do not necessarily guarantee safety during learning or execution phases” [33]. For this reason, safe RL has been a topic of interest for researchers. Safe RL is defined as “the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment process” [34].

According to García and Fernández, there are two main categories of safe RL. One of the main categories is based on the modification of the exploration process via external knowledge. This
external knowledge takes place in the form of providing initial knowledge, deriving a policy from
demonstrations, or via teacher advice. The other main category is via modification of the
optimization criteria. Modification of the optimization criteria takes place in the form of worst-
case criterion, risk-sensitive criterion, or constrained criterion, among others [34].

2.3.3.3 Safety Best Practices Applicable to Reinforcement Learning
As the above-mentioned safe reinforcement strategies may not be sufficient to achieve a high level
of safety confidence in an RL environment, other safety best practices should be considered.
Varshney provides recommendations on how to increase safety confidence in a system. Although
the ideal scenario is inherently safe design, Varshney recognizes that other strategies such as safe
fail, safety reserves, and procedural safeguards [35].

Safe fail, a best practice in safety, is a term used when a system fails when operating but remains
in a safe state. In RL, safe fail occurs when the model indicates a lack of high confidence in the
prediction, therefore, it does not provide such prediction to the user, allowing for user intervention.
To detect the lack of high confidence levels in the outputs of algorithms, some researchers
recommend the use of monitoring function techniques. These monitoring function techniques
could take the form of uncertainty estimation, in-distribution error detectors, and out-of-
distribution error detectors [32].

Another best practice, safety reserves, is also known as safety margins. Some researchers describe
safety reserves as the “difference between model’s performance on the training set and operational
performance open-world.” To address this challenge, the use of model robustness techniques is
recommended. These model robustness techniques could take the form of robustness to domain
shift and robustness to corruptions and perturbations [32].
Finally, Varshney also recommends the use of procedural safeguards [35]. These safety best practice procedures are outside the main operation of the system. Examples include warnings, cautions and advisories, training with published technical orders, and audits. The goal of this technique is to create awareness, proper training, and a universal design that would result in accidental system misuse.

Safety is becoming increasingly important in the development of RL algorithms. In a complex system, there are multiple levels of interfaces; for this reason, it is vital to integrate safety at every level. It is in this process that system safety is challenged with how to best implement the functional coverage and the development coverage.
Chapter 3: Methods

3.1 Overview of Methods
RL poses difficulty in measuring safety risk due to its inherent complexity. The complexity is influenced by factors such as the size of the state and action spaces, the complexity of the reward function, and the presence of uncertainty when interacting in a stochastic environment. This, in turn, makes interpreting the agent’s behavior difficult and makes proving design correctness with testing under reasonable time constraints challenging. The safety risk that RL poses is similar to that of complex systems that employ software. According to the NASA Software Safety Guidebook, “If the inherent reliability of software cannot be accurately measured or predicted, and most software designs cannot be exhaustively tested, the level of effort required to meet safety goals must be determined using other characteristics of the system” [23]

Because of complex systems’ safety challenges, such as RL, Ericson suggests in “Hazard Analysis Techniques for System Safety, Second Edition” the most practical way to assure safety in a complex system is by applying a systematic approach consisting of development coverage and functional coverage [36]. The functional coverage approach aims to perform hazard identification and mitigation assurance via a hazard analysis. The development coverage approach focuses on utilizing identified activities, also known as Level of Rigor (LOR) activities, that will assure a high-quality product that is expected to be safer. It is worth noting that safety risk cannot be eliminated in its entirety because all associated hazards cannot be eliminated. Theoretically, a low level of safety risk can be accomplished by developing a product to a specific set of development processes, requirements, analyses, and tests.
The systematic approach discussed above used by many safety experts comprises a Safety Criticality Matrix (SCM) that depends on the magnitude of potential mishap consequences (severity category) and other factors such as the function’s degree of autonomy (control category). Per MIL-STD-882E, the severity category rating measures the severity of a hazard from “Negligible (4)” to “Catastrophic (1).” The Software Control Category (SCC) rating consists of the degree of control, complexity, and timing criticality of the system, scaled from 1 to 5. The product of the severity categorization and software control categorization is the Software Critical Index (SwCI) [20]. In many other industry standards, Safety Integrity Level (SIL) is used in place of SwCI. This SIL provides the LOR determination which consists of development processes, requirements, analyses, and tests that need to be performed to attain a level of safety assurance and integrity. To develop the framework of LOR activity determination for RL, the SCM categories are adapted from traditional safety standards. It is important to note that the development team must review and tailor the proposed SCM depending on the functionality of the system, its objective, lessons learned, and requirements of an acceptable system safety effort, which may be governed by ethical and legal guidelines. The tailoring effort shall be adequately explained, justified, and approved by the acceptance authority.

As industry standards have not been developed or tailored for the inclusion of RL, the goal that I put forth is to achieve high levels of safety confidence by following the proposed approach in this research. Further, the goal is to provide the developer, end-user, or acceptance authority with a method to assess the safety risk of a system that employs RL. This method will allow the developer, end-user, or acceptance authority to determine if a system that employs RL is acceptably safe. The research includes a series of unique methods, LOR activities, that can be curated into guidelines that provide a developer or a project team with recommendations related to the development of
RL subsystems, better equipping the developer to identify, assess, and mitigate the safety risks associated with RL. To conceptualize this rather expansive topic, I utilize best practices such as those found in industry-vetted safety guidelines and other approaches for safe RL model development that result in a high level of safety confidence. It is recognized that these guidelines are not meant to replace LOR activities from traditional system safety standards. Instead, these guidelines should be used in conjunction with LOR activities from traditional safety standards to increase the safety confidence of the entire system.

Since RL safety risk measured in a controlled environment often differs from safety risk in operational settings or the real world, this paper recommends an assurance safety test to aid in the development of a safety case which also necessitates other evidence gathered through the different development stages. The assurance safety test is often considered to be the best method for formalizing evidence for complex systems in the context of hazard sources and mitigation approaches. This process will ascertain a high level of confidence that the system that employs RL has a low level of safety risk. The safety case for this research addresses a particular driving situation: a vehicle in a three-lane highway driving scenario in which the vehicle is surrounded by up to three road vehicles, in which longitudinal and lateral safe distance must be maintained. Part of this safety case utilizes the MATLAB RL toolbox for the simulation. While it is not possible to showcase all the applicable LOR activities due to resource constraints in the MATLAB-simulated environment, the approach demonstrated in this research exemplifies increasing the safety confidence of the system by lowering the safety risk using LOR activities.

The process of developing a system contains iterative stages, but some of the activities within these stages may be performed in parallel. Activity dependencies make it necessary to revisit some of the activities as new information becomes available. These stages are represented in the figure
below, with RL considered to be a subsystem. Because validation is usually conducted throughout the different stages of the model life cycle, requirements are validated through the different stages against the system-level description using traceability and reviews. Therefore, validation is not specifically reflected here. It is also important to note that, unlike traditional subsystem development, RL models involve other stages such as model data management and model training and evaluation. A complete guide to the system life cycle development is not intended.

**Figure 3.1** Reinforcement learning model development.

### 3.1.1 Platform
MATLAB Reinforcement Learning Toolbox™ provides the framework to develop, train, and evaluate RL policies. In addition, the MATLAB Deep Learning Toolbox™ is used along with the MATLAB Reinforcement Learning Toolbox™ to integrate deep learning techniques into the RL workflow.

MATLAB Automated Driving Toolbox™ provides a co-simulation framework to model and visualize performance using synthetic data in a virtual simulated environment with Unreal Engine® from Epic Games®. The toolbox offers functions and blocks for training policies to
implement controllers and decision-making algorithms for autonomous systems, which interact with the Automated Driving Toolbox™. These toolboxes are part of the Simulink environment for modeling.

### 3.1.2 Data
Initially, safety-critical applications in the context of autonomous vehicles are explored to identify relevant case studies for RL. This research depends on a completely simulated environment (synthetic data) since it is not feasible to engage in trial-and-error with an actual autonomous vehicle. Simulations (synthetic datasets) are generated in MATLAB to simulate autonomous driving scenarios, which are used to train and evaluate the agent. The MATLAB environment can generate large amounts of data, leading to an assumption of data sufficiency. There are three different types of datasets: training, cross-validation, and testing. These datasets are independently generated without the need for preprocessing or augmentation.

It can be assumed that the datasets do not contain noise impacting the algorithm; datasets are from the same distribution; and there is no adversarial behavior on the training datasets.

### 3.2 Process
This section describes the activities that provide the means for satisfying the different model development objectives. Meeting these objectives reduces the safety risk of RL operating in safety-critical systems. The primary purpose of this section is to define the recommended model development LOR activities that can be extracted, tailored, and implemented based on the detailed objectives for the applicable model development stage. As the intent of the paper is to describe an approach agnostic to the technology in use, the analyst is encouraged to tailor the approach to the technology being assessed, keeping in mind the points described below. Where tailoring is implemented, it must be adequately explained, justified, and approved by the acceptance authority.
While some of the LOR activities are derived from government and industry standards, lessons learned, and best practices, other LOR activities recommended in this research are RL-specific. As the scope of this paper is limited to tasking related to RL, these guidelines should not be used in isolation from safe autonomy considerations such as UL4000, other system safety standards, and best practices. This research only considers offline RL. The data-driven process being dependent on data exposure necessitates the identification of data distributional shift and the resulting re-training process. It is not the intent of this research to prescribe the specific tools and specific methods of how the activities described herein are performed.

The following provides source examples, but the list is not all-inclusive:

- Road Vehicles – Functional Safety (ISO 26262:2018) [21]
- Safety of Intended Functionality (SOTIF ISO/PAS 21448:2022) [22]
- Department of Defense Standard for System Safety (MIL-STD-882E) [20]
- Joint Software System Safety Engineering Handbook (JSSSEH Version 1.0) [25]
- JSSSEH Implementation Guide (JS-SSA-IG Rev. B) [26]
- NASA Software Safety Guidebook (NASA-GB-8719.13) [23]
- Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (ARP4761) [24]
- Software Considerations in Airborne Systems and Equipment Certification (DO-178C) [27]
- EASA Concept Paper: First Usable Guidance for Level 1&2 Machine Learning Applications (Proposed Issue 2) [37]
• Process Standard for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing AI (ARP6983 Rev. DRAFT 5) [38]
• Lessons learned, best practices, etc.

It is important to note that failure to perform the applicable recommended activities increases safety risk. This safety risk must be documented and communicated to the acceptance authority. The table below provides guidance for safety risk determination. It is recommended that a safety risk assessment such as the table below is implemented because historical data or failure rate cannot reliably determine the safety risk associated with an RL model.

Table 3.1 Safety risk determination.

<table>
<thead>
<tr>
<th>Risk Levels</th>
<th>Description of Risk Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High</strong></td>
<td>Loss of functionality or malfunction that upon occurring during normal or credible off-nominal operations or tests:</td>
</tr>
<tr>
<td></td>
<td>• Can lead directly to a catastrophic, or</td>
</tr>
<tr>
<td></td>
<td>• Places the system in a condition where no independent functioning interlocks preclude the potential occurrence of a catastrophic mishap.</td>
</tr>
<tr>
<td><strong>Serious</strong></td>
<td>• Can lead directly to a critical mishap, or</td>
</tr>
<tr>
<td></td>
<td>• Places the system in a condition where only one independent functioning interlock or human action remains to preclude the potential occurrence of a catastrophic mishap.</td>
</tr>
<tr>
<td></td>
<td>• Places the system in a condition where one or no independent functioning interlocks preclude the potential occurrence of a critical mishap.</td>
</tr>
<tr>
<td><strong>Medium</strong></td>
<td>• Can lead directly to a marginal or negligible mishap, or</td>
</tr>
<tr>
<td></td>
<td>• Places the system in a condition where two or more independent functioning interlocks or human action remains to preclude the potential occurrence of a catastrophic mishap.</td>
</tr>
<tr>
<td></td>
<td>• Places the system in a condition where only two or more independent functioning interlocks or human actions remain to preclude the potential occurrence of a critical mishap.</td>
</tr>
<tr>
<td><strong>Low</strong></td>
<td>• Would be a contributor to a marginal or negligible mishap, but two or more independent functioning interlocks or human actions remain.</td>
</tr>
</tbody>
</table>
3.2.1 System Definition and Safety Planning – System Development

There are several activities performed during the system definition and safety planning to ensure system safety. These activities define how the development process will identify and mitigate safety risks, define safety requirements, ensure appropriate system design, and provide evidence of compliance with safety standards, if any. Safety planning establishes a System Safety Program (SSP) within the development life cycle. When the SSP is properly implemented, it will ensure that requirements are identified, prioritized, classified, and traced through design, implementation, verification, and acceptance with minimal impact.

Some of the system definition and safety planning stage objectives include:

- The system (hardware/software) life cycle(s), including the inter-relationships between the processes, feedback mechanisms, and transition criteria are determined.
- The development process activities of the system life cycle that address the system requirements and safety criticality level(s) are defined.
- System development standards accordant with the system safety objectives for the system are defined.
- Development plans that address system safety have been produced.
- The compliance criteria according to the respective SIL(s) is proposed to the certification authority.

Some of the system definition and safety planning stage outputs include:

- System description
- Design constraints
- System safety objectives
Inadequate planning is likely to result in schedule delays, cost increases, and a potential increase in safety risk. For a more comprehensive list of objectives, activities, and outputs see specific industry standards and guidance documents.

### 3.2.1.1 Guidelines for System Definition and Safety Planning

System definition and safety planning aim to ensure that the RL within a system has been developed with a level of confidence per the SIL of the function it supports. To accomplish this, there must be proper documentation of the process to determine and meet the SIL. As described in the international standard for the safety of automotive systems, ISO 26262, SIL (known as Automotive Safety Integrity Level in the automotive industry) classification aids in defining the safety requirements [21]. Automotive Safety Integrity Level (ASIL) identification is performed through a HARA that scrutinizes the severity, exposure, and controllability of the agent operating scenario.

However, as systems expand with the use of RL, the safety risk assessment must not depend on the resulting safety risk computed from severity, exposure, and controllability. Unlike hardware, “determining the probability of failure of a single software function is difficult at best and cannot be based on historical data,” according to MIL-STD-882E [20]. Because RL can be implemented via software, it is important to develop safety risk assessment categories specific to RL.

For example, the research paper “On a Formal Model of Safe and Scalable Self-driving Vehicles” makes the argument that historical data cannot reliably verify that an autonomous agent meets the
safety requirements. According to the authors’ reasoning, an agent would need to drive more than $10^6$ hours without a single fatal accident to claim a probability equal to or safer than a human driver, with a human driver having a probability of a fatal accident of 1 in $10^6$ hours. However, to introduce to the market autonomous agents as a safer alternative to human drivers, developers must strive for a more rigorous target, such as $10^{-9}$, which requires 30 billion meaningful miles, or “high value” driving experiences as opposed to driving on empty roads, to provide evidence of meeting the requirement [39].

Another part of the challenge of utilizing historical data to verify safety requirements presents when there are changes to the model, which poses a specific concern for RL algorithms as the behavior may change with each learning evolution. “On a Formal Model of Safe and Scalable Self-driving Vehicles” also notes that for a single change, “testing must start all over again as it is impossible to know if that code change has resulted in a new failure that was not present during the first 30 billion mile drive” [39].

Additionally, ASIL’s dependence on controllability poses a challenge in autonomous vehicles. According to the “The Notion of Controllability,” controllability definitions and categorization are developed by “domain experts” that assess human factors in vehicle handling. However, the controllability categorization is no longer applicable to autonomous vehicles [40].

For these reasons, the safety risk assessment shall eliminate the probability of relying on historical data and controllability as determining factors. Instead, other processes found in MIL-STD-882E provide a more valid approach to safety risk determination which relies on the severity and the degree of control that software exercises over hardware, known as the control category [20].
The product of the severity and control categorization is the SIL. This SIL provides a measure of development coverage activities to achieve high confidence in safety. These development coverage activities, also known as LOR, dictate how systems are architected, designed, implemented, and verified. These development coverage activities in conjunction with the functional coverage activities aim to increase the safety confidence that the system will function as intended.

In addition, the SIL approach implements a systematic process that considers other factors such as cost throughout the system life cycle. This approach is organized in a matrix form known as SCM. In the ideal scenario, all LOR activities would be applied to all subsystems regardless of the criticality level. In practicality, an acceptable level of safety approach must be balanced with the cost associated with a rigorous development process. When an SSP is successfully implemented via an SCM, it will ensure that requirements are identified, prioritized, classified, and traced through design, implementation, test, and acceptance with minimal impact.

To address the challenges associated with RL, this research defines SILs for RL, Reinforcement Learning Integrity Levels (RLILs). These RLILs are used to determine RL-specific LOR activities that will increase the safety confidence that the system will function as intended and are not meant to replace LOR activities from traditional system safety standards as seen in the figure below. RLIL dictates RL development processes, requirements, analyses, and tests that need to be performed to attain a level of confidence that the system is safe. The RLILs are adapted from traditional safety standards to address RL.

To achieve the RLIL objectives, the developer should plan and tailor the set of activities described in this research. The developer may utilize alternative methods to the guidelines described.
However, the developer should provide evidence that the RLIL objectives have been met. It is important to know that not all RLIL activities will have the same effectiveness. Some will impact the safety confidence more than others.

![Diagram of System safety assurance and integrity process.](image)

To develop the framework of activity determination to meet the project objectives, the development team must review and tailor the severity categories, control categories, SCM, and LOR activities depending on the functionality of the system, its objective, lessons learned, and requirements of an acceptable system safety effort, which may be governed by ethical and legal guidelines. Existing standards may provide valuable insight for the derivation of the safety risk assessment approach. The tailoring effort shall be adequately explained, justified, and approved by the acceptance authority. The SCM proposed in this research and used to determine the LOR
activities required to be imposed on the RL subsystem for the safety assurance and integrity of the functionality is included below.

Table 3.2 Reinforcement learning safety criticality matrix.

<table>
<thead>
<tr>
<th>CONTROL CATEGORY</th>
<th>SEVERITY CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Catastrophic (1)</td>
</tr>
<tr>
<td>Autonomous</td>
<td>RLIL 1</td>
</tr>
<tr>
<td></td>
<td>Critical (2)</td>
</tr>
<tr>
<td>Semi-Autonomous</td>
<td>RLIL 2</td>
</tr>
<tr>
<td>Redundant Fault Tolerant</td>
<td>RLIL 3</td>
</tr>
<tr>
<td></td>
<td>Marginal (3)</td>
</tr>
<tr>
<td></td>
<td>RLIL 4</td>
</tr>
<tr>
<td></td>
<td>Negligible (4)</td>
</tr>
</tbody>
</table>

Note: The functionality could also be assessed as having no safety impact, RLIL5.

3.2.2 System Requirements – System Development
Identification of system requirements is a fundamental input to the system safety process which is executed alongside or immediately after hazard analyses. A description of each different type of hazard analysis can be found in MIL-STD-882 [20].

The system requirements are expected to be generated from the relevant standards and regulations or derived by domain experts. According to the NASA Software Safety Guidebook, “The requirements of software components are typically expressed as functions with corresponding inputs, processes, and outputs, plus additional requirements on interfaces, limits, ranges, precision, accuracy, and performance. There may also be requirements on the data of the program set, its attributes, relationships, and persistence, among others” [23].

The system requirements levied from standards or regulations, either established internally by the program or externally by government or industry must be cited within the project documentation and tailored for the specific project as applying existing standards in their entirety to new
technologies may be invalid. Such system requirements provide the minimum boundaries that must be met to ensure that the system is safe and will not cause harm to people, itself, other systems, or the environment. The system definition and safety planning stage provides an opportunity to review and determine acceptable deviations. These should be specifically approved by the acceptance authority.

Once the regulatory and standard requirements have been identified, the available specific system information is gathered to determine the project-specific safety requirements. These safety requirements are allocated to both software and hardware to prevent or mitigate the effects of functional failures and malfunctions. Usually, these will be derived during the system requirements stage and further refined and verified by subsequent stages. Understanding the system design and its intended functionality is imperative for a systematic derivation of safety requirements that include functional, reliability, integrity, and design constraints. The main method for the systematic derivation of safety requirements is through hazard analyses, which is a fundamental step in the safety assessment process. Industry and military standards in the aerospace industry indicate that a Preliminary Hazard Analysis (PHA) and a Functional Hazard Analysis (FHA) be completed as some of the first hazard analyses to identify high-level requirements. In addition, the FHA aids in the identification of safety-significant functionality and their respective SIL. Higher SIL levels (e.g., RLIL 1) require higher levels of safety assurance and integrity within the development process. To develop a high level of safety confidence in the model, it is paramount to properly execute the applicable LOR activities.

The two types of requirements, generated from the relevant standards and regulations or derived by domain experts, are prioritized, categorized, and traced through design, implementation, test, and acceptance.
One of the system requirements stage objectives is:

- Requirements are identified and documented, including derived high-level safety requirements from the PHA and process requirements from the FHA.

Some of the system requirements stage outputs include:

- Identification of hazards and failure/malfunction conditions
- Derived high-level system requirements
- Process requirements (level of rigor associated with the development assurance activities)

The main purpose of this stage is to identify SIL(s) to define the applicable LOR activities and to derive complete and accurate requirements to mitigate, detect, and compensate for functional failures and malfunctions. For a more comprehensive list of objectives, activities, and outputs see specific industry standards and guidance documents.

3.2.2.1 Guidelines for System Requirements

The primary purpose of the PHA and the FHA is to derive high-level safety requirements to eliminate or mitigate hazards. In addition, the FHA aims to identify all safety-significant functionality (including RL functionality), assess functional failure and malfunction consequences, and assign SIL(s) and respective LOR activities to each safety-significant functionality. As both the PHA and FHA are analyses for systematic derivation of high-level requirements, the FHA is especially helpful in identifying process requirements. Therefore, these guidelines emphasize the significance of the FHA and the steps to effectively perform the FHA.

During the FHA, it is important to recognize that mishaps are not only caused by functional failures or malfunctions. They can also be caused by element interactions within the system or interactions
with the environment. For example, at automation level 3 per SAE J3016, a hazardous condition may be caused by the human operator overestimating the abilities of the autonomous driving assistant which can lead to a reduction of human monitoring, a result of human misuse, and subsequent reduction of the driver skill level which is critical for correcting any system malfunctions [41]. Therefore, it is important to consider an analysis that derives requirements to mitigate hazards as of result of functional failures, malfunctions, and unsafe interactions. To best accomplish this, the System-Theoretic Process Analysis (STPA) method may be valuable to augment the FHA.

According to the STPA Handbook, the STPA methodology first identifies the accidents and hazardous states that need to be mitigated. The next step in the STPA process is to create a hierarchical control structure, a model of the system, derived from functional requirements. This control structure represents the functional relationships within the system through feedback control loops that begin as abstract and become more detailed over time. To identify potential accident causal factors, the analyst assesses the model for control actions that could lead to the defined losses identified during the first step. These are known as Unsafe Control Actions (UCAs), and their identification requires an explanation of why these UCAs may manifest in the system. These UCAs are used to derive high-level safety requirements and constraints for the system [42].

According to “Considerations in Assuring Safety of Increasingly Autonomous Systems,” there are several challenges to successfully implementing the STPA methodology [43].

These main challenges are determining hazards from complex architecture, the complexity of authority and interactions between the different intelligent controllers (e.g., controllers trained using RL), and subjectivity in determining UCAs and causal factors. To overcome these challenges, researchers propose ensuring elements have a coherent relationship with the
architecture, extending the STPA methodology to include functionalities and assumptions of each entity using specific notations and techniques, automating part of the analysis, and using simulations to identify additional hazardous scenarios [43].

To create a coherent relationship with the architecture, the safety practitioner should work closely with the developers to ensure the hierarchical control structure accurately represents the functional architecture. This will ensure the identification of unforeseen hazards from the possible interaction between components. To build upon the STPA methodology, the functional assumptions should include communication between entities inside the hierarchical control structure. This will minimize the subjectivity with which UCAs are identified in this process. In addition, STPA is most effectively utilized when subjectivity is minimized. This means that objective means rather than dependence on the analyst’s experience, knowledge, and skill must be included. Therefore, automation and simulations should be used to accurately identify hazards.

After the identification of unsafe scenarios, formalization of the safety requirements must occur. These safety requirements are derived from unsafe control actions or hazards and developed to specify constraints in training, cross-validation, and testing to name a few. To develop a consensus of the requirements within a program, each high-level requirement must be written in objective language to ensure that it is accurate, unambiguous, verifiable, consistent, traceable, sufficiently detailed (quantitative terms and tolerances where applicable) and that the requirements do not conflict with each other.

While natural language is often used to write requirements, it may not be the most effective method for specific RL requirements because of subjectivity and interpretability. Formal notation is also used because of its preciseness but may be challenging to write and understand. Therefore, the
authors of “Considerations in Assuring Safety of Increasingly Autonomous Systems” propose leveraging the benefit of both methods through a property-based requirements notation to provide ease of interpretation for humans and machines. Property-based requirements notation should capture system and environment properties [43].

Also, there are additional hazards associated with RL and its limitations (e.g., estimated low performance in visibility conditions). In addition, RL model faults must also be considered. According to the NASA Software Safety Guidebook, “A fault is any change in the state of an item which is considered anomalous and may warrant some type of corrective action. A failure is the inability of a system or component to perform its required functions within specified performance requirements” [23]. A fault can become a failure when a system is unable to detect and/or recover from a fault. Similarly, a failure can become a hazard when a system is unable to detect and/or recover from a failure. Researcher. indicate that an example of a fault leading to a system hazard occurs when faults in the neural network result in inadequate generalization. These faults may be caused by too steep of a learning rate, inadequate layer connectivity, etc. [31]. Although these faults are not exclusive to RL, these types of faults still need to be considered for comprehensive hazard analysis.

The FHA is essential in identifying process requirements that will dictate which activities need to be performed to meet the project objectives. The STPA method is a valuable tool that could be utilized in an FHA for the identification of hazards associated with functional failures, malfunctions, and unsafe interactions.

3.2.3 System/Subsystem Design – System Development
Included within the design stage are the static and dynamic aspects that will meet the high-level requirements, which are used to develop detailed architecture and low-level requirements to later
be implemented. The detailed architecture and low-level requirements are later used to influence other design features such as the RL model.

Paramount to the design stage is designing with minimum safety risk in mind. This means that design choices are made to eliminate hazards or reduce the risk of hazards. Various architectural approaches can help do this. Some of the architectural strategies include multiple-version dissimilar implementations, partitioning, and safety monitoring.

Low-level requirements are essential to system safety. Low-level requirements aid in defining the behavior and constraints of individual components. In addition, low-level requirements facilitate fault tolerance mechanisms and error-handling procedures.

Some of the system/subsystem design stage objectives include:

- The system/subsystem architecture, the preferred system architecture for safety (partitioning, safety monitoring, etc.), is developed from high-level requirements.
- The low-level requirements are developed from high-level requirements, including derived low-level safety requirements from detailed hazard analyses.

Some of the system/subsystem design stage outputs include:

- Derived low-level requirements
- System/subsystem architecture

The main purpose of this stage is to identify design features and methods to eliminate hazards associated with functional failures and malfunctions as well as hazards associated with element interactions with the environment. If hazards cannot be eliminated, the developer should incorporate design features and methods to detect, mitigate, and compensate for these hazards. For
a more comprehensive list of objectives, activities, and outputs see specific industry standards and guidance documents.

3.2.3.1 Guidelines for System/Subsystem Design
The process of assessing RLIL requires an understanding of the level of independence between components in terms of both their function, which includes high-level requirements, and their design, which encompasses shared design elements and could extend to programming languages and tools. Therefore, architectural strategies such as multiple-version dissimilar implementations, partitioning, and safety monitoring, need to be considered for an appropriate criticality assessment. These architectural strategies need to consider methods for fault detection, fault isolation, conflict resolution, and fail-safe as the inherent uncertainty present in RL subsystems tends to spread and magnify at the system level. By revisiting the assessment of the RLIL, the development process can ensure the correct application of the required LOR. This enables the establishment of appropriate safety assurance and integrity activities, including verification and validation, aimed at reducing the safety risk as a result of hazards associated with functional failures, malfunctions, and with element-environment interactions.

3.2.4 Conceptual Model Design – RL Model Development
According to SAE’s “Aerospace Recommended Practice ARP6983,” it is necessary to explicitly define the system operating domain specific to the RL-based subsystem's function to lower the safety risk. This operating domain is defined as the Operational Design Domain (ODD), which must accurately reflect operating conditions with appropriate margins to accommodate abnormal scenarios [38].

As discussed in “EASA Concept Paper: First Usable Guidance for Level 1 & 2 Machine Learning Applications,” the risk-based methodology is driven by a requirement-based safety assurance and
integrity process [37]. In addition, the effectiveness of an RL-based subsystem is determined by its ability to satisfy the higher-level requirements that have been flowed down to the RL subsystem. To meet this intent, there are requirements posed on both the data and the design. Requirements on the data aid in ensuring that appropriate training, cross-validation, and verification sets are obtained, which is measured through characteristics such as data relevancy, completeness, balance, accuracy, traceability, and synchronization. Requirements on the design aid in ensuring model generalization, accuracy, reliability, and robustness. This requirement derivation and generation process may be iterative since additional requirements may be identified during subsequent stages.

An additional step before model data management and model design can begin is selecting and defining the RL environment. This includes selecting methods, tools, notations, programming languages, and libraries for the RL environment.

Some of the conceptual model design stage objectives include:

- The ODD is accurately characterized.
- RL model requirements are derived.
- The RL environment is selected and defined.

Some of the conceptual model design stage outputs include:

- ODD characterization
- RL model requirements on data
- RL model requirements on design
- RL environment

The LOR activities for the above-mentioned objectives and outputs are listed below.
3.2.4.1 Guidelines for Conceptual Model Design

Before RL model development can begin, functionality must be allocated to the RL model. This functional allocation is performed to characterize the ODD and to derive RL requirements which are later implemented in the detailed design stage. The RL-related requirements shall provide an adequate definition of the RL functionality, including limitations so that subsequent tasking can be performed. Some limitations are known and documented in the system description while others are revealed because of subsequent activities (i.e., hazard analyses). These activities will identify limitations, consequences, and mitigating measures. During this process, there should be careful
notation of functional requirements, and more importantly, safety-critical functionality that must be maintained to prevent a catastrophic event (such as death or loss of system).

3.2.4.1.1 ODD Characterization
One of the key elements of training an agent during later steps is the accurate description of the ODD. This poses a challenge, however, because disturbances and noises prevent a true duplication of the ODD in a digital model. Part of the process to create an ODD will be the identification of assumptions. A violation of these assumptions may potentially negate the verification performed at a later step in the process. According to van Wesel and Goodloe’s paper “Challenges in the Verification of Reinforcement Learning Algorithms,” the assumptions that should be investigated include operating environment assumptions, platform assumptions, assumptions on data, and assumptions on the algorithm [3].

To characterize the ODD, referred to by some authors as the “environment,” initial conditions and dynamic behavior are necessary elements to detail. In addition, agent interactions with the environment shall be defined with action and observation signals. According to authors in “Toward Verified Artificial Intelligence,” the main challenges for environment characterization are unknown variables, characterizing with fidelity, and defining human behavior. To overcome these challenges, Seshia et al. propose an introspective environment model, algorithmically identifying the system’s assumptions about the environment that fulfills the specification, active data-driven modeling, use of expert knowledge to define the environment, use of environment interactions to update the environment at run time, probabilistic formal modeling, and combining non-deterministic and probabilistic modeling [44].
3.2.4.1.2 RL Requirements
The process of model requirement derivation is iterative and can be broken down into two types of requirements, model requirements on data and model requirements on design.

Imposing requirements on the data should ensure that datasets are relevant, complete, balanced, accurate, traceable, and synchronized. Ashmore et al. define some of these qualities. According to researchers, data relevancy refers to datasets that reflect possible scenarios in the intended operational environment. Data completeness evaluates the different scenario possibilities (e.g., nominal and hazardous scenarios) and the congruency of the data features in the operating environment, requiring coverage of each state and action. Balanced data, on the other hand, refers to the data’s distribution, requiring enough of each scenario across datasets, ensuring that the agent encounters a diverse set of states and actions. Accurate data considers aspects such as the collection method (e.g., sensor and simulation accuracy) which impact the representativeness of the samples in the operational environment [44]. According to SAE’s “Aerospace Recommended Practice ARP6983,” data traceability refers to the ability to trace the origin of the data used for training, cross-validation, and verification. Data synchronization ensures data integrity and coherence across the ODD. Both data qualities are crucial for ensuring transparency, reproducibility, and understanding of the training and verification process [38].

Imposing requirements on the design should ensure that the model is robust, accurate, reliable, and able to generalize. According to the National Institute of Standards and Technology in “Artificial Intelligence Risk Management Framework (AI RMF 1.0),” robustness is the ability to provide consistent performance on tasks that are similar to training (but not identical) in the presence of abnormal conditions and data distributional shift (deviation between the training data distribution and the distribution of data during verification). Accuracy is the similarity of perceived
observations, computations, or estimates to true values. Reliability is the model’s ability to execute as required over time intervals, measuring the correctness of the expected use [46]. Generalization is the ability of an agent to provide consistent performance during normal conditions on tasks that are similar but not identical to the training task.

3.2.4.1.3 RL Environment
Identifying appropriate methods, notations, programming languages, and libraries for an RL environment is essential to building a safe RL subsystem. Some of the considerations should take into account the complexity of the RL environment and whether it requires a physical simulation. Simulators can significantly speed up training, especially for real-world tasks, where data collection may be slow, costly, or hazardous. In addition, other elements such as the integration process and compatibility with existing infrastructure should be taken into consideration. By any means of implementing the RL environment, tool qualification should be performed following process activities described in existing design assurance guidance documents to ensure tools will not introduce errors or fail to detect errors.

3.2.5 Model Data Management – RL Model Development
The success of deep learning techniques in a real-world environment can be largely attributed to the capability of neural networks to acquire generalized models from the environment using massive amounts of data. Collecting massive amounts of data for a safety-critical system in a real-world environment could be hazardous. However, existing large data collections or tools that generate synthetic data can aid in the learning process of an RL agent. To complement existing datasets, it has been shown that training agents in a simulated environment can improve learning and generalization. According to researchers, this simulated environment does not need to be a high-fidelity representation of the real world. High-fidelity simulations do not necessarily correlate
to good learning and generalization [47]. RL agents utilizing these large collections of real and simulated data will be better equipped to generalize to the real world.

According to “EASA Concept Paper: First Usable Guidance for Level 1 & 2 Machine Learning Applications,” “The test data set should be ideally collected from real data, complemented by synthetic data where appropriate (e.g., data at the limit or beyond flight envelope)” [37]. In this approach, datasets should be analyzed for consistency. This ensures datasets do not conflict (e.g., state or action spaces in different datasets are not consistent, leading to conflicting behaviors).

The primary input of the model data management stage is the requirements for the model, of which many are derived and flowed down. These requirements should ensure that the datasets are relevant, complete, balanced, and accurate. These requirements should ultimately ensure that the model can generalize beyond the training conditions and ensure accuracy, reliability, and robustness. As a result, a safety case argument will consider the systematic process of analyzing, collecting, preprocessing, and augmenting the datasets. In addition, datasets need to be properly managed to ensure traceability and synchronization.

The output artifacts from the model data management stage are a training dataset and a verification dataset. It is a best practice to also use a cross-validation dataset during the training stage to evaluate the model's performance, which helps identify concerns such as overfitting.

Some of the model data management stage objectives include:

- Training, cross-validation, and verification datasets are relevant, complete, balanced, and accurate.
- Training, cross-validation, and verification datasets are properly managed to ensure traceability and synchronization.
Some of the model data management stage outputs include:

- Training dataset
- Cross-validation dataset
- Verification dataset

Appropriate metrics (e.g., situation coverage, state distribution, outlier detection) with the applicable documentation should be utilized to guide these two activities to attain dataset relevancy, completeness, balance, and accuracy.

The LOR activities for the above-mentioned objectives and outputs are listed below.

Table 3.4 Level of rigor activities for model data management.

<table>
<thead>
<tr>
<th>Level of Rigor Activity</th>
<th>RL Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>A structure and organization of data used for training, cross-validation, and testing datasets is specified for the evaluation of dataset relevancy, completeness, balance, and accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Appropriate metrics (e.g., situation coverage, state distribution, outlier detection) are utilized to guide the data generation/collection process and possibly the augmentation process to attain dataset relevancy, completeness, balance, and accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Data sources are identified, chosen, and analyzed for all possible variations to ensure data relevancy, completeness, balance, and accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>- Hazard analyses are included in the systematic identification of hazardous behaviors during generation/collection and augmentation of datasets to establish dataset completeness. Hazardous behaviors include hazards associated to previous experience (historical information), loss of functionality, and malfunction.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Trusted data sources in conjunction with data-transit guarantees are utilized to ensure dataset accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Experimental design methods (e.g., randomized environment) are utilized during the collection and augmentation of datasets to ensure dataset relevancy, completeness, balance.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Datasets are collected from real data and may be complemented by synthetic data to ensure dataset completeness and balance.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Data is augmented to account for sensor error to ensure dataset accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Adversarial examples, specifically designed to cause the model to make a mistake, are used to augment datasets to establish completeness.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Datasets are analyzed for relevancy, completeness, balance, and accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>- Exploratory data analysis (systematic and controlled collection of datasets), which understands the structure and properties of the data (identifying patterns, trends, and relationships in the data, and detecting any potential problems or anomalies) are implemented to ensure dataset relevancy, completeness, balance, and accuracy.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Training, cross-validation, and verification datasets are properly managed for each model or submodel to ensure traceability and synchronization.</td>
<td>R R R R</td>
</tr>
<tr>
<td>- Training dataset should be larger than the verification dataset</td>
<td>R R R R</td>
</tr>
<tr>
<td>Independent datasets are utilized but are derived from the same dataset distribution, when possible, to prevent data leakage (knowledge of verification data in the training data).</td>
<td>R R R R</td>
</tr>
<tr>
<td>Proper documentation and justification of potential weakness and uncertainty in the data collection and augmentation process is provided to ensure dataset relevancy and accuracy.</td>
<td>R R R R</td>
</tr>
</tbody>
</table>

Note: “R” indicates an activity required for the RLIL.
3.2.5.1 Guidelines for Model Data Management
The initial step in the model data management stage involves collecting data samples through the process of observing and gathering experience data from an agent’s interaction with an environment. However, collecting massive amounts of data for a safety-critical application through this process may be hazardous. Existing large data collections can aid in the learning process of an RL agent to reduce this safety risk. These data collections are a result of experience data gathered from observing and measuring a real-world system or representation to build an RL model, according to “Assuring the Machine Learning Lifecycle: Desiderate, Methods, and Challenges” [44]. To ensure the accuracy of datasets, it is important to use reliable data sources in combination with guarantees for data transit.

Sometimes, data collected from various sources may have differences in their characteristics, which may necessitate preprocessing to create consistent datasets for training and verification. Preprocessing may also involve simplifying complex data, removing noise, and selecting features that can assist with training. It may also be possible to utilize synthetic data in this process. This synthetic data may be obtained through simulations. When using synthetic data, the allocation of data into training, cross-validation, and test datasets is not applicable because scenarios are created and run in real time. It is important to note that the scenario generation for each of the datasets (training, cross-validation, and testing) needs to be independent. However, proof that this synthetic data is representative of the target environment is required [44]. Therefore, accreditation and validation of simulations and models are required.

Ashmore et al. also state that it is necessary to analyze aspects of the data collection such that data is relevant, complete, balanced, and accurate. To achieve these data qualities, a systematic derivation of requirements is necessary. This systematic derivation includes hazard analyses,
which drive the data collection process in conjunction with robust data collection methods. Additionally, some authors propose supporting the data relevancy by properly documenting the justification for how the data is collected, whether it be through experimental design or simulations [44]. For completeness, the authors in “Ontology based Scene Creating for the Development of Automated Vehicles” propose expert knowledge to systematically identify hazardous behaviors in operating scenarios [48]. However, it is not practical to identify all nominal and hazardous scenarios through expert identification. This technique must also be utilized in conjunction with a systematic process to collect data guided by a coverage metric such as the one proposed by “Towards Dependability Metrics for Neural Networks” [49]. Like completeness, balance requires a systematic process to collect data guided by a balance metric. Like relevancy, some authors propose supporting the data accuracy by properly documenting potential weaknesses [44] or uncertainty in the data collection process so it can be accounted for in the latter stages. Robust data collection guided by the derivation of requirements improves the ability of the model to generalize in the target environment.

In addition, exploratory data analysis provides a systematic process and may offer assurance that the dataset is relevant, complete, balanced, and accurate. Exploratory data analysis requires analyzing and visualizing the data to better understand the dynamics between the environment and the agent. To build a compelling safety case argument, some challenges need to be addressed. One challenge, according to Domingos in “A Few Useful Things to Know About Machine Learning,” is figuring out how much data is needed to achieve good generalization [50]. Also, another challenge according to Faria in “Machine Learning Safety: An Overview” is identifying and including credible unlikely events in the training dataset. In addition, many RL implementations rely on the premise that the distribution of the data used for training is the same as the distribution
of the data that the model will encounter during operation. However, in real-world scenarios, this premise may not be valid because of challenges in collecting enough training data [7].

When datasets are too hazardous to collect, too time-consuming to collect, or unavailable, data augmentation is a technique employed to supplement the existing dataset by generating additional samples based on existing ones, which can be performed in a simulated environment. According to some researchers, data augmentation is commonly utilized to enhance the ability of a model to generalize. Often this is accomplished by intentionally increasing the data amount and diversity. Datasets may also be augmented by implementing randomized data variations or performing normalization to improve data generalization [32]. Data augmentation may also be implemented to account for sensor errors to better the performance of the model. This may be accomplished by adding random noise, applying distortion, and removing portions of the dataset, among other strategies. In addition, datasets may also be augmented by generating adversarial examples to improve policy robustness.

Data augmentation is primarily utilized to achieve dataset completeness, balance, and accuracy. Therefore, data augmentation can aid in generalization, specifically in situations where the amount of data is insufficient, noisy, or highly variable. Similar to the collection of datasets, data augmentation obtained through simulations requires accreditation and validation of simulations.

3.2.6 Detailed Model Design – RL Model Development
Included within the design stage are several key elements that will meet the higher-level requirements. Through an iterative approach, this process centers around the identified learning environment, with the primary goal of determining the policy type and developing the logical architecture, objective function, and learning algorithm. These elements serve as the foundation for building, training, and optimizing the model. These elements are crucial for prioritizing safety
because they directly influence the behavior and decision-making of the agent within its environment.

Paramount to the design stage is designing with minimum safety risk in mind. This means that design choices are made to eliminate hazards or reduce the safety risk of hazards. The goal for RL should be to learn a policy that maximizes a reward function while avoiding hazardous situations, which are mainly caused in RL by model mismatches and inadequate policy [7]. Also, it is important to note that due to function approximation such as neural networks, there is still a probability of an unsafe action even after convergence. These neural networks are a key element of the logical architecture, which may impact the model’s composition. The model composition may consist of one or multiple model elements, each model implemented by a separate neural network.

Recent researchers have referred to the goal of maximizing the reward function while maintaining safety as Safe Reinforcement Learning (SRL). According to García & Fernández, “Safe Reinforcement Learning can be defined as the process of learning policies that maximize the expectation of the return in problems in which it is important to ensure reasonable system performance and/or respect safety constraints during the learning and/or deployment processes” [34]. SRL is implemented through the means of change in the objective function, also known as optimization criteria, and change in the exploration process. While the optimization criteria and the exploration process are shaped during the design stage, they must be properly executed during later stages. The developer must be able to implement a deterministic policy that can generalize common hazards to an unseen setting and discriminate unsafe inputs. A balance must be found between computational efficiency, generalization, and discrimination. Therefore, the developer must consider implementing the SRL design assurance methods mentioned above, taking into
account the advantages and disadvantages of these approaches. It is important to note that sound design engineering practices are not a substitute for other LOR activities such as testing.

Also, during the design stage, other design assurance methods should be implemented depending on the RLIL that help reduce the safety risk. Some of these design assurance methods include uncertainty estimation, out-of-distribution detection (OOD), distribution shift detection, and bias-variance optimization among others. Like SRL design assurance methods, a balance must be found between computational efficiency, generalization, and discrimination. Therefore, the developer must consider implementing the different design assurance methods, taking into account the advantages and disadvantages of these approaches.

Some of the detailed model design stage objectives include:

- The model logical architecture is developed.
- The objective function is developed.
- The learning algorithm is developed.
- The type of policy is determined.

Some of the detailed model design stage outputs include:

- Model logical architecture
- Objective function
- Learning algorithm(s)
- Policy type

The main purpose of this stage is to identify design features and methods to eliminate hazards caused by element interactions within the system or interactions with the environment. If hazards
cannot be eliminated, the developer should incorporate design features and methods to detect, mitigate, and compensate for these hazards.

The LOR activities for the above-mentioned objectives and outputs are listed below.

Table 3.5 Level of rigor activities for detailed model design.

<table>
<thead>
<tr>
<th>Level of Rigor Activity</th>
<th>Activity Details</th>
<th>RL Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfaces are defined in the form of data flow and control flow.</td>
<td>- A data analysis is conducted to evaluate the data's intended use to ensure that the architecture, objective function, and policy is appropriate for the specific application. - An interface analysis is conducted to ensure compatibility with components and system in the environment, and to ensure compliance with requirements and constraints</td>
<td>R R R R</td>
</tr>
<tr>
<td>The model or submodel architecture(s), objective function(s), learning algorithm(s), and other elements are developed from the derived requirements and consistent with interfaces to ensure safety and performance criteria (e.g., accuracy, reliability, generalization, and robustness).</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>The type of policy is determined from the derived requirements and consistent with other model elements to ensure safety and performance criteria.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>Bias-variance optimization is employed and provides evidence of the reproducibility of the training process.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>Uncertainty characterization is employed to maintain the integrity of the model output.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>Out-of-distribution detection guarantee is employed to maintain the integrity of the model output.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>Distribution shift detector is employed to maintain the integrity of the model output.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>Optimization criterion for safe reinforcement is employed to prioritize safety.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>Change in the exploration process for safe reinforcement learning is employed to prioritize safety during learning with hardware.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>An element analysis is conducted to examine model elements for hazard and mitigation identification, this includes a logic analysis to evaluate the safety-critical equations, objective function(s), learning algorithm(s), and control logic of the model.</td>
<td></td>
<td>R R R R</td>
</tr>
<tr>
<td>A functional analysis is conducted to ensure compliance with requirements.</td>
<td></td>
<td>R R R R</td>
</tr>
</tbody>
</table>

Note: “R” indicates an activity required for the RLIL.

3.2.6.1 Guidelines for Detailed Model Design

One fundamental concern is how to prevent the agent from making unsafe decisions when confronted with inputs that greatly differ from the data it has been trained on so far. While some approaches may help with this challenge, there is no universal solution to this issue. Successful strategies often involve a combination of various techniques and thoughtful design choices to achieve optimal performance in diverse environments.

3.2.6.1.1 Bias-Variance Optimization

In RL, bias and variance refer to two common sources of error that can affect model generalization. Bias error occurs when a complex problem is approximated using a simpler model. When a model has high bias, it tends to underfit the data, indicating its inability to grasp the underlying patterns
and relationships in the dataset. Such a model makes overly simplistic assumptions, resulting in suboptimal performance not only on the training data but also on unseen data.

Variance errors, on the other hand, describe the model’s sensitivity to changes in the training data. A model with high variance tends to overfit the data, indicating that it performs successfully on the training data but struggles to generalize to unseen data. Overfitting happens when the model memorizes noise or outliers in the training data instead of understanding the genuine underlying patterns.

There are different methods to address bias and variance. An example of one of the most common bias reduction methods includes increasing the capacity of the function approximation. Common variance reduction methods include experience replay, different exploration strategies, and regularization techniques such as dropout or weight decay. It is important to note that bias and variance optimization methods can be used in conjunction. Various RL algorithms and frameworks may adopt a blend of these approaches to manage the trade-offs between bias and variance. The selection of the method relies on the RL problem, the architecture, and other elements. An effective bias-variance optimization can lead to more generalizable models.

3.2.6.1.2 Uncertainty Characterization
Uncertainty characterization in RL involves estimating and handling uncertainties associated with different facets of the model and environment. Uncertainty can stem from multiple sources, such as stochasticity in the environment, scarce data, and the exploration-exploitation dilemma in RL. Uncertainty characterization in RL can be performed to support key model characteristics such as generalization and robustness. Some common methods employed are model-based RL with ensemble models, Bayesian neural networks, and bootstrapped ensembles, among others. Notably, the choice of uncertainty characterization method relies on the specific RL problem, the available
data, and the computational resources. Effectively managing uncertainty can lead to more
generalizable and robust models, demonstrating strong performance in a wide array of complex
and diverse environments.

3.2.6.1.3 Out-of-Distribution Detection Guarantee
OOD detection guarantee in RL refers to the model’s ability to detect instances that deviate
considerably from the training dataset. During training, the agent learns from its interaction with
the environment, forming the basis of its knowledge. OOD detection enables the agent to identify
when it confronts situations outside of its training dataset, enabling it to put the system in a safe
state. Some common methods implemented include density estimation, one-class support vector
machines, and reconstruction error. Selecting an appropriate OOD detection guarantee method
tailed to the RL problem is crucial. Effective OOD detection plays a vital role in the robustness
and safety of the model.

3.2.6.1.4 Distribution Shift Detector
A distributional shift detector is designed to identify drifts in the dataset. The agent acquires
knowledge through interactions with the environment, and when the environment's characteristics
evolve, the data distribution can also change. This concept is referred to as distribution shift.
According to Juba in “Query-Driven PAC Learning for Reasoning,” a combination of formal
methods and data-driven methods can be implemented to address distribution shift. This is based
on learning a "filter" on the training distribution, resulting in evaluation from the raw observation
and internal state data available to the system. It is also able to utilize observation/state values from
the training distribution to satisfy the filter condition and provide a worst-case guarantee of a safe
system when the filter satisfies the safety properties [51]. Selecting an appropriate distribution shift
detector method is crucial, as it should be tailored to match the particular RL problem, anticipated
types of distribution shifts, and available data. A well-functioning distribution shift detector allows
the model to adaptively address environmental changes, ensuring reliable performance in the real world.

3.2.6.1.5 Optimization Criterion
One element of risk is associated with the stochasticity of the environment, which means that even an optimal policy may result in mishaps when operating in a stochastic environment. Maximizing the return does not prevent all mishaps. Therefore, a criterion that evaluates risk associated with the stochastic environment is needed. This can be implemented by augmenting the return by including a measurement of risk, such as the variance of returns [52] [53] or the potential for worst-case scenarios.

RL algorithms iterate to find the optimal solution. This optimal solution, or the ultimate goal, is evaluated through the objective function. According to Rudner and Toner, to design an RL model that operates safely, correctly defining the objective function for a specific task is key because it specifies how the model should change as it receives more information via a reward system. The challenge to accurately reflect the developer’s intentions in the objective function becomes more problematic with a complex system and complex operating environment [54]. With the ultimate goal in mind and leveraging the derived safety requirements, the objective function is specified.

One factor that makes the specification of the objective function challenging is describing objectives or complex stochastic environments in mathematical terms. Another factor is that human developers may not anticipate every possible decision or situation when the system is operating in such a stochastic environment. These safety concerns may manifest even with infinite datasets and optimal learning. If left undetected during the model development, these safety concerns may later slowly present in the operating field with possibly catastrophic results.
Two safety concerns are associated with incorrectly specifying the objective function, reward hacking, and negative side effects. These safety concerns are a result of oversimplifying the objective function or overlooking the alternative consequences as the developer tends to focus on maximizing the cumulative sum of the rewards or minimizing other metrics.

According to “Concrete Problems in AI Safety,” it is beneficial to analyze how an objective function may be corrupted (or “hacked”) when determining the objective. However, with a complex system in a complex environment, determining the most well-suited objectives may be challenging. Therefore, the authors suggest mitigating the hazards that may result from reward hacking. Similarly, negative side effects can be approached by designing the objective with the agent as well as the environment in mind. For example, the authors suggest that instead of designing the objective function as “perform task X,” they suggest designing the objective function as “perform task X subject to common-sense constraints on the environment.” To put this concept into a structured framework, various methods can be utilized, including defining an impact regularizer, learning an impact regularizer, penalizing influence, and rewarding uncertainty [30].

According to “Considerations in Assuring Safety of Increasingly Autonomous Systems,” another safety concern is limited oversight, a concern in which “the objective function itself may be correct or at least subject to proper evaluation (for example explicitly consulting a human on a given situation), but it is too expensive to do so frequently, leading to possible hazardous behavior by bad extrapolations from limited samples” [43]. It is important to note that consequences such as negative side effects and reward hacking can be amplified when limited oversight is present. One possible solution to this concern is utilizing semi-supervised RL. In this type of RL, the agent is rewarded for a small number of states or episodes.
Because the specified objective function and the envisioned objective function often differ, it is important to consider SRL strategies to mitigate this concern during the design stage. One SRL strategy, according to García & Fernández, is the introduction of a risk measure to mitigate the risk introduced by the environment’s stochasticity. This risk measure is incorporated through the modification of the objective function. “A Comprehensive Survey on Safe Reinforcement Learning” discusses several approaches to address the modification of the objective function to account for risk: the risk-sensitive criterion, constrained criterion, and the worst-case criterion [34].

In situations without risk, the expectation of the return concerning policy \( \pi \) is defined as:

\[
\max_{\pi \in \Pi} E_\pi(R) = \max_{\pi \in \Pi} E_\pi\left( \sum_{t=0}^{\infty} \gamma^t r_t \right)
\]  

(1)

The concept of risk for a specific system is introduced and shaped during the system definition and safety planning stage and further refined during the system requirements stage through the identification of constraints and unacceptable consequences.

According to García & Fernández, in risk-sensitive criterion, the agent needs to determine the appropriate point on the spectrum between getting a large reward and evading unacceptable consequences even when the probability of occurrence is low. In this safe RL approach, the risk in the objective function is defined as “the variance of the return” or as “the probability of entering an error state.” To control the sensitivity of risk, this approach includes a variable \( \beta \). This variable can be 0, or neutral to risk. The risk can also be positive, when \( \beta > 0 \), which describes avoiding risk. Additionally, the risk can be negative, when \( \beta < 0 \), which describes risky preferences. The researchers have approached the risk-sensitive criterion in two different ways, a risk-sensitive
approach based on exponential functions, in which $R$ represents the cumulative sum of the rewards also known as the return:

$$\max_{\pi \in \Pi} \beta^{-1} \log E_\pi(\exp^{\beta R}) = \max_{\pi \in \Pi} \beta^{-1} \log E_\pi(\exp^{\beta \sum_{i=0}^{\infty} \gamma^i r_i}) \quad (2)$$

or a risk-sensitive approach based on the weighted sum of the return and risk, in which $\omega$ represents the different forms of risk:

$$\max_{\pi \in \Pi} \left( E_\pi(R) - \beta \omega \right) \quad (3)$$

In constrained criterion, the goal is maximum the expected sum of the rewards within specific constraints, $c_i \in C$. This addition of specific constraints results in an extension of the MDP, $< S, A, T, R, C >$. In most cases, this constraint cannot be transgressed, a hard constraint. The expectation of the return is defined in terms of the policy $\pi$, in which $c_i$ is the ith constrain in $C$, $\alpha_i$ is the values boundary, and is $h_i$ represents the return function:

$$\max_{\pi \in \Pi} E_\pi(R) \text{ subject to } c_i \in C, c_i = \{h_i \leq \alpha_i\} \quad (4)$$

In worst-case criterion, the policy is optimized to output in the greatest return for the worst-case scenario. The rationale behind this policy is to reduce the impact of variability and the likelihood of unacceptable situations. This variability is a result of the uncertainty due to randomness in the environment and from the system’s inherent uncertainty. Variability due to randomness in the environment can be represented by the equation below, in which $\Omega^\pi$ is a set of trajectories of the form $(s_0, a_0, s_1, a_1)$. 

58
Variability due to the system’s inherent uncertainty can be represented by the equation below, in which $P$ is an uncertainty set of possible transition matrices; $p$ is the transition model; $E_{\pi,p}(\cdot)$ is the expectation concerning policy $\pi$.

$$\max_{\pi \in \Pi} \min_{w \in \Omega_\pi} E_{\pi,w}(R) = \max_{\pi \in \Pi} \min_{w \in \Omega_\pi} E_{\pi,w}(\sum_{t=0}^{\infty} \gamma^t r_t) \quad (5)$$

$$\max_{\pi \in \Pi} \min_{p \in P} E_{\pi,p}(R) = \max_{\pi \in \Pi} \min_{p \in P} E_{\pi,p}(\sum_{t=0}^{\infty} \gamma^t r_t) \quad (6)$$

Despite the low likelihood of occurrence, worst-case criterion agents optimize the worst-case scenario return [34].

Although these strategies in isolation cannot act as solutions to safety concerns such as reward hacking, avoiding negative side effects, and limited oversight, these approaches can complement the intentional design to mitigate hazards from propagating in complex environments.

### 3.2.6.1.6 Exploration Process Change

The RL agent must balance the need to gain more knowledge about the environment to maximize the return while learning. Exploration and exploitation strategies are techniques used in RL to select actions during the learning phase. Many exploration methods rely on random exploratory components (e.g., $\epsilon$ – greedy) to efficiently explore the state space. However, many of these methods do not take into account the potential safety risks associated with certain actions when learning in the real world.

To address this concern and avoid hazardous situations when learning with hardware, the exploration process may be adapted by incorporating prior knowledge about the task. According
to García & Fernández in “A Comprehensive Survey on Safe Reinforcement Learning”, this can be done through “(i) providing initial knowledge, (ii) deriving a policy from a finite set of demonstrations and, (iii) providing teach advice” [34]. Although incorporating external knowledge such as a human teacher may be the most direct approach to learning a safe policy, it is unrealistic to have a human teacher train an agent over innumerable scenarios.

Another SRL strategy, according to García & Fernández in “A Comprehensive Survey on Safe Reinforcement Learning”, it is the modification of the exploration process that allows for the consideration of risk by modifying the action value [34]. According to Gehring and Precup, one method in the exploration process with risk consideration is defined by the concept of controllability. In controllable states, the action effects are more predictable. Less controllability is described as significant variability in the temporal difference error signal for a specific state [55]. As previously mentioned, this concept of risk for a specific system is introduced and shaped during the system definition and safety planning stage and further refined during the system requirements stage through the identification of constraints and unacceptable consequences. According to Gehring and Precup, the controllability of a state at each time step is defined as:

$$C'(s_t, a_t) \leftarrow C(s_t, a_t) - \alpha'(|\delta_t| + C(s_t, a_t))$$

(7)

Where $\alpha' = \beta \alpha$ is a learning rate and $\delta_t$ is the temporal-difference (TD) error signal. This algorithm picks greedy actions to seek controllable areas of the environment per [55]:

$$Q(s_t, a_t) + wC'(s_t, a_t)$$

(8)

Other researchers such as Law implement in the exploration process a risk measure that considers the effects of the action under uncertainty and actions that may result in a catastrophic negative
event. This risk measure is defined by the weighted sum of the normalized anticipated return of action and the entropy. In RL, entropy describes the outcomes of the system’s stochasticity.

\[ U_r(s, a) = p \times (1 - Risk(s, a)) + (1 - p) \times Q(s, a) \quad (9) \]

The first term measures the risk of an action while the second term is the normalized anticipated long-term reward (i.e., return) of that action [56].

Although these strategies in isolation cannot act as solutions to safety concerns while exploring with hardware, these approaches can complement the intentional design to mitigate hazards from propagating in complex environments.

3.2.7 Model Coding/Building – RL Model Development
In the coding/building stage, the requirements, architecture, and other model elements are used to generate the source code or build the model. In addition, model hyperparameters are identified and implemented to meet model requirements and, if applicable, system/subsystem requirements.

During this stage, there must be awareness of safety-significant design elements in which errors of these safety-significant design elements can compromise safety. Therefore, detailed documentation and proper flow down of all identified safety concerns is necessary.

During this stage, syntax, semantics, and structure techniques for source code and/or building blocks are heavily relied upon to aid in the proper implementation of the design. When these techniques are properly implemented, methods such as code analysis aids in the determination of compliance with safety requirements and proper implementation of the design.

Some of the model coding/building stage objectives include:
• Hyperparameters are selected ensuring compliance with previously identified requirements.

• The RL Model is developed (coded/built) from the architecture, algorithm(s), and hyperparameters.

One of the model coding/building stage outputs includes:

• RL model built

The LOR activities for the above-mentioned objectives and outputs are listed below.

### Table 3.6 Level of rigor activities for model coding/building.

<table>
<thead>
<tr>
<th>Level of Rigor Activity</th>
<th>RL Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model hyperparameters are selected and consistent with the architecture and other model elements, ensuring compliance with the derived requirements.</td>
<td>R R R R</td>
</tr>
<tr>
<td>The model is developed (coded/built) from the hyperparameters and model elements defined in the detailed design stage. This process follows activities described in existing design assurance guidance documents such as DO-178, DO-331, DO-332, and system safety guidance documents such as ISO 262662, MIL-STD-882, etc.</td>
<td>R R R R</td>
</tr>
</tbody>
</table>

**Note:** “R” indicates an activity required for the RLIL.

For a more comprehensive list of LOR activities for software development, object-oriented software development, and formal methods see specific industry standards and guidance documents.

### 3.2.8 Model Training and Evaluation – RL Model Development

At the conclusion of the coding/building stage, model elements, along with hyperparameters, are identified and implemented. However, the model parameters, such as neural network weights and biases, are determined at this stage by iteratively training and evaluating the model.

This stage focuses on finding the optimal parameters by using the training and cross-validation datasets to meet model requirements and, if applicable, system/subsystem requirements. Once the agent has learned a policy by utilizing the training dataset, it is evaluated using the cross-validation
dataset to determine its effectiveness in achieving the goal. This evaluation requires assessing the agent's behavior in different scenarios and measuring its performance within safety constraints. The test dataset must not be utilized for training or optimizing the model to ensure the learning process remains impartial and accurately reflects its actual performance. It is important to note that for some specific RL problems, in place of the datasets, the RL agent may be trained and tested by iteratively interacting with the environment.

The safety of the system is affected by the training process because system behaviors emerge during this process. The goal of this training process is to learn a policy that maximizes a reward function while avoiding hazardous situations, which are mainly caused in RL by model mismatches and inadequate policy [7]. To achieve this goal, several activities need to take place at this stage such as the inclusion of safety risk management metrics, which play a specific role during and after model training. These safety risk management metrics improve the model's safety and performance by aiding developers in identifying inherent uncertainties. Since these metrics cannot act as solutions to all safety concerns, researchers have also proposed other methods to reduce the safety risk such as analyzing the learning algorithm for stability which focuses on factors such as convergence, performance variability, and resistance to perturbations to achieve safety and performance criteria.

Some of the model training and evaluation stage objectives include:

- Parameters are determined using training datasets.

- The model is evaluated and optimized using cross-validation datasets to achieve safety and performance criteria specified in the requirements.

One of the model training and evaluation stage outputs includes:
- Optimized policy

A safety policy is dependent on factors such as environment and task elements, hyperparameters, and parameters that must be fine-tuned during the training and evaluation stage.

The LOR activities for the above-mentioned objectives and outputs are listed below.

Table 3.7 Level of rigor activities for model training and evaluation.

<table>
<thead>
<tr>
<th>Level of Rigor Activity</th>
<th>RL Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety risk management metrics are developed to achieve safety and performance criteria (e.g., accuracy, reliability, and robustness) specified in the requirements. - This could include statistical testing</td>
<td>R R R R</td>
</tr>
<tr>
<td>Training stopping criteria are developed consistent with safety risk management metrics.</td>
<td>R R R R</td>
</tr>
<tr>
<td>The learning algorithm is analyzed for stability focusing on factors such as convergence, performance variability, and resistance to perturbations to achieve safety and performance criteria</td>
<td>R R R R</td>
</tr>
<tr>
<td>Any model optimization that affects the model behavior and occurs after the completion of model training is assessed and properly addressed.</td>
<td>R R R R</td>
</tr>
</tbody>
</table>

Note: “R” indicates an activity required for the RLIL.

3.2.8.1 Guidelines for Model Training and Evaluation

An RL agent can undergo training in two ways: offline during system development or online while the system is operational. A significant drawback of online learning in terms of ensuring functional safety is the inability to develop and evaluate safety cases in advance. Therefore, this research recommends that training of safety-critical systems is performed offline to develop and evaluate safety cases in advance.

During this offline training, if the agent's performance within safety constraints is not satisfactory, the policy should be refined by adjusting the model design elements or modifying the agent's behavior. This process may require going back to previous stages in the RL model development to refine the design stage or change the agent's training strategy.

3.2.9 Model Verification – RL Model Development

Once an optimal policy has been reached, ensuring reasonable system performance within safety constraints, the effort will focus on verification of model requirements. According to some
researchers, one of the main challenges of this stage is to ensure that the trained model can generalize to data that was not utilized during the model training and evaluation stage. Therefore, this stage encompasses a set of activities that serve as proof of the model's capacity to generalize [44]. In addition, there should be a set of activities that serve as proof of the model’s accuracy, reliability, and robustness. This verification is typically conducted throughout the model development (conceptual design stage, data management stage, detailed design stage, etc.).

Verification extends beyond testing. With reviews, analyses, and tests, the objective of verification is to demonstrate the absence of errors and evidence of correct specified requirements satisfaction which is something that testing alone cannot provide. Verification encompasses a variety of methods and tools for assessing the correctness and quality of the model throughout its development.

For testing, a verification dataset produced by the data management stage should be used for evaluating the performance of the trained model. The verification dataset may share similarities with the training dataset, but it also needs to contain safety test cases that have been intentionally derived. To identify safety-specific testing within test cases, developers rely on the previous analyses that identified hazards, causal factors, and their respective requirements from previous stages. These test cases must be reviewed by the developer to ensure that the necessary evidence of risk mitigation validation is included. An error analysis is essential to understand if the model fails to generalize. If the error breaches the performance criteria set by a requirement stage, then it is required to revert to a previous stage such as the data management stage or the training and evaluation stage [44].

Some of the model verification stage objectives include:
• Evidence is provided that the model implementation meets the requirements.

• Traceability is established between model requirements, the implementation, and the verification procedures and results.

One of the model verification stage outputs includes:

• Verified model

Convincing verification results in an ML environment should demonstrate essential characteristics such as comprehensiveness, transparency, and contextual relevancy [44].

The LOR activities for the above-mentioned objectives and outputs are listed below.

Table 3.8 Level of rigor activities for model verification.

<table>
<thead>
<tr>
<th>Level of Rigor Activity</th>
<th>RL Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal and functional requirements-based testing is performed to demonstrate the model’s ability to respond to normal input conditions, verifying that the model complies with accuracy and reliability requirements.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Performance testing is conducted to demonstrate the model's ability to generalize, the ability to function safely in unseen situations by testing with real-world operating conditions.</td>
<td>R R R R</td>
</tr>
<tr>
<td>Robustness testing should be conducted to demonstrate the model's ability to function safely even when exposed to abnormal inputs and events (e.g. edge cases, corner cases, outlier cases, failure conditions).</td>
<td>R R R R</td>
</tr>
<tr>
<td>Robustness testing should be conducted to demonstrate the model's ability to function safely even when exposed to distributional shift (i.e., data distribution during operation differs from the data distribution during training).</td>
<td>R R R R</td>
</tr>
</tbody>
</table>

Note: “R” indicates an activity required for the RLIL.

Other verification LOR activities for all the model development stages are listed below.
### Table 3.9: Level of rigor verification activities for model development.

<table>
<thead>
<tr>
<th>Level of Rigor Objective</th>
<th>Level of Rigor Activity</th>
<th>RL Integrity Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify outputs of conceptual model design.</td>
<td>The ODD is accurately characterized (unambiguous and sufficiently detailed) and consistent (does not conflict with the CONOPS).</td>
<td>R R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are accurate (unambiguous and sufficiently detailed) and consistent (do not conflict with each other).</td>
<td>R R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are derived to ensure model generalization (model’s ability to provide consistent performance during normal conditions on tasks that are similar but not identical to the training task).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are derived to ensure model accuracy (model’s ability to closely predict or closely approximate the true values or optimal actions in a given environment).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are derived to ensure model reliability (model’s ability to execute as required over time intervals, measuring the correctness of the expected use).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are derived to ensure model robustness (model’s ability to provide consistent performance on tasks that are similar to training, but not identical, in the presence of abnormal conditions and data distributional shift).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are derived to ensure model robustness (model’s ability to provide consistent performance on tasks that are similar to training, but not identical, in the presence of data distributional shift).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are verifiable via review, analysis, or testing.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements are compatible with target computer.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirements traceability matrices are reviewed for coverage and completeness.</td>
<td>R R</td>
</tr>
<tr>
<td>Verify outputs of model data management.</td>
<td>Training, cross-validation, and verification datasets are compatible with requirements.</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>Datasets containing synthetic data used to support training and verification are accredited and validated in accordance with standards.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Training and verification datasets are relevant to the intended behavior in the intended operational environment.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Training and verification datasets are complete (samples are distributed across the input domain and subspaces).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Training and verification datasets have balanced features (sufficient amount of each scenario across datasets)</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Training and verification datasets are accurate (samples reflect the operational environment)</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Training and verification datasets are traceable (datasets are properly tagged, labeled, and/or stored).</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Training and verification datasets are synchronized (e.g., winter data being tagged as summer data).</td>
<td>R R</td>
</tr>
<tr>
<td>Verify outputs of detailed model design.</td>
<td>The model logical architecture and other elements are compatible with requirements.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model logical architecture and other elements are consistent, ensuring accurate relationship between model elements.</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>The model logical architecture and other elements conform to standards, including best practices</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The policy is consistent with other model elements.</td>
<td>R R</td>
</tr>
<tr>
<td>Verify outputs of model coding/building.</td>
<td>Verification of outputs is described in existing design assurance guidance documents such as DO-178, DO-331, DO-332, and system safety guidance documents such as ISO 262662, MIL-STD-882, etc.</td>
<td></td>
</tr>
<tr>
<td>Verify outputs of training and evaluation.</td>
<td>Trained model meets safety risk management metrics.</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>Trained model conforms to standards, including best practices.</td>
<td>R R</td>
</tr>
<tr>
<td>Verify outputs of testing.</td>
<td>Test cases and acceptance test criteria identified are consistent with the RLCLs of the model functionality.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Verification procedures have the correct verification cases. Test cases demonstrate that the model satisfies its requirements and verifies that those anomalous conditions or model errors cannot lead to a hazardous condition as identified by the hazard analyses from the System Safety process.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model satisfies the explainability needs (programmatic description of how the model predictions are generated within its functional purpose) to perform verification activities.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Test results are correct and have discrepancies explained.</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>The model generalizes.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model is accurate with respect to requirements.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model is reliable with respect to requirements.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model is robust with respect to requirements.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model is robust to data distributional shift.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>The model is compatible with target computer, selected tests should be performed in the integrated target computer environment.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Requirement coverage analysis is performed to verify the implementation of all requirements.</td>
<td>R R</td>
</tr>
<tr>
<td></td>
<td>Test structural coverage analysis (e.g., neuron coverage, k-multi section neuron coverage, k-neuron coverage, strong neuron activation coverage) should be performed to ensure all model elements were exercised by the requirements.</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>Test non-structural coverage should be achieved, measuring variation in behavior of the model between the distinct inputs fed at testing and at training.</td>
<td>R</td>
</tr>
</tbody>
</table>

**Note:** “R” indicates an activity required for the RLIL.

### 3.2.9.1 Guidelines for Model Verification

#### 3.2.9.1.1 Nominal and Functional Requirements-Based Testing

To establish confidence in the accuracy of the model, precise mathematical specification and verification are required. This necessitates to first identify the specification for the model
requirements. Next, verification via testing must be performed to ensure that the model requirements were achieved. Testing challenges arise as deterministic RL policies may not be perceived as deterministic due to the environment’s nondeterministic responses to the RL algorithm. One challenge in testing such conditions is the difficulty of reproducing specific scenarios. One of the main reasons is that the scenarios may only arise if it receives a precise sequence of inputs from the environment, which makes it challenging to create an accurate test case with the exact conditions required for verification. To address this concern, a systematic analysis process should be implemented with the use of expert knowledge to establish appropriate test cases and acceptance criteria. In addition, according to Koopman and Wagener, to overcome this challenge, researchers employ the use of verification engines [57]. Authors also proposed overcoming this challenge by combining scenarios and safety properties through a tool to create test cases [58].

To establish confidence in the reliability of the model, specification of model requirements should be employed in conjunction with probabilistic programming tools. Probabilistic programming tools such as SCENIC, which establishes soft and hard constraints to generate scenarios, can be employed to accomplish this [59]. Like testing for accuracy, expert knowledge should be utilized to establish appropriate test case and acceptance criteria.

3.2.9.1.2 Performance Testing
Performance tests are comparable in nature to requirements-based tests. However, the emphasis is on the model’s ability to generalize. A challenge associated with testing the model’s ability to generalize is assessing the pass criteria of test results because there is no single correct behavior for a particular safety test case. As a result, according to Koopman and Wagener, determining whether a test has passed or not may require safety envelopes. To ensure confidence that the system
always meets the test pass criteria, several tests may be necessary because passing a test once does not guarantee passing every test [57]. To address this challenge, it is common to use tools capable to perform error analysis, which is essential to understand errors made by the model when working with new data, i.e., failure to generalize. According to researchers, if the error breaches the performance criteria set by a requirement stage, then it is required to revert either to the data management stage or the training stage [44]. However, the developers may opt to also revert to other stages such as the detailed model design stage. In addition to the performance criteria, it is important to place specific emphasis on the safety criteria as well.

Performance testing may not necessarily verify the implementation of requirements. As a result, the nominal and functional requirements-based testing and performance testing approaches complement each other.

3.2.9.1.3 Robustness Testing
Verifying that the model adheres to all the necessary model requirements and is capable of generalization is not sufficient for safety-critical systems. To ensure that the model meets all the required specifications and safety criteria, additional safety testing should be conducted to demonstrate its ability to respond to abnormal input conditions and specific hazardous scenarios. Identification of abnormal input conditions and specific hazardous scenarios requires a systematic process and the aid of tools for identifying faults, failures, and hazardous scenarios. This systematic process is the result of expert knowledge via the system hazard analysis process.

In addition to testing for abnormal input conditions and specific hazardous scenarios, developers should account for data distributional shift. Developers should include test cases where the dataset is from a different distribution than the training dataset. These test cases should evaluate how the
model behaves when encountering data significantly different than its that in its training dataset. These test cases help assess the model's robustness capabilities.

3.2.9.1.4 Test Coverage Analysis
To evaluate and ensure the sufficiency of testing for safety-critical applications, systems that employ deterministic code require test coverage analysis such as requirement coverage analysis and structural coverage analysis (e.g., statement and modified condition/decision coverage). This type of test coverage analysis and resulting metrics has been required by design assurance and safety standards such as ISO26262 [21], DO-178 [27], and JSSSEH [25]. Although there is some debate about the degree to which test coverage guarantees accurate functionality, a high degree of test coverage can enhance users' confidence in the system.

RL implemented in safety-critical applications, just like deterministic code, requires thorough testing to verify the capability of a system to perform consistently and efficiently in diverse situations. Therefore, many researchers have adjusted the traditional test structural coverage methodology to rely on neuron activation values, according to Kim, et al. This methodology involves counting the number of neurons that are activated and meet specific requirements [60]. However, this structural coverage methodology as prescribed in traditional software standards face many challenges when applied to RL. One challenge is the complexity and cost of testing in high-dimensional space, requiring a great volume of test scenarios to ensure sufficient coverage. In addition, traditional structural coverage methodologies lack the necessary level of detail to detect the nuanced behaviors exhibited by RL. Thus, it is uncertain if the metrics for structural coverage can be employed to evaluate the thoroughness of the testing process, providing little insight into the accuracy or comprehensiveness of the knowledge that the network topology and parameters
represent. However, there are indications that implementing structural coverage leads to superior testing outcomes compared to entirely random tests [61].

Because traditional structural methodologies do not fully verify the correctness of the RL policy, researchers have proposed complementing test coverage with a non-structural coverage approach. The test coverage metrics from the non-structural approach provide additional error-revealing capacity [62]. The non-structural approach is correlated to the system behavior in relation to their training data. This approach measures variation in behavior of the RL subsystem between the distinct inputs fed at testing and at training. According to human logic, similar inputs should produce similar behaviors. While some behavior variation is expected, drastic variation may not be satisfactory. Due to issues such as overfitting that lead to inaccurate and partial representation of the environment, fresh and unseen test data is essential. The test coverage analyses, structural coverage and non-structural coverage, need to be complemented with the correct error analysis.

3.2.10 System/Subsystem Implementation - System Development
Once model verification activities have been performed, the effort will focus on producing and loading the model into the target computer, host computer, or target computer emulator, among others for subsystem/system integration.

During this stage, it is important to conduct an analysis that verifies correct integration of the model and compliance with safety requirements. In addition, it is paramount to test and evaluate the agent policy with the actual target platform and subsystem hardware, validating models and simulations against actual hardware. Any inadequate or incorrect inputs identified during this analysis must be returned to the appropriate stage as feedback for appropriate corrective action.

Some of the system/subsystem implementation stage objectives include:
• Model is loaded onto target platform for system/subsystem integration.
• Model compatibility is ensured with target platform, ensuring that no conflicts exist between the model and the hardware/software features of the target platform, especially system response times and input/output hardware.

One of the system/subsystem implementation stage outputs includes:

• Integrated model

For a more comprehensive list of objectives, activities, and outputs see specific industry standards and guidance documents.

3.2.10.1 Guidelines for System/Subsystem Implementation
Given the vast number of test scenarios and data points involved in the model verification, it would be impractical to rerun all the tests on the target platform and subsystem hardware within an appropriate timeframe. As a result, a more efficient approach for implementing testing on the target platform is necessary. This approach can be developed based on the specific characteristics of the model. For example, when complying with relevant model-based development and verification standards, the process should ensure that the model generated for system/subsystem integration is functionally and structurally equivalent to the model used during model verification. Therefore, confidence can be established in the implementation activities without rerunning all model verification testing. However, it is known that numerical computations on varying platforms may lead to different outputs and, as a result, impact the model’s output. This has been seen in models using neural networks. As a result, the implementation activities on the target platform should prioritize confirming the absence of any adverse effects such as alterations in defined model properties or any other adverse effects caused by discrepancies in numerical computations [63].
3.2.11 System Verification and Risk Assessment - System Development

Once the model has been verified and integrated, the effort will focus on system verification of safety requirements and the determination and reporting of the residual safety risk. System verification involves employing a diverse set of reviews, analyses, and tests to ensure reasonable system performance within safety constraints. Reviews and analyses assess the model’s ability to generalize as well as its accuracy, reliability, robustness, completeness, and verifiability of the system requirements and system architecture among others. These reviews and analyses include a technical assessment of the different stages.

In addition, it is necessary to perform system testing in its operational environment to validate models and simulations against actual system hardware. There are multiple reasons to employ testing. These include defect identification, system validation, functionality verification, performance verification, and requirements verification. Due to the complexity of the interactions between the model, the system, and the environment, generic testing by itself does not provide the required evidence that the model can operate safely in the system context without posing undesirable or unacceptable risk. To provide this required evidence of acceptable risk level, testing should comprise of safety-specific testing that verifies model- and system-related hazards are mitigated.

Per the NASA Software Safety Guidebook, there are some testing best practices that should be considered such as, “All test should be traceable to the requirements and all requirements should be tested” [23]. Similar to the system/subsystem implementation, given the vast number of test scenarios and data points involved in the model verification, it would be impractical to rerun all the tests with the system in its operational environment within an appropriate timeframe. To identify safety-specific testing within test cases, developers rely on the previous analyses that
identified hazards, causal factors, and their respective requirements from previous stages. These test cases must be reviewed by the developer to ensure that the necessary evidence of risk mitigation validation is included. Testing should provide assurance that model- and system-derived requirements are correctly specified for the system. This assurance, or validation, occurs throughout the different stages but is formalized with an assurance safety test. With higher levels of SIL, the testing rigor must increase. For higher levels of SIL, or safety-critical systems, testing in a simulated environment is not sufficient because of the difficulty to accurately model the real environment. For these reasons, a method relying on specific testing with the system in its operational environment aids in the final safety assessment.

Some of the system verification and risk assessment stage objectives include:

- An assurance safety test is conducted.
  - The agent policy is evaluated with the system in its operational environment, validating models and simulations against actual hardware.
  - Model training environment and operational environment deltas are identified and assessed for their impact on generalization, accuracy, reliability, and robustness.
  - Training with actual target hardware in its operational environment may be necessary for cases with unsatisfactory results.
- System performance within safety constraints is verified.
  - Evidence (safety artifacts) supporting the safety case is collected.
  - Evidence is provided, assuring that residual risks have been eliminated or minimized as low as reasonably practicable.
Risk assessment is performed.

One of the system verification and risk assessment stage outputs includes:

- Safety case

To generate a safety case that demonstrates a high-level of confidence that the model has a low level of safety risk, evidence of extensive safety analysis is needed for derivation of safety requirements from identified hazards. This process continues collecting evidence of the different system development stages, ensuring safety requirements are satisfied. In addition, developers must ensure hazard data is used for test cases in which safety verification and validation testing is performed. The safety case cannot be completed until a system-level assurance safety test is performed in its target environment or in an environment that mimics the behavior of the operational environment. The safety case is often considered to be the best method for formalizing evidence for complex systems in the context of hazard sources and mitigation approaches. Depending on the SIL level, independent verification and validation by an independent organization are required for the review and assessment of the system development of safety-critical systems.

For a more comprehensive list of objectives, activities, and outputs see specific industry standards and guidance documents.

**3.2.11.1 Guidelines for System Verification and Risk Assessment**

3.2.11.1.1 Assurance Safety Testing
Relying solely on the accumulation of hours in the operational environment to establish the safety confidence on a system that employs RL is not a feasible method of ensuring safety confidence. This approach requires an enormous number of hours to create a convincing statistical argument.
Additionally, the credibility of the data gathered from testing is at risk of being weakened by any changes made to the model, such as security updates. In addition, a system that employs RL must be able to navigate uncommon situations in a safe manner, even though such events are infrequent in regular operation. To address this, the use of simulations in the model verification stage is necessary for safety-critical systems to exercise hazardous scenarios. Simulations with high fidelity testing can be utilized to explicitly confirm the assumptions and simplifications made in lower fidelity testing.

However, there is an inherent risk that the model simulations are not entirely accurate and/or complete. Since RL safety risks measured in simulations and in controlled environments often differ from safety risks in real-world operational settings, an assurance safety test with the actual system in its operational environment is needed to formalize evidence in the context of hazard sources and mitigation approaches. The simulations and controlled environments are used to deliberately create scenarios that are known to be rare while the final assurance safety testing will ensure that the model requirements are achieved in the target environment. This approach can accelerate the evaluation process by focusing on difficult scenarios. An example of this is Waymo, which employs simulated, closed-course testing, and on-road testing to evaluate their autonomous vehicles. If the test with the actual system is not conducted successfully, it is necessary to deploy the learning algorithm and the policy to the target platform to continue learning with the actual target system.

According to Koopman and Wagner, isolating the various testing objectives can also aid in the verification process. This can be accomplished by dividing the verification process into separate procedures for requirements, adequacy of the environmental model, and adequacy of test
scenarios. Isolating the various testing objectives will ensure that the safety-critical system successfully completes a test according to the test objectives [57].

3.2.11.1.2 Safety Case
Currently, industry standards are used to assure safe design of safety-related systems. For example, in the aerospace industry, guidelines like ARP4754A and DO-178C are used, while in the automobile industry, guidelines like ISO26262 and ISO/PAS 21448 are used. However, as industry standards have not been developed or tailored for the inclusion of RL, many researchers suggest that certification should rely on the concept of a safety case [43]. ISO 26262 states that a safety case is an “argument that the safety requirements for an item are complete and satisfied by evidence compiled from work products of the safety activities” [21].

For systems employing deterministic code, the safety case is primarily based on a functional coverage approach and a development coverage approach employed during the software development. The safety case is also based on mitigations imposed on the system as a whole to ensure safe deployment in the operating environment. For systems employing deterministic code, the safety case should instill confidence that the system will perform as intended and will not lead to or permit hazardous outcomes. Evidence supporting the safety case is gathered through safety activities. A list of some of the safety artifacts produced by the activities that support the safety case are shown below:

- Configuration Management Plan (CMP)
- Design Description
- Evidence of Code Level Review
- Evidence of Compliance with Best Practices for Safety-Critical Code Development
- Evidence of Structural Coverage
• Executable Object Code
• Functional Flow to Subsystems and Software Items
• Hazard Analyses (FHA, PHA, SSHA, SHA, FTA, ETA etc.)
• Hazard Tracking Database
• List of Derived and Generic Safety-significant Requirements (GSSR)
• List of Safety-Significant Functions
• Parameter Data Item File
• Plan for Software Aspects of Certification (PSAC)
• Safety (functional) Tread Analysis
• Safety Case/Safety Assessment Report (SAR)
• SIL Table
• Safety Requirements-to-Code Traceability
• Safety Requirements-to-Design Traceability
• Safety Requirements-to-Test Case Trace
• Software Code Standards
• Software Configuration Index
• Software Configuration Management
• Software Configuration Management Records
• Software Configuration Plan
• Software Control Category
• Software Safety Criticality Matrix (SSCM)
• Software Design Artifacts
• Software Design Standard
For systems that employ RL, the system’s safety case argument should also be based on the functional coverage approach and the development coverage approach. These approaches are employed during the model development to instill confidence that the system will perform as intended and will not lead to or permit hazardous outcomes during deployment in the intended environment. To instill confidence in the capacity to handle unpredictable situations and make necessary adjustments, evidence is gathered from different stages such as the ability to detect errors at run-time. While the model should be designed to detect and accommodate such errors at run-
time, researchers have proposed that a “pervasive runtime monitor integrated with the architecture can be an effective means of mitigation” [43]. Just like systems that employ deterministic code, run-time monitoring at the system level further increases the confidence in the safety case.

This safety case should provide evidence that residual risks have been eliminated or minimized as low as reasonably possible. Per the “Artificial Intelligence Risk Management Framework: Second Draft” trying to eliminate all risk may result in impractical expectations and misallocation of resources because avoiding all risk is impossible [46]. Results from the model development will determine the model’s contribution to a mishap risk. Per MIL-STD-882E, the determination of the contribution to a mishap risk requires an analysis of confidence of the verification activities of each safety-significant requirement and function. Both quantitative and qualitative evidence and judgement are required for the risk assessment. “Insufficient evidence or evidence of inadequate software system safety program application should be assessed as risk” [20]. Table 3.1 shows the risk assessment criteria for this research.
Chapter 4: Results

4.1 Results Summary
The safety case addresses a vehicle path-following control. The goal is to have the path-following control functionality commanding the vehicle most of the time, while the remote pilot acts as backup. The vehicle simulation will sense other vehicles’ locations and directions in a three-lane highway setting. The highway setting for this particular safety case is I-64 in the vicinity of Washington University in St. Louis. The system is active when driving and is only deactivated by shutdown or by the user. Also, the system defaults to a safe state (e.g., slow down or stop) if a sensor failure is detected.

The guidelines showcased in this safety case address known unsafe and unknown unsafe hazards through two overlapping processes. While known unsafe scenarios may be deterministically analyzed, unknown unsafe scenarios must be addressed by implementing a rigorous development that brings assurance and integrity to safety significant functions. This approach is based on (1) a hazard analysis process that involves identifying mitigations and (2) the process of LOR activities that involve employing specific measures to ensure the model is developed to a quality standard. These guidelines generate results that are comprehensive, contextually relevant, and comprehensible.

While it is not possible to showcase all the applicable LOR activities identified in the methodology section due to resource constraints in the MATLAB simulated environment, the approach demonstrated in this research exemplifies increasing the safety confidence of the system by lowering the safety risk using the two overlapping processes.
The verification approach of the RL model proposed in this research relies on training the model through an iterative approach. First, the model is trained with a set of baseline LOR activities to build a safety case. Then additional LOR activities are selected to influence the RL model. The model is evaluated based on the increased performance and safety of the system. Safety is measured based on unacceptable events (e.g., collision or transgression of safe distance). To verify the performance of the RL model, a methodology based on scenarios is employed. The distribution of real-world scenarios is characterized by a probabilistic model. Scenarios are randomly selected and evaluated through closed-loop simulations. This process is continuous, selecting more LOR activities and evaluating the performance and safety of the system by ensuring that requirements related to generalization, robustness (anomalies and distributional shift), accuracy, and reliability are satisfied.

4.1.1 System Definition and Safety Planning – System Development

4.1.1.1 System Description
Path-following control is a crucial aspect of this vehicle’s high-level functionality, enabling it to navigate roads seamlessly and safely. The virtual assistant for this use case is for an RL-enabled system to aid in planning driving maneuvers or serve as the primary decision-making source for the vehicle's path-following control operation.

The virtual driving assistant, equipped with RL capabilities, receives various inputs including the vehicle's route plan, local positional information, traffic conditions, and meteorological data. Based on these inputs, it generates a set of rate targets for the path-following control system to execute.

For a vehicle operating in complex and uncertain environments, such as detect-and-avoid and target-tracking functions, this decision-making capability is indispensable.

The decision-making component of the virtual driving assistant leverages navigational data from GPS, local detect-and-avoid sensor data, weather information, and traffic data to determine safe and
efficient maneuvering along the planned route. Safety is of paramount importance, but in an vehicle’s path-following control operation, the efficiency of maneuvering is also valuable. This includes generating energy-aware trajectories, minimizing path length or mission time, and maximizing area coverage, contributing to overall mission improvement.

Furthermore, the decision-making component can incorporate real-time system status data to refine its decisions, ensuring adaptability based on the vehicle's current capabilities and conditions.

RL emerges as a robust solution for this application, as it excels at handling unforeseen scenarios. The trained neural network, serving as the inference model, exhibits low computational requirements, making it efficient for real-time implementation. To explore this application further, a neural network will undergo training with RL to control the path-following control functionality. It will receive communication data and inputs from local detect-and-avoid sensors, ultimately generating commands for the vehicle's path-following control. The goal is to reach a desired destination without colliding with obstacles or deviating from the intended path, ensuring both safety and efficiency.

4.1.1.2 Design Constraints
The vehicle consists of high-level functionality: sense-plan-act. The system includes:

- Lane detection
- Route planning
- Object recognition
- Virtual driving assistant (Path-following control)

The project’s focus is the virtual driving assistant (path-following control). The planning functionality is implemented by an RL end-to-end algorithm that maps sensory inputs directly to the various actuators within the vehicle.
Design Constraints:

- The vehicle is located in a three-lane highway environment.
- There are three to five vehicles on the highway, excluding the agent.
- The agent and the other vehicles shall not come to a full stop.
- Only the agent adjusts driving conditions to avoid collisions and does not interact with other vehicles.
- All vehicles respect highway speed limits, and there is no reckless driving.

Assumptions:

- Operating environment assumptions:
  - The system fully functions under any environmental conditions.
  - There are no obstructions or pedestrians apart from the vehicles defined above.
  - Lane markings are consistent and clearly marked.

- Platform assumptions:
  - Sensors operate properly in all conditions and do not fail (data is reliable).
  - Sensors can accurately detect the environment, and no vehicles on the road are occluded.
  - Actuators operate properly in all conditions and do not fail.
  - The system functionality does not degrade.
  - There are no dormant faults in the system.
  - There are no external faults that propagate from the environment.
  - There is no error propagation in the system.
4.1.1.3 System Safety Objectives
The vehicle operates in a three-lane highway environment. It preserves a safe environment by maintaining longitudinal and lateral safe distance, defined as collision avoidance.

Safe longitudinal distance occurs when vehicles driving in the same direction with one vehicle in front of the other brake and accelerate longitudinally to avoid a collision. Safe lateral distance occurs when vehicles driving in the same direction in parallel lanes use lateral acceleration and braking to avoid collision.

4.1.1.4 Reinforcement Learning Safety Criticality Level(s)
As ISO 26262 severity categories only considered bodily injury [21], severity categories like the ones in MIL-STD-882E would be more appropriate. MIL-STD-882E also considers monetary loss. The rationale for necessitating the inclusion of monetary loss is that damage to the system may be calculated as monetary loss. In addition, MIL-STD-882E provides more detailed definitions. Just as in the severity categorization, MIL-STD-882E provides a valid approach for SCC which is the degree of control that software exercises over hardware [20]. Because this safety case focuses on RL, it is more appropriate to utilize the language Control Category (CC).

However, as the MIL-STD-882E provides guidelines for the defense industry, severity categorization requires tailoring [20]. The main components of severity categorization that are addressed in this standard include environmental impact, bodily injury, and monetary loss. The environmental portion of this categorization will not be assessed in this safety case and will be removed to avoid confusion. Similarly, bodily injury will not be assessed in this safety case and will not be included in the severity categorization. While the original MIL-STD-882E provided monetary loss values that aligned with Department of Defense projects [20], the scope of this safety case will center on vehicles which have lower monetary values. Monetary values have been adjusted to reflect this.
In addition, it must be acknowledged that collisions should be avoided at all costs. Therefore, to achieve a high safety standard, near-collisions are penalized.

Table 4.1 Severity categories.

<table>
<thead>
<tr>
<th>Description</th>
<th>Severity Category</th>
<th>Mishap Result Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catastrophic</td>
<td>1</td>
<td>Collision resulting in loss of system or monetary value equal to or exceeding $50k*.</td>
</tr>
<tr>
<td>Critical</td>
<td>2</td>
<td>Reduction of safety margins, resulting in near** collision.</td>
</tr>
<tr>
<td>Marginal</td>
<td>3</td>
<td>Violation of safe distance, not resulting in near collision.</td>
</tr>
<tr>
<td>Negligible</td>
<td>4</td>
<td>Safety margins maintained with suboptimal*** driving conditions.</td>
</tr>
</tbody>
</table>

*Approximate cost of a new 2022 Tesla, Model 3, including the $10k extra charge for the self-driving feature
**Violation of safe distance in which the agent is less than 1 meter but greater than 0 meters away from the lead vehicle
***Agent decelerates 3.6 meters within 1 second or less

CC require tailoring to account for industry levels of driving automation per SAEJ3016 [41].
### Table 4.2 Control categories.

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Autonomous (AT)</td>
<td>Functionality that exercises control authority over possibly safety-significant hardware systems, subsystems, or components without the opportunity for operator intervention to prevent the occurrence of a mishap or hazard. Complete driving automation (where no operator intervention is anticipated under any conditions).</td>
</tr>
<tr>
<td>2</td>
<td>Semi-Autonomous (SAT)</td>
<td>Functionality that exercises control authority over possibly safety-significant hardware systems, subsystems, or components with the opportunity for safe detection and intervention by an independent mechanism or operator to control the mishap or hazard. There is conditional driving (operator or mechanism must remain aware and ready to take over) automation or high driving automation (driver must take over if required conditions are not met). Functionality that produces data related to safety-significant time-sensitive system operations, allowing the operator the opportunity to verify the information and take appropriate action.</td>
</tr>
<tr>
<td>3</td>
<td>Redundant Fault Tolerant (RFT)</td>
<td>Functionality that exercises redundant, independent fault tolerant control authority over possibly safety-significant hardware systems, subsystems, or components with the opportunity for safe detection and intervention by two or more independent mechanisms or operators to control the mishap or hazard. There is conditional driving (operators or mechanisms must remain aware and ready to take over) automation or high driving automation (operators or mechanisms must take over if required conditions are not met). Functionality that produces data related to safety-significant system operations, allowing an operator to verify the information and take appropriate action.</td>
</tr>
<tr>
<td>4</td>
<td>No Safety Impact (NSI)</td>
<td>Functionality that does not have command or control authority over safety significant systems, subsystems, or components. Functionality that produces data that does not involve safety-significant systems.</td>
</tr>
</tbody>
</table>

### 4.1.1.5 Reinforcement Learning Safety Criticality Matrix

The safety criticality level is used to determine the LOR activities required to be imposed on the path-following control for the integrity and assurance of the functionality. Because this safety case focuses on RL, the SCM derived in the methodology section is utilized. The functionality could also be assessed as having no safety impact, RLIL5.
### REINFORCEMENT LEARNING SAFETY CRITICALITY MATRIX

<table>
<thead>
<tr>
<th>CONTROL CATEGORY</th>
<th>SEVERITY CATEGORY</th>
<th>RLIL 1</th>
<th>RLIL 2</th>
<th>RLIL 3</th>
<th>RLIL 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autonomous</td>
<td>Catastrophic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Critical</td>
<td>RLIL 1</td>
<td>RLIL 2</td>
<td>RLIL 3</td>
<td>RLIL 4</td>
</tr>
<tr>
<td>Semi-Autonomous</td>
<td>Marginal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Negligible</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redundant Fault Tolerant</td>
<td></td>
<td>RLIL 3</td>
<td>RLIL 3</td>
<td>RLIL 4</td>
<td>RLIL 4</td>
</tr>
</tbody>
</table>

### 4.1.1.5 Definition and Evidence of Verification Activities

Verification activities may take place at the system level or the subsystem level. At the system level, verification of a system that relies on RL for some functionality necessitates an assurance safety test to aid in the development of a safety case. The safety case also necessitates other evidence gathered through the different development stages at the system level (e.g., system definition and planning, requirements, design, etc.) and the subsystem level (e.g., conceptual model design, detailed model design, model data management, model coding/building, model training, and evaluation, etc.). This process ascertains a high level of confidence that the system has a low level of safety risk.

The safety case for my research addresses a particular driving situation: a vehicle in a three-lane highway driving scenario in which the vehicle can be surrounded by up to three road vehicles, in which longitudinal and lateral safe distance must be maintained. Part of this safety case utilizes the MATLAB RL toolbox for the simulation. While it is not possible to showcase all applicable verification activities at the system level due to resource constraints in the MATLAB simulated environment, RL model verification is performed to exemplify increasing the safety confidence of the system by lowering the safety risk using RLIL activities.
4.1.2 System Requirements – System Development

4.1.2.1 High-Level Requirements

The CONOPS and hazard analyses such as the PHA and FHA identify high-level requirements. The high-level requirements derived for this safety case are:

Nominal and Functional Requirements:

a) The path-following control function shall maintain a safe separation from obstacles.

b) The path-following control function shall prioritize the requirements according to the following order:
   1. Collision avoidance with other obstacles during merging.
   2. Maintain safe distance from other obstacles during merging.
   3. Reach the merging point.

c) The path-following control system speed shall not fall below the minimum or exceed the maximum speed range defined in the performance requirements.

d) The path-following control system acceleration shall not fall below the minimum or exceed the maximum acceleration range defined in the performance requirements.

e) The path-following control function shall generate steering commands within the allowable range of the vehicle's steering defined in the performance requirements.

f) The path-following control function shall operate at a frequency of 10 Hz.

g) The vehicle shall successfully merge. Successful merging is defined as:
   o Merge is completed without collision.

Performance Requirements:

h) The path-following control system shall not operate below 0.5 m/s.

i) The path-following control system shall not operate above 29 m/s.

j) The path-following control system shall not operate below -4 m/s^2.
k) The path-following control system shall not operate above 2 m/s^2.

l) The path-following control system shall not operate the steering below -15 degrees (-0.2618 rad).

m) The path-following control system shall not operate the steering above 15 degrees (0.2618 rad).

4.1.2.2 Process Requirements
The basic path-following control function shall adhere to safety standards equivalent to the RLIL identified in the FHA.

To identify the RLIL, each hazard associated with the path-following control functionality is assessed for severity consequence and CC. Hazard severity is determined by assessing the worst credible end effect of loss of function or malfunction. The worst credible severity consequence of identified hazards for the path-following control functionality in this safety case is identified as having a “Catastrophic (1)” severity. The CC associated with the path-following control functionality in this safety case is assessed as “Redundant Fault Tolerant (RFT).” The RFT categorization is a result of the vehicle’s inclusion of a safety monitoring function that evaluates the inputs and outputs from the path-following control. In the case of invalid commands, it brings the vehicle into a safe state and initiates the transition to the remote pilot. This safety monitoring function adds an additional safety mechanism, in addition to the remote pilot, to mitigate or control a mishap or hazard.

With this “Catastrophic (1)” rating, the functionality associated with the path-following control is assessed as “Safety-critical,” which is a term associated with Critical or Catastrophic consequences. Because of this "Safety-critical” identification, the level of analysis and verification activities to provide a sufficient level of safety confidence is of paramount importance. The resulting RLIL and its respective LOR have been determined as “RLIL 3.”
Table 4.4 Safety case reinforcement learning safety criticality matrix results.

<table>
<thead>
<tr>
<th>CONTROL CATEGORY</th>
<th>SEVERITY CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Catastrophic (1)</td>
</tr>
<tr>
<td>Autonomous</td>
<td>RLIL 1</td>
</tr>
<tr>
<td>Semi-Autonomous</td>
<td>RLIL 2</td>
</tr>
<tr>
<td>Redundant Fault Tolerant</td>
<td>RLIL 3</td>
</tr>
<tr>
<td></td>
<td>Critical (2)</td>
</tr>
<tr>
<td>Autonomous</td>
<td>RLIL 2</td>
</tr>
<tr>
<td>Semi-Autonomous</td>
<td>RLIL 2</td>
</tr>
<tr>
<td>Redundant Fault Tolerant</td>
<td>RLIL 3</td>
</tr>
<tr>
<td></td>
<td>Marginal (3)</td>
</tr>
<tr>
<td>Autonomous</td>
<td>RLIL 3</td>
</tr>
<tr>
<td>Semi-Autonomous</td>
<td>RLIL 3</td>
</tr>
<tr>
<td>Redundant Fault Tolerant</td>
<td>RLIL 4</td>
</tr>
<tr>
<td></td>
<td>Negligible (4)</td>
</tr>
<tr>
<td>Autonomous</td>
<td>RLIL 4</td>
</tr>
<tr>
<td>Semi-Autonomous</td>
<td>RLIL 4</td>
</tr>
<tr>
<td>Redundant Fault Tolerant</td>
<td>RLIL 4</td>
</tr>
</tbody>
</table>

Table 4.5 Safety case level of rigor tasks results.

<table>
<thead>
<tr>
<th>RLIL</th>
<th>Level of Rigor Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>RLIL 1</td>
<td>Program shall perform analysis of requirements, architecture, design, data, code/model building, and conduct in-depth safety specific testing.</td>
</tr>
<tr>
<td>RLIL 2</td>
<td>Program shall perform analysis of requirements, architecture, design, and data; and conduct in-depth safety-specific testing.</td>
</tr>
<tr>
<td>RLIL 3</td>
<td>Program shall perform analysis of requirements, architecture, and data; and conduct in-depth safety-specific testing.</td>
</tr>
<tr>
<td>RLIL 4</td>
<td>Program shall conduct safety-specific testing.</td>
</tr>
</tbody>
</table>

As this safety case addresses a vehicle’s path-following control, vehicle safety standards such as ISO 26262 should be used as guidelines to identify requirements. The failure rates specified in ISO 26262 depend on the ASIL. The RLIL 3 failure rate for this safety case that corresponds to the ASIL safety goal is 10E-6.

4.1.2.3 Identification of Hazards and Failure Conditions
The hazards and failure conditions associated with the path-following control are identified through hazard analyses such as the PHA and FHA. The hazards and failure conditions considered may include:
• **Lack of Human Intervention**: The path-following control aims to operate without human intervention. However, there may be situations where the subsystem encounters scenarios it cannot handle or where human intervention becomes necessary. If humans are not adequately prepared or able to take control in such situations, it can lead to accidents or unsafe conditions.

• **System Malfunction or Failure**: The path-following control relies on complex software systems, sensors, and actuators. Failures or malfunctions in any of these components can lead to incorrect decisions or loss of control, potentially resulting in accidents or dangerous situations.

• **Sensor Limitations and Environmental Conditions**: The path-following control relies on sensors, such as cameras, LiDAR, radar, and GPS, to perceive their surroundings. Adverse weather conditions like heavy rain, snow, or fog can impact sensor performance, potentially leading to degraded perception and decision-making capabilities.

• **Complex Traffic Scenarios**: Real-world driving environments can be highly complex, with various road users, unpredictable behaviors, and dynamic conditions. Ensuring that the path-following control can handle these complex scenarios safely and reliably is a significant challenge.

For this particular safety case, the primary focus is on the complex traffic scenario of unpredictable vehicle(s) behavior.

• Hazard: Unpredictable vehicle(s) behavior

• Causal Factor: Vehicle(s) behaving unpredictably, such as sudden changes in acceleration.
• Worst Credible Consequence: The vehicle may fail to anticipate or respond appropriately to unpredictable vehicle(s) acceleration, resulting in collision or transgression of safe distance.

• Mitigation: Adapt vehicle behavior to the presence of other vehicle(s), adjusting speed and trajectory to ensure safe interaction and sufficient safe distance(s).

4.1.3 System/Subsystem Design – System Development

4.1.3.1 System/Subsystem Architecture

For this safety case, the path-following control implemented via RL is considered to be part of the vehicle. The goal is to have the path-following control functionality commanding the vehicle most of the time, while the remote pilot acts as backup.

To execute the control commands, a traditional controller is used to directly control the vehicle. In addition, the vehicle includes a safety monitoring function that evaluates the inputs and outputs from the path-following control. In case of invalid commands, it brings the vehicle into a safe state and initiates the transition to the remote pilot. The path-following control utilizes a combination of diverse sensors and communication links, which are integrated to create a comprehensive situational overview. All other components are comprised of standard software and hardware items, including the target computer for a vehicle which is suitable for hard real-time and safety-critical embedded software.

The simulation environment does not encompass the communication links, perception, sensors, and sensor fusion software, as well as their associated simulation models. However, this omission does not hinder the suitability of the simulation. Furthermore, the simulation does not consider the remote pilot functionality, as there is no direct interaction between the path-following control and the remote pilot.
4.1.3.2 Derived Low-Level Requirements
Using the STPA as guidelines, a hazard analysis is performed to derive low-level requirements. STPA requires the identification of unacceptable losses of the specific functions on which the analysis focuses; this may include system loss, property damage, and loss of mission, among others [42]. This safety case focuses on the worst credible loss:

- L-1: Loss of vehicle

Assumption: transgression of safe distance is assessed as lower severity than loss of the vehicle.

System-level hazards are identified by the states or conditions that will lead to the above-mentioned losses.

- H-1: Vehicle violates longitudinal minimum separation safe distance on the highway. [L-1]
- H-2: Vehicle violates lateral minimum separation safe distance on the highway. [L-1]

Vehicle path-following control functionality is assessed as an SSF because the loss of functionality or malfunction results in a mishap.

The following Safety Goals (SG) mitigate or avoid the above-mentioned hazards:

- SG-1: Prevent vehicle from violating longitudinal minimum separation safe distance* on the highway.
- SG-2: Prevent vehicle from violating lateral minimum separation safe distance* on the highway.

The behavior model of the path-following control is constructed to analyze potential hazards by analyzing complex interactions and dependencies implemented by the RL functionality. The safety control structure path-following control is shown below.
The following Control Actions (CA) achieve the above-mentioned SG:

- Provide corrective action (braking) when longitudinal separation safe distance is violated.
- Provide corrective action when the vehicle moves out of the lane when lateral separation safe distance is violated.

Evaluating each CA against the different types of hazardous behaviors identifies UCAs:

- A control action required for safety is not provided.
- An unsafe action is provided.
- A potentially safe control action is provided too early, too late, or out of sequence.
- A safe control action is stopped too soon or applied too long.
### Table 4.6 Undesired control action.

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Undesired Control Actions (Control Action: not provided, provided when not required, provided too early, etc.)</th>
<th>System-level Hazard</th>
<th>Violated Safety Goals</th>
</tr>
</thead>
</table>
| **CA1**: Provide corrective action (braking) when longitudinal separation safe distance is violated | **UCA1**: Control algorithm does not provide braking command when the distance to lead vehicle violates longitudinal minimum safe distance, resulting in collision  
**UCA2**: Control algorithm provides braking command when trailing vehicle violates the minimum longitudinal safe distance, resulting in a collision | H-1 | SG-1 |
| **CA2**: Provide corrective action when the vehicle moves out of lane when lateral separation safe distance is violated | **UCA3**: Control algorithm does not maintain lane position when lateral safe distance is violated, resulting in collision | H-2 | SG-2 |

The above-mentioned UCAs identify the System Constraints:

- **SC-1**: Vehicle must maintain minimum longitudinal separation safe distance on the highway, resulting in minimum separation safe distance violation detection and measures taken to prevent collision. [H-1]
- **SC-2**: Vehicle must maintain minimum longitudinal separation safe distance on the highway, resulting in safe distance violation detection and avoiding unintentional measures to prevent collision. [H-1]
- **SC-3**: Vehicle must maintain minimum lateral separation safe distance on the highway, resulting in minimum separation safe distance violation detection and measures taken to prevent collision. [H-1]
The safety constraints are translated into corresponding safety requirements. The Responsibility-Sensitive Safety (RSS) approach provided in “On a Formal Model of Safe and Scalable Self-Driving Vehicles” offers a framework for deriving safe distance [39]. This framework is presented below.

Scenario 1, UCA1: A longitudinal distance between a vehicle $c_a$ (vehicle) that drives behind another vehicle $c_l$ (lead vehicle), where both vehicles are driving in the same direction, is safe with respect to a response time $t$. If $c_a$ will accelerate by at most $a_{maxlong}$ (vehicle maximum acceleration) during the response time, and from there on will brake by at least $a_{minlong}$ (vehicle minimum braking) until a full stop then it won’t collide with $c_l$.

Using the equations of motion, the corresponding safety constraint for UCA1 can be modified from Shalev et al. (2017). See below:

$$d_{minlong} = [v_{along}t + \frac{1}{2}a_{maxlong}t^2 + \frac{(v_{along} + ta_{maxlong})^2}{2a_{minlong}} - v_l]$$

$v_{along}$ is the longitudinal velocity of the vehicle. $v_l$ is the longitudinal velocity of the lead vehicle, and $d_{minlong}$ is the minimum safe longitudinal distance [39].

Scenario 2, UCA3: A lateral distance between vehicles $c_a$, $c_r$ (vehicle to the right of the vehicle) with $c_a$ having a lateral velocity $v_{alat}$ is safe with respect to parameters $t$, $a_{minlat}$ (minimum lateral acceleration), $a_{maxlat}$ (maximum lateral acceleration), $d_0$ (initial distance), if during the time interval $[0, t]$ the vehicle will apply lateral acceleration of $a_{maxlat}$ towards the right vehicle, and after that the vehicle will apply lateral braking of $a_{minlat}$ until it reaches zero lateral velocity, then the final lateral distance between them will be at least $d_0$ [39].

Using the equations of motion, the corresponding safety constraint for UCA3 can be modified from Shalev et al. (2017). See below:
\[ d_{\text{minlat.}} = d_0 + \left( \frac{v_{\text{lat.}} + v_{\text{lat.},t}}{2} - \frac{v_{\text{lat.},t}^2}{2a_{\text{minlat.}}} \right) \]  

(11)

\( d_{\text{minlat.}} \) is the minimum safe lateral distance [39].

The safety requirements derived above are not duplicate/redundant or contradictory as one requirement refers to lateral safe distance while the other requirement refers to longitudinal safe distance.

4.1.4 Conceptual Model Design – RL Model Development

4.1.4.1 ODD Characterization

The ODD is developed from the CONOPS and later influenced by the hazard analyses such as the Subsystem Hazard Analysis (SSHA) and System Hazard Analysis (SHA).

There are two high-level decision-making actions considered. The two main actions the agent will take are longitudinal acceleration and steering angle. Longitudinal acceleration can be analyzed for hard braking, braking, maintaining velocity, and acceleration. Steering angle can be analyzed for maintaining the lane, changing the lane to the left, and changing the lane to the right. In the case analyzed in this paper, these are the minimal set of actions to attain path-following control functionality (collision avoidance and minimal trip duration). The observations of the agent include:

- Vehicles’ relative longitudinal distance (m)
- Vehicles’ relative lateral distance (m)
- Vehicles’ relative velocity (m)
- Agent vehicle’s yaw angle (rad)
- Relative time gap (s)
- Set velocity (m/s)
- Curvature of the road (rad)
The ODD for the safety case is shown below:

- Road types: urban public multi-lane interstate highway in controlled areas with designated vehicle lanes
- Roadway surface: asphalt
- Roadway geometry: straight
- Roadway users: only vehicles
- Allowable light conditions: day and night, with appropriate lighting conditions for visibility
- Allowable weather conditions: regular weather conditions, excluding extreme conditions such as heavy rain, dense fog, heavy snowfall
- Vehicle type: semi-autonomous car
- Constant supervision by a human operator: no
- Phases of automated operation: normal driving phases, including acceleration, deceleration, maintaining a steady speed, and steering
- Vehicle speed range: 40-60 miles per hour
- Other vehicles in the same area: up to three other vehicles
- Other vehicles' speed range: 40-60 miles per hour
- Connectivity: no connectivity with other vehicles or infrastructure

The hazard analyses are performed based on the characterization of the ODD to identify requirements to eliminate hazards or mitigate their associated safety risk. To mitigate the potential failures of hazards associated with the path-following control, fail-operational and fail-safe mitigations are identified.

Fail-operational mitigations:

- Decreased ODD functionality (e.g., speed, automation, maneuverability, etc.)
Fail-safe mitigations:

- Remote emergency take-over by a trained operator
- Safely moving out of the lane
- Safely stopping in a lane

In addition, the hazard analyses performed identify the following elements as sources of edge cases, corner cases, and outlier cases:

- Maneuver behaviors
- ODD component

Maneuver behaviors pertain to specific control tasks such as following lanes and changing lanes. The ODD component pertains to operating parameters such as speed (5 miles below and above the speed limits).

4.1.4.2 RL Model Requirements

The RL model requirements are derived, compatible with system/subsystem architecture and requirements.

RL Model Requirements on Design:

The objective is to derive requirements that guarantee generalization, accuracy, reliability, and robustness against abnormalities and distributional shifts.

- DE1 - The observations of the agent should encompass other vehicle(s) states.
- DE2 - A reward shall be given to the agent as follows:
  - A negative constant amount if the simulation is terminated early.
  - A constant amount if the lateral error complies with derived requirements.
  - A constant amount if the velocity error complies with derived requirements.
• A proportional amount based on the amount of time the agent has to reach the final point of interest.

• DE3 - A fixed random generator seed shall be set for reproducibility.

• DE4 - The agent shall demonstrate its ability to safely operate in accordance with the National Highway Traffic Safety Administration (NHTSA). Safe operation is defined as:
  - Operation without vehicle collision (defined as zero meters of distance between the agent and the vehicle in front).

• DE5 - The agent shall demonstrate robustness in safe operation in accordance with the NHTSA. The agent demonstrates robustness through edge cases, corner cases, and outliers traced to hazard analyses. Safe operation is defined as:
  - Operation without vehicle collision (defined as zero meters of distance between the agent and the vehicle in front).

• DE6 - The agent shall successfully perform path-following control, including merging, without collision with a success rate of at least 99.999%.

• DE7 - The agent shall successfully perform path-following control, including merging, without a reduction of safety margins with a success rate of at least 99.99%.

• DE8 - The agent shall successfully perform path-following control, including merging, without transgression of safe distance with a success rate of at least 99.9%.

• DE9 - The agent shall not exceed the maximum longitudinal velocity error by 1 m/s.

• DE10 - The agent shall not have a longitudinal acceleration error below – 0.1 or above 0.1 m/s².

• DE11 - The agent shall not have a yaw error below – 0.1 or above 0.1 rad.

• DE12 - The agent shall not have a lateral error below – 0.1 or above 0.1 m.

Requirements traceability is performed to ensure compliance.
Table 4.7 Model requirements.

<table>
<thead>
<tr>
<th>RL Model Requirements on Design</th>
<th>Traceability</th>
<th>Rationale</th>
<th>Safety Tagging</th>
</tr>
</thead>
<tbody>
<tr>
<td>DE1 - The observations of the agent should encompass other vehicle(s) states.</td>
<td>a, b</td>
<td>Rewards play a fundamental role in reinforcement learning by providing feedback to the agent, guiding its learning process, and enabling the optimization of its actions towards desired goals.</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>DE2 - A reward shall be given to the agent as follows:</td>
<td>-</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>A negative constant amount if the simulation is terminated early.</td>
<td>n/a</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>A constant amount if the lateral error complies with derived requirements.</td>
<td>n/a</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>A constant amount if the velocity error complies with derived requirements.</td>
<td>n/a</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>A proportional amount based on the amount of time the agent has reach the final point of interest.</td>
<td>n/a</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>DE3 - A fixed random generator seed shall be set for reproducibility.</td>
<td>-</td>
<td>A fixed random generator is needed in simulations for reinforcement learning to ensure the reproducibility and consistency of experiments.</td>
<td>Not safety</td>
</tr>
<tr>
<td>DE4 - The agent shall demonstrate its ability to safely operate in accordance with the National Highway Traffic Safety Administration (NHTSA). Safe operation is defined as:</td>
<td>-</td>
<td>Generalization is needed to demonstrate successful performance or adaptation to new environments using knowledge of similar situations.</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>Operation without vehicle collision (defined as zero meters of distance between the agent and the vehicle in front)</td>
<td>n/a</td>
<td>-</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>DE5 - The agent shall demonstrate robustness in safe operation in accordance with the National Highway Traffic Safety Administration (NHTSA). The agent demonstrates robustness through edge cases, corner cases, and outliers traced to hazard analyses. Safe operation is defined as:</td>
<td>-</td>
<td>Robustness is needed to ensure proper functionality even in the presence of irregular inputs and situations.</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>Operation without vehicle collision (defined as zero meters of distance between the agent and the vehicle in front)</td>
<td>n/a</td>
<td>-</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>DE6 - The agent shall successfully perform path-following control, including merging, without collision with a success rate of at least 99.999%.</td>
<td>g</td>
<td>-</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>DE7 - The agent shall successfully perform path-following control, including merging, without a reduction of safety margins with a success rate of at least 99.99%.</td>
<td>g</td>
<td>-</td>
<td>Safety Critical</td>
</tr>
<tr>
<td>DE8 - The agent shall successfully perform path-following control, including merging, without transgression of safe distance with a success rate of at least 99.9%.</td>
<td>g</td>
<td>-</td>
<td>Safety Related</td>
</tr>
<tr>
<td>DE9 - The agent shall not exceed the maximum longitudinal velocity error by 1 m/s.</td>
<td>i</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>DE10 - The agent shall not have a longitudinal acceleration error below –0.1 or above 0.1 m/s².</td>
<td>k, j</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>DE11 - The agent shall not have a yaw error below –0.1 or above 0.1 rad.</td>
<td>l, m</td>
<td>-</td>
<td>Not safety</td>
</tr>
<tr>
<td>DE12 - The agent shall not have a lateral error below –0.1 or above 0.1 m.</td>
<td>l, m</td>
<td>-</td>
<td>Not safety</td>
</tr>
</tbody>
</table>
RL Model Requirements on Data:

The objective is to derive requirements that guarantee datasets' relevance, completeness, balance, and accuracy.

- DA1 - Synthetic data shall be used for episode generation of training and verification datasets.
- DA2 - Synthetic data shall be generated independently for training and verification datasets.
- DA3 - The parameters defining the datasets shall be set within the ODD.
- DA4 - The parameters used for the verification dataset shall reflect the real-world scenarios within the ODD.
- DA5 - Hazard analyses shall be used to identify edge cases, corner cases, and outliers within the ODD.
- DA6 - The episode (scenarios) shall end when any of the following occur:
  - The lateral deviation exceeds 1 meter.
  - The velocity of the agent is below 1.1 mph (0.5 m/s).
  - The distance between the agent and the vehicle in front is below zero meters.
Table 4.8 Model requirements safety tagging.

<table>
<thead>
<tr>
<th>RL Model Requirements on Data</th>
<th>Traceability</th>
<th>Rationale</th>
<th>Safety Tagging</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA1 - Synthetic data shall be used for episode generation of training and verification datasets.</td>
<td>-</td>
<td>Since it is not feasible to use an actual vehicle for trial-and-error in this safety case, a completely synthetic environment is used.</td>
<td>Not safety</td>
</tr>
<tr>
<td>DA2 - Synthetic data shall be generated independently for training and verification datasets.</td>
<td>-</td>
<td>Data leakage is prevented by establishing a clear boundary between the data used for training and the data used for evaluation. This is achieved by maintaining a separate set of trajectories specifically designated for evaluation or testing purposes.</td>
<td>Not safety</td>
</tr>
<tr>
<td>DA3 - The parameters defining the datasets shall be set within the ODD.</td>
<td>h, l, j, k, l, m</td>
<td></td>
<td>Not safety</td>
</tr>
<tr>
<td>DA4 - The parameters used for the verification dataset shall reflect the real-world distribution within the ODD.</td>
<td>h, l, j, k, l, m</td>
<td></td>
<td>Not safety</td>
</tr>
<tr>
<td>DA5 - Hazard analyses shall be used to identify edge cases, corner cases, and outliers within the ODD.</td>
<td>-</td>
<td>Hazard analyses are the primary source of edge case identification.</td>
<td>Not safety</td>
</tr>
<tr>
<td>DA6 - The episode (scenarios) shall end when any of the following occur:</td>
<td>h, l, m</td>
<td></td>
<td>Not safety</td>
</tr>
<tr>
<td>The lateral deviation exceeds 1 meter.</td>
<td>n/a</td>
<td></td>
<td>Not safety</td>
</tr>
<tr>
<td>The velocity of the agent is below 1.1 mph (0.5 m/s).</td>
<td>n/a</td>
<td></td>
<td>Not safety</td>
</tr>
<tr>
<td>The distance between the agent and the vehicle in front is below zero meters.</td>
<td>n/a</td>
<td></td>
<td>Not safety</td>
</tr>
</tbody>
</table>

4.1.4.3 RL Environment

The following tools are used for this safety case RL environment:

- RL framework: MATLAB Reinforcement Learning Toolbox™
- DL framework: MATLAB Deep Learning Toolbox™
- Autonomous Driving System simulation framework: MATLAB Automated Driving Toolbox™
- Environment framework: Simulink
- Visual simulation scenario: Unreal Engine® from Epic Games®
- Data management: MATLAB
- Post-processing: Excel
- Operating system: Windows
The environment model includes the agent (a bicycle model) and other vehicles within the scenario and incorporates a model of the agent's physical limitations and dynamics. The agent's physical limitations and dynamics are specified by the following parameters:

- Total vehicle mass (kg)
- Yaw moment of inertia (mNs^2)
- Longitudinal distance from the center of gravity to front tires (m)
- Longitudinal distance from the center of gravity to rear tires (m)
- Cornering stiffness of front tires (N/rad)
- Cornering stiffness of rear tires (N/rad)
- Longitudinal time constant

### 4.1.5 Model Data Management – RL Model Development
RL is the branch of machine learning where an agent interacts with the environment to learn through trial-and-error experiments. Since it is not feasible to use an actual vehicle for trial-and-error in this safety case, a completely synthetic environment is used. Simulations generated from MATLAB, a trusted data source for synthetic data, are created to represent real-world scenarios that are utilized to train the agent. Because the MATLAB environment can generate data until any of the stopping criteria are met, data sufficiency is assumed. The training, cross-validation, and test datasets are generated independently and do not require to be preprocessed or augmented.

#### 4.1.5.1 Training and Cross-Validation Datasets
This model provides the necessary mechanisms to reset the scenario by generating new scenarios randomly, encompassing all aspects of the synthetic environment. The automatic scenario generation represents the entire ODD. However, during the generation of training and cross-validation datasets, the distribution of real-world scenarios may differ from these datasets. Because the datasets are obtained through MATLAB simulations derived from the ODD, the datasets are synchronized
(simulated scenarios operate on a set time step) and traceable (simulated scenarios traced to ODD). The occurrence of a violation of safe distance is crucial to be kept at a minimum to ensure successful learning. This metric and other metrics such as episode number are automatically registered in the Reinforcement Learning Episode Manager. Additionally, the environment model includes other vehicles within the scenario and incorporates a model of vehicle dynamics.

4.1.5.2 Verification Dataset
The goal of the verification dataset is to resemble the real-world environment as closely as possible. The verification dataset utilizes the same automatic scenario generation as described in the training and cross-validation datasets; however, this dataset is independently generated. In addition, safety-specific testing is performed by deterministically identifying parameters to test model generalization, accuracy, reliability, and robustness. Testing parameters are derived from the NHTSA’s “A Framework for Automated Driving System Testable Cases and Scenarios” in addition to other parameters derived from hazard analyses [64]. This process ensures that synthetic data is relevant, complete, balanced, and accurate.

4.1.6 Detailed Model Design – RL Model Development
A data flow and control flow analysis evaluate the intended use of data in RL and ensure the appropriateness of the architecture, objective function, and policy for the application. The data flow analysis helps ensure that the data used in the objective function and policy calculation accurately represents the agent's state, environment dynamics, and desired learning objectives. The control flow analysis is crucial for ensuring that the learning algorithm updates the policy based on the appropriate data and follows the desired training schedule.

4.1.6.1 Model Logical Architecture
Initially, a single submodel that worked for the entire ODD was utilized to generate the next activities of the submodel design process (build/code and train/optimize the RL model). This single submodel was built using a single fully connected feed-forward multi-layer neural network to comply with the
RL requirements. Because the RL model requirements could not be met by a single submodel, a separate submodel for each separate task (i.e., a submodel for lateral control and another submodel for longitudinal control) is implemented.

4.1.6.2 Objective Function
RL requires a reward function or signal which is specified in the objective function to perform learning. The reward function complies with the data requirements for scenario termination. For example, the scenario is terminated when the distance between the agent and the vehicle in front is below zero, resulting in the agent not collecting more rewards. Therefore, the agent will learn to avoid colliding with the leading vehicle. Initially, one objective function was created and implemented for the safety case to maintain safe longitudinal and lateral distance in a three-lane highway driving scenario in which the vehicle can be surrounded by up to three traffic vehicles. Like the model logical architecture, because the RL model requirements could not be met by a single objective function, two separate objective functions are implemented for each task. The equations implemented below are sourced from “Train Multiple Agents for Path Following Control” [65].

For longitudinal control, the reward \( r_t \), provided at every time step \( t \), is the following:

\[
 r_t = -(10e^2_v + 100a^2_{t-1}) \times 0.001 - 10F_t + M_t 
\]

(12)

Here, \( a_{t-1} \) is the acceleration input from the previous time step, and:

- \( F_t = 1 \) if the simulation is terminated early, otherwise \( F_t = 0 \).
- \( M_t = 1 \) if \( e^2_v < 1 \), otherwise \( M_t = 0 \).

For lateral control, the reward \( r_t \), provided at every time step \( t \), is the following:

\[
 r_t = -(100e^2_l + 500u^2_{t-1}) \times 0.001 - 10F_t + 2H_t 
\]

(13)

Here, \( u_{t-1} \) is the steering input from the previous time step, \( a_{t-1} \) is the acceleration input from the previous time step, and:
• $F_t = 1$ if the simulation is terminated early, otherwise $F_t = 0$.

• $H_t = 1$ if $e_t^2 < 1$, otherwise $H_t = 0$.

### 4.1.6.3 Learning Algorithm
Initially, one learning algorithm was implemented for the safety case. Like the model logical architecture and objective function, two separate learning algorithms are implemented for each task, a deep deterministic policy gradient (DDPG) for longitudinal control and a deep Q-network (DQN) for lateral control.

### 4.1.6.4 Policy Type
There are two different types of policies used in RL, deterministic and stochastic policies. For safety-related models, the learned policy must be deterministic. Therefore, both learned policies in this safety case are deterministic.

A logic analysis takes place during this part of the process to identify potential errors in the logical architecture, objective function, and learning algorithm. This analysis is concluded at a later stage.

### 4.1.7 Model Coding/Building – RL Model Development
MATLAB provides a framework to train multiple agents for path-following control. For this safety case, a MATLAB framework was modified to meet the safety case requirements. The resulting model is provided below.

#### 4.1.7.1 Model Coding

##### 4.1.7.1.1 Environment

- $m = 1600$; % total vehicle mass (kg)
- $I_z = 2875$; % yaw moment of inertia (mNs^2)
- $l_f = 1.4$; % longitudinal distance from center of gravity to front tires (m)
- $l_r = 1.6$; % longitudinal distance from center of gravity to rear tires (m)
- $C_f = 19000$; % cornering stiffness of front tires (N/rad)
- $C_r = 33000$; % cornering stiffness of rear tires (N/rad)
- $\tau = 0.5$; % longitudinal time constant
- $x0_{center} = 50$; % initial position for center car (m)
- $y0_{center} = 1$; % initial position for center car on Y (m);
- $\text{yaw}_0_{center} = 0$; % initial position for center car on Yaw (rad);
- $v0_{center} = 24$; % initial velocity for center car (m/s)
x0_ego = 10; % initial position for ego car (m)
v0_ego = 18; % initial velocity for ego car (m/s)
x0_right = 30; % initial position for right car on X (m)
y0_right = -3.5; % initial position for right car on Y (m)
yaw0_right = 0; % initial position for right car on Yaw (rad)
v0_right = 22; % initial velocity for right car (m/s)
x0_left = 35; % initial position for left car on X (m)
y0_left = 5.5; % initial position for left car on Y (m)
yaw0_left = 0; % initial position for left car on Yaw (rad)
v0_left = 23.5; % initial velocity for left car (m/s)

D_default = 10; % default spacing between lead and ego cars (m)
t_gap = 0.75; % time gap for distance maintaining (s)
v_set = 31; % set velocity for ego car (m/s)

amin_ego = -4; % minimum acceleration for ego car (m/s^2)
amax_ego = 2; % maximum acceleration for ego car (m/s^2)
umin_ego = -0.5; % minimum steering angle (rad)
umax_ego = 0.5; % maximum steering angle (rad)

rho = 0.00075; % curvature of road (1/m)
e1_initial = 0.2; % initial lateral deviation (m)
e2_initial = -0.1; % initial yaw angle error (rad)

% Opening Simulink model.
mdl = "rlMultiAgentPFC_visual_3_r2";
open_system(mdl)

% Observation and action specifications for longitudinal control loop.
obsInfo1 = rlNumericSpec([3 1]);
actInfo1 = rlNumericSpec([1 1], LowerLimit=-3, UpperLimit=2);

% Observation and action specifications for lateral control loop.
obsInfo2 = rlNumericSpec([6 1]);
actInfo2 = rlFiniteSetSpec((-15:15)*pi/180);

% Observation and action specifications as a cell array.
obsInfo = {obsInfo1,obsInfo2};
actInfo = {actInfo1,actInfo2};

% Simulink environment interface, specifying the block paths for both agent blocks. The order of the block paths must match the order of the observation and action specification cell arrays.
blks = mdl + ['/RL Agent1', '/RL Agent2'];
env = rlSimulinkEnv(mdl,blks,obsInfo,actInfo);

% Reset function for the environment using the ResetFcn property. The function pfcResetFcn randomly sets the initial poses of the lead and ego vehicles at the beginning of every episode during training.
env.ResetFcn = @pfcResetFcn;
in = setVariable(in,'x0_center',40+randi(60,1,1)); % random value for initial position of center car
in = setVariable(in,'x0_right',40+randi(60,1,1)); % random value for initial position of right car
in = setVariable(in,'x0_left',40+randi(60,1,1)); % random value for initial position of left car
in = setVariable(in,'e1_initial', 0.5*(-1+rand)); % random value for lateral deviation
in = setVariable(in,'e2_initial', 0.1*(-1+2*rand)); % random value for relative yaw angle

4.1.7.1.2 Agents
% Fix the random seed for reproducibility.
rng(0)

% Both agents operate at the same sample time (10 Hz).
Ts = 0.1; % sample time (seconds)

% Longitudinal Control
% Creating a DDPG agent for longitudinal control.
agent1 = createACCAgent(obsInfo1,actInfo1,Ts);

L = 96;
statePath = [
    featureInputLayer(observationInfo.Dimension(1),'Normalization','none','Name','observation')
    fullyConnectedLayer(L,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(L,'Name','fc2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(L,'Name','fc3')
    additionLayer(2,'Name','add')
    reluLayer('Name','relu3')
    fullyConnectedLayer(L,'Name','fc4')
    reluLayer('Name','relu4')
    fullyConnectedLayer(L,'Name','fc5')
    reluLayer('Name','relu5')
    fullyConnectedLayer(L,'Name','fc6')
    reluLayer('Name','relu6')
    fullyConnectedLayer(L,'Name','fc7')
    reluLayer('Name','relu7')
    fullyConnectedLayer(L,'Name','fc8')
    reluLayer('Name','relu8')
    fullyConnectedLayer(L,'Name','fc9')
    reluLayer('Name','relu9')
    fullyConnectedLayer(L,'Name','fc10')
    reluLayer('Name','relu10')
    fullyConnectedLayer(L,'Name','fc11')
    reluLayer('Name','relu11')
    fullyConnectedLayer(L,'Name','fc12')
    reluLayer('Name','relu12')
    fullyConnectedLayer(L,'Name','fc13')
];

actionPath = [
    featureInputLayer(actionInfo.Dimension(1),'Normalization','none','Name','action')
    fullyConnectedLayer(L,'Name','fc14')
    reluLayer('Name','relu14')
    fullyConnectedLayer(L,'Name','fc15')
];

criticNetwork = layerGraph(statePath);
criticNetwork = addLayers(criticNetwork, actionPath);

criticNetwork = connectLayers(criticNetwork,'fc15','add/in2');
% View the critic network configuration.
figure
plot(criticNetwork)

criticOptions = rlRepresentationOptions('LearnRate',1e-3,'GradientThreshold',1);

critic = rlQValueRepresentation(criticNetwork,observationInfo,actionInfo,...
    'Observation',{{'observation'}},'Action',{{'action'}},criticOptions);

actorNetwork = [
    featureInputLayer(observationInfo.Dimension(1),'Normalization','none','Name','observation')
    fullyConnectedLayer(L,'Name','fc1')
    reluLayer('Name','relu1')
    fullyConnectedLayer(L,'Name','fc2')
    reluLayer('Name','relu2')
    fullyConnectedLayer(L,'Name','fc3')
    reluLayer('Name','relu3')
    fullyConnectedLayer(L,'Name','fc4')
    reluLayer('Name','relu4')
    fullyConnectedLayer(L,'Name','fc5')
    reluLayer('Name','relu5')
    fullyConnectedLayer(L,'Name','fc6')
    reluLayer('Name','relu6')
    fullyConnectedLayer(L,'Name','fc7')
    reluLayer('Name','relu7')
    fullyConnectedLayer(L,'Name','fc8')
    reluLayer('Name','relu8')
    fullyConnectedLayer(L,'Name','fc9')
    reluLayer('Name','relu9')
    fullyConnectedLayer(L,'Name','fc10')
    tanhLayer('Name','tanh1')
    scalingLayer('Name','ActorScaling1','Scale',2.5,'Bias','-0.5')];

actorOptions = rlRepresentationOptions('LearnRate',1e-4,'GradientThreshold',1,'L2RegularizationFactor',1e-4);
actor = rlDeterministicActorRepresentation(actorNetwork,observationInfo,actionInfo,...
    'Observation',{{'observation'}},'Action',{{'ActorScaling1'}},actorOptions);

agentOptions = rlDDPGAgentOptions(...
    'SampleTime',Ts,...
    'TargetSmoothFactor',1e-3,...
    'ExperienceBufferLength',1e6,...
    'DiscountFactor',0.99,...
    'MiniBatchSize',64);
agentOptions.NoiseOptions.Variance = 0.6;
agentOptions.NoiseOptions.VarianceDecayRate = 1e-5;

agent = rlDDPGAgent(actor,critic,agentOptions);

% Lateral Control
% Creating a DQN agent for lateral control.
agent2 = createLKAAgent(obsInfo2,actInfo2,Ts);
L = 24; % number of neurons
statePath = [ ]

featureInputLayer(observationInfo.Dimension(1), 'Normalization', 'none', 'Name', 'observation')
    fullyConnectedLayer(L, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    fullyConnectedLayer(L, 'Name', 'fc2')
    additionLayer(2, 'Name', 'add')
    reluLayer('Name', 'relu2')
    fullyConnectedLayer(L, 'Name', 'fc3')
    reluLayer('Name', 'relu3')
    fullyConnectedLayer(L, 'Name', 'fc4')
    reluLayer('Name', 'relu4')
    fullyConnectedLayer(1, 'Name', 'fc5')];

actionPath = [ ]
    featureInputLayer(actionInfo.Dimension(1), 'Normalization', 'none', 'Name', 'action')
    fullyConnectedLayer(L, 'Name', 'fc6')
    reluLayer('Name', 'relu6')
    fullyConnectedLayer(L, 'Name', 'fc7')];

criticNetwork = layerGraph(statePath);
criticNetwork = addLayers(criticNetwork, actionPath);
criticNetwork = connectLayers(criticNetwork, 'fc7', 'add/in2');

% View the critic network configuration.
figure
plot(criticNetwork)

criticOptions = rlRepresentationOptions('LearnRate', 1e-3, 'GradientThreshold', 1);
critic = rlQValueRepresentation(criticNetwork, observationInfo, actionInfo, ...
    'Observation', {'observation'}, 'Action', {'action'}, criticOptions);

agentOptions = rlDQNAgentOptions(...
    'SampleTime', Ts,...
    'UseDoubleDQN', true,...
    'TargetSmoothFactor', 1e-3,...
    'DiscountFactor', 0.99,...
    'ExperienceBufferLength', 1e6,...
    'MiniBatchSize', 64);

agent2 = rlDQNAgent(critic, agentOptions);

4.1.7.1.3 'Training
Tf = 30; % simulation time
maxepisodes = 5000;
maxsteps = ceil(Tf/Ts);
trainingOpts = rlTrainingOptions(...
    MaxEpisodes=maxepisodes,...
    MaxStepsPerEpisode=maxsteps,...
    Verbose=false,...
    Plots="training-progress",...
    StopTrainingCriteria="AverageReward",...
StopTrainingValue=[320,695]);

doTraining = true;
if doTraining
    USE_PRE_TRAINED_MODEL = true;
    if USE_PRE_TRAINED_MODEL
        % Load experiences from pre-trained agent
        load("rlPFCAgents.mat")
        trainingStats = train([agent1,agent2],env,trainingOpts);
    else
        % Train the agent.
        trainingStats = train([agent1,agent2],env,trainingOpts);
    end
else
    % Load pretrained agents for the example.
    load("rlPFCAgents.mat")
end

4.1.7.1.4 Testing
% Robust testing
simOptions = rlSimulationOptions('MaxSteps',maxsteps,'NumSimulations',1000);
experience = sim(env,[agent1, agent2],simOptions);

% Nominal and functional/performance testing, using deterministic initial conditions
e1_initial = -0.4; % initial lateral deviation (m)
e2_initial = 0.1;  % initial yaw angle angle (rad)
x0_center = 80;   % initial position for center car on X (m);
x0_right = 70;    % initial position for right car on X (m);
y0_right = -3.5;  % initial position for right car on Y (m);
x0_left = 30;     % initial position for left car on X (m);
y0_left = 5.5;    % initial position for left car on Y (m);
sim(mdl)         % opening the Simulink model;
4.1.7.2 Model Building

4.1.7.2.1 Path-Following Control Using Reinforcement Learning

Figure 4.2 Path-following control using reinforcement learning.
4.1.7.2.2 Agent Vehicle and Environment

Figure 4.3 Agent vehicle and environment.
Figure 4.4 Agent vehicle.

Figure 4.5 Agent vehicle 3DOF model.
Figure 4.6 Agent sensor.

Figure 4.7 Lead vehicle.
Figure 4.8 Right vehicle.

Figure 4.9 Left vehicle.
4.1.7.2.3 Signal Processing for Longitudinal Control

Figure 4.10 Signal processing for longitudinal control.

Figure 4.11 Velocity error.
Figure 4.12 Safe distance.

Figure 4.13 Simulation termination longitudinal control.

Figure 4.14 Reward function.
4.1.7.2.4 Signal Processing for Lateral Control

Figure 4.15 Signal processing for lateral control.

Reward Function

Figure 4.16 Reward function.
4.1.8 Model Training and Evaluation – RL Model Development
The training is performed, executing the corresponding MATLAB script in the identified MATLAB environment. The training is monitored and evaluated using the Reinforcement Learning Episode Manager shown below.
The reward function is evaluated for stopping criteria as specified below.

Table 4.9 Stopping criteria.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max episodes</td>
<td>5000</td>
</tr>
<tr>
<td>Average Reward for DDPG</td>
<td>320</td>
</tr>
<tr>
<td>Average Reward for DQN</td>
<td>695</td>
</tr>
</tbody>
</table>

4.1.8.1 Optimized Policy
Multiple numbers of hyperparameters are implemented using manual trial and error using values shown below:
Table 4.10 Hyperparameter selection for DDPG - critic.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of neurons</td>
<td>[48, 144]</td>
</tr>
<tr>
<td>Number of hidden layers</td>
<td>[3, 10]</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>[1e-3, 1e-1]</td>
</tr>
<tr>
<td>Activation Function</td>
<td>ReLU</td>
</tr>
<tr>
<td>Gradient Threshold</td>
<td>[0.1, 1]</td>
</tr>
</tbody>
</table>

Table 4.11 Hyperparameter selection for DDPG - actor.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of neurons</td>
<td>[48, 144]</td>
</tr>
<tr>
<td>Number of hidden layers</td>
<td>[3, 10]</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>[1e-4, 1e-1]</td>
</tr>
<tr>
<td>Activation Function</td>
<td>ReLU</td>
</tr>
<tr>
<td>Gradient Threshold</td>
<td>[0.1, 1]</td>
</tr>
<tr>
<td>L2 Regularization Factor</td>
<td>[1e-4, 1e-1]</td>
</tr>
<tr>
<td>Experience Buffer Length</td>
<td>[1e4, 1e6]</td>
</tr>
<tr>
<td>Gamma</td>
<td>[0.9, 1.0]</td>
</tr>
<tr>
<td>Minibatch Size</td>
<td>[64, 256]</td>
</tr>
</tbody>
</table>

Table 4.12 Hyperparameter selection for DQN - critic.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of neurons</td>
<td>[24, 72]</td>
</tr>
<tr>
<td>Number of hidden layers</td>
<td>[3, 10]</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>[1e-3, 1e-1]</td>
</tr>
<tr>
<td>Activation Function</td>
<td>ReLU</td>
</tr>
<tr>
<td>Gradient Threshold</td>
<td>[0.1, 1]</td>
</tr>
<tr>
<td>Experience Buffer Length</td>
<td>[1e4, 1e6]</td>
</tr>
<tr>
<td>Gamma</td>
<td>[0.9, 1.0]</td>
</tr>
<tr>
<td>Minibatch Size</td>
<td>[64, 256]</td>
</tr>
</tbody>
</table>

4.1.9 Model Verification – RL Model Development

4.1.9.1 RL Model Requirements on Data Verification

- DA1 - Synthetic data shall be used for episode generation of training and verification datasets.
  
  - Verification method: inspection
  
  - Means of compliance: The synthetic datasets consist of an agent in a three-lane highway driving scenario in which the agent can be surrounded by up to three traffic vehicles. The training simulation environment is represented below.
DA2 - Synthetic data shall be generated independently for training and verification datasets.

  o Verification method: inspection
  
  o Means of compliance: The verification dataset utilizes the same automatic scenario generation as the training and cross-validation datasets; however, this dataset is independently generated. In addition, safety-specific testing is performed by deterministically identifying parameters to test the model. Testing parameters are derived from the NHTSA’s “A Framework for Automated Driving System Testable Cases and Scenarios” in addition to other parameters derived from hazard analyses [64].

DA3 - The parameters defining the datasets shall be set within the ODD.

  o Verification method: inspection
  
  o Means of compliance: A sample of the parameters defining the datasets is shown below.
\[
m = 1600; \quad \% \text{total vehicle mass (kg)}
\]
\[
I_z = 2875; \quad \% \text{yaw moment of inertia (mNs}^2\text{)}
\]
\[
I_f = 1.4; \quad \% \text{longitudinal distance from center of gravity to front tires (m)}
\]
\[
I_r = 1.6; \quad \% \text{longitudinal distance from center of gravity to rear tires (m)}
\]
\[
C_f = 19000; \quad \% \text{cornering stiffness of front tires (N/rad)}
\]
\[
C_r = 33000; \quad \% \text{cornering stiffness of rear tires (N/rad)}
\]
\[
tau = 0.5; \quad \% \text{longitudinal time constant}
\]
\[
x_0\_center = 50; \quad \% \text{initial position for center car (m)}
\]
\[
y_0\_center = 1; \quad \% \text{initial position for center car on Y (m);}
\]
\[
yaw_0\_center = 0; \quad \% \text{initial position for center car on Yaw (rad);}
\]
\[
v_0\_center = 24; \quad \% \text{initial velocity for center car (m/s)}
\]
\[
x_0\_ego = 10; \quad \% \text{initial position for ego car (m)}
\]
\[
v_0\_ego = 18; \quad \% \text{initial velocity for ego car (m/s)}
\]
\[
x_0\_right = 30; \quad \% \text{initial position for right car on X (m)}
\]
\[
y_0\_right = -3.5; \quad \% \text{initial position for right car on Y (m)}
\]
\[
yaw_0\_right = 0; \quad \% \text{initial position for right car on Yaw (rad)}
\]
\[
v_0\_right = 22; \quad \% \text{initial velocity for right car (m/s)}
\]
\[
x_0\_left = 35; \quad \% \text{initial position for left car on X (m)}
\]
\[
y_0\_left = 5.5; \quad \% \text{initial position for left car on Y (m)}
\]
\[
yaw_0\_left = 0; \quad \% \text{initial position for left car on Yaw (rad)}
\]
\[
v_0\_left = 23.5; \quad \% \text{initial velocity for left car (m/s)}
\]

- **DA4** - The parameters used for the verification dataset shall reflect the real-world scenarios within the ODD.
  - Verification method: inspection
  - Means of compliance: Testing parameters are derived from the NHTSA’s “A Framework for Automated Driving System Testable Cases and Scenarios” in addition to other parameters derived from hazard analyses focused on historical data or credible scenarios [64].

- **DA5** - Hazard analyses shall be used to identify edge cases, corner cases, and outliers within the ODD.
  - Verification method: inspection
  - Means of compliance: the hazard analyses performed identify the following elements as sources of edge cases, corner cases, and outlier cases:
    - Maneuver behaviors: merging
- Edge case: driving velocity at 60 mph (26.8 m/s)
- Corner case: merging at 40 mph (17.9 m/s) and merging at 60 mph (26.8 m/s)
- Outlier case: driving 5 mph (2.5 m/s) below the minimum speed limit, 40 mph (17.9 m/s), and driving 5 mph (2.5 m/s) and driving above the maximum speed limit, 60 mph (26.8 m/s)

- DA6 - The episode (scenarios) shall end when any of the following occur: the lateral deviation exceeds 1 meter; the velocity of the agent is below 0.5 m/s; and the distance between the agent and the vehicle in front is below zero meters.
  - Verification method: inspection
  - Means of compliance: The simulation terminates when the longitudinal velocity of the agent is below 1.1 mph (0.5 m/s) or the distance between the agent and the vehicle in front is below zero meters. The simulation terminates when the magnitude of lateral deviation exceeds 1 meter.

![Diagram](image)

Figure 4.20 Simulation termination longitudinal control.
4.1.9.2 RL Model Requirements on Design Verification

- DE1 - The observations of the agent should encompass other vehicle(s) states.
  
  - Verification method: inspection
  
  - Means of compliance: The synthetic datasets consist of an agent in a three-lane highway driving scenario in which the agent can be surrounded by up to three traffic vehicles. The training simulation environment is represented below.
Figure 4.22 Training simulation.

- DE2 - A reward shall be given to the agent as follows: a negative constant amount if the simulation is terminated early; a constant amount if the lateral error complies with derived requirements; a constant amount if the velocity error complies with derived requirements; and a proportional amount based on the amount of time the agent has reached the final point of interest.
  - Verification method: inspection
  - Means of compliance:
    - For longitudinal control, the reward $r_t$, provided at every time step $t$, is the following:

$$r_t = -(10e_{2V} + 100a_{t-1}^2) \times 0.001 - 10F_t + M_t$$  \hspace{1cm} (14)

Here, $a_{t-1}$ is the acceleration input from the previous time step, and:
- $F_t=1$ if the simulation is terminated early, otherwise $F_t=0$.
- $M_t=1$ if velocity error ($e_{2V} < 1$), otherwise $M_t=0$. 

129
For lateral control, the reward $r_t$, provided at every time step $t$, is the following:

$$r_t=-(100e_i^2+500u_{t-1}^2) \times 0.001-10F_t+2H_t$$  (15)

Here, $u_{t-1}$ is the steering input from the previous time step, $a_{t-1}$ is the acceleration input from the previous time step, and:

- $F_t=1$ if the simulation is terminated, otherwise $F_t=0$.
- $H_t=1$ if lateral error ($e_i^2<1$), otherwise $H_t=0$.

- DE3 - A fixed random generator seed shall be set for reproducibility.
  - Verification method: inspection
  - Means of compliance: the code for random reproduction is shown below.

```matlab
%Reset function for the environment using the ResetFcn property. The function
pfcResetFcn randomly sets the initial poses of the lead and ego vehicles at the
beginning of every episode during training.
env.ResetFcn = @pfcResetFcn;
in = setVariable(in,'x0_center',40+randi(60,1,1)); % random value for initial
position of center car
in = setVariable(in,'x0_right',40+randi(60,1,1)); % random value for initial
position of right car
in = setVariable(in,'x0_left',40+randi(60,1,1)); % random value for initial
position of left car
in = setVariable(in,'e1_initial', 0.5*(-1+2*rand)); % random value for lateral
deviation
in = setVariable(in,'e2_initial', 0.1*(-1+2*rand)); % random value for relative
yaw angle
```

- DE4 - The agent shall demonstrate its ability to safely operate in accordance with the NHTSA. Safe operation is defined as operation without vehicle collision (defined as zero meters of distance between the agent and the vehicle in front).
  - Verification method: testing
  - Means of compliance: perform lane change at low-speed merge, perform lane change at high-speed merge, perform vehicle following at low speed, and perform vehicle following at high speed.
ODD Characteristics

- Road types: urban public multi-lane interstate highway in controlled areas with designated vehicle lanes
- Roadway surface: asphalt
- Roadway geometry: straight
- Roadway users: only vehicles
- Allowable light conditions: daylight
- Allowable weather conditions: sunny weather conditions
- Vehicle type: semi-autonomous car
- Constant supervision by a human operator: no
- Phases of automated operation: normal driving phases, including acceleration, deceleration, maintaining a steady speed, and steering
- Vehicle speed range: 45-55 mph (20.1-24.6 m/s)
- Other vehicles in the same area: up to three other vehicles
- Other vehicles' speed range: 40-60 mph (17.9-26.8 m/s)
- Connectivity: no connectivity with other vehicles or infrastructure

Failure Behaviors

- None

Table 4.13 DE4 testing parameters.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Agent Desired Speed m/s</th>
<th>Center Min. Speed m/s</th>
<th>Center Initial Position m</th>
<th>Right Min. Speed m/s</th>
<th>Right Initial Position m</th>
<th>Left Min. Speed m/s</th>
<th>Left Initial Position m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No merging</td>
<td>20.1</td>
<td>19.8</td>
<td>50</td>
<td>17.9</td>
<td>45</td>
<td>17.9</td>
<td>40</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>21</td>
<td>20</td>
<td>40</td>
<td>19</td>
<td>30</td>
<td>19.5</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>Merging right</td>
<td>17.9</td>
<td>17.9</td>
<td>50</td>
<td>18.5</td>
<td>50</td>
<td>17.9</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>Merging left</td>
<td>18.4</td>
<td>18.2</td>
<td>40</td>
<td>19</td>
<td>30</td>
<td>20.8</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>No merging</td>
<td>22.4</td>
<td>20.2</td>
<td>55</td>
<td>18.4</td>
<td>40</td>
<td>18.5</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>No merging</td>
<td>23.6</td>
<td>20.6</td>
<td>45</td>
<td>20</td>
<td>35</td>
<td>19</td>
<td>35</td>
</tr>
<tr>
<td>7</td>
<td>Merging right</td>
<td>24</td>
<td>17.9</td>
<td>50</td>
<td>20.8</td>
<td>50</td>
<td>17.9</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>Merging left</td>
<td>24.2</td>
<td>19</td>
<td>35</td>
<td>18</td>
<td>30</td>
<td>20.7</td>
<td>50</td>
</tr>
<tr>
<td>9</td>
<td>No merging</td>
<td>24.4</td>
<td>20.6</td>
<td>50</td>
<td>19.4</td>
<td>40</td>
<td>19.8</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>No merging</td>
<td>24.6</td>
<td>20.8</td>
<td>55</td>
<td>20.8</td>
<td>45</td>
<td>18.2</td>
<td>40</td>
</tr>
</tbody>
</table>
Table 4.14 DE4/DE9/DE10/DE11/DE12 testing results.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Collision</th>
<th>Near Collision</th>
<th>Safe Distance Transgression</th>
<th>Suboptimal Driving</th>
<th>Velocity Error Outside Limits</th>
<th>Accel. Error Outside Limits</th>
<th>Yaw Error Outside Limits</th>
<th>Lateral Error Outside Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Merging right</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Merging left</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Merging left</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Merging right</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>Merging left</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 4.23 Sample of not merging – safe and relative distance (DE4).
Figure 4.24 Sample of not merging – longitudinal velocity (DE4).

Figure 4.25 Sample of not merging – acceleration and steering (DE4).
Figure 4.26 Sample of not merging – yaw error (DE4).

Figure 4.27 Sample of not merging – lateral error (DE4).
Figure 4.28 Sample of not merging – lane position (DE4).

Figure 4.29 Sample of merging left – safe and relative distance (DE4).
Figure 4.30 Sample of merging left – longitudinal velocity (DE4).

Figure 4.31 Sample of merging left – acceleration and steering (DE4).
Figure 4.32 Sample of merging left – yaw error (DE4).

Figure 4.33 Sample of merging left – lateral error (DE4).
Figure 4.34 Sample of merging left – lane position (DE4).

Figure 4.35 Sample of endurance testing – relative distance (DE4).
• DE5 - The agent shall demonstrate robustness in safe operation in accordance with the NHTSA. The agent demonstrates robustness through edge cases, corner cases, and outliers traced to hazard analyses. Safe operation is defined as operation without vehicle collision (defined as zero meters of distance between the agent and the vehicle in front).
  o Verification method: testing
  o Means of compliance: edge case following at 40 mph (17.9 m/s) and 60 mph (26.8 m/s); corner case merging at 40 mph (17.9 m/s) and 60 mph (26.8 m/s); outliers following at 65 mph (29.1 m/s) and merging at 65 mph (29.1 m/s)

ODD Characteristics
  o Road types: urban public multi-lane interstate highway in controlled areas with designated vehicle lanes
  o Roadway surface: asphalt
  o Roadway geometry: straight
  o Roadway users: only vehicles
- Allowable light conditions: daylight
- Allowable weather conditions: sunny weather conditions
- Vehicle type: semi-autonomous car
- Constant supervision by a human operator: no
- Phases of automated operation: normal driving phases, including acceleration, deceleration, maintaining a steady speed, and steering
- Vehicle speed range: 35-65 mph (15.6-29.1 m/s)
- Other vehicles in the same area: up to three other vehicles
- Other vehicles' speed range: 35-65 mph (15.6-29.1 m/s)
- Connectivity: no connectivity with other vehicles or infrastructure

Failure Behaviors
- None

Table 4.15 DE5 testing parameters.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Agent Desired Speed m/s</th>
<th>Center Min. Speed m/s</th>
<th>Center Initial Position m</th>
<th>Right Min. Speed m/s</th>
<th>Right Initial Position m</th>
<th>Left Min. Speed m/s</th>
<th>Left Initial Position m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No merging</td>
<td>17.9</td>
<td>19.8</td>
<td>50</td>
<td>18.9</td>
<td>35</td>
<td>15.6</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>26.8</td>
<td>21.8</td>
<td>55</td>
<td>20</td>
<td>30</td>
<td>23.1</td>
<td>40</td>
</tr>
<tr>
<td>3</td>
<td>Merging right</td>
<td>17.9</td>
<td>17.9</td>
<td>50</td>
<td>17.9</td>
<td>40</td>
<td>19.9</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>Merging left</td>
<td>26.8</td>
<td>23</td>
<td>50</td>
<td>23</td>
<td>30</td>
<td>23.1</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>Merging right</td>
<td>29.1</td>
<td>26.8</td>
<td>55</td>
<td>26.8</td>
<td>55</td>
<td>18.7</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>No merging</td>
<td>29.1</td>
<td>23.8</td>
<td>55</td>
<td>21.8</td>
<td>40</td>
<td>18.1</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 4.16 DE5/DE9/DE10/DE11/DE12 testing results.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Collision</th>
<th>Near Collision</th>
<th>Safe Distance Transgression</th>
<th>Subobtimal Driving</th>
<th>Velocity Error Outside Limits</th>
<th>Accel. Error Outside Limits</th>
<th>Yaw Error Outside Limits</th>
<th>Lateral Error Outside Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Merging right</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Merging left</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Merging right</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No merging</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
Figure 4.37 Sample of outlier case – safe and relative distance (DE5).

Figure 4.38 Sample of outlier case – longitudinal velocity (DE5).
Figure 4.39 Sample of outlier case – acceleration and steering (DE5).

Figure 4.40 Sample of outlier case – yaw error (DE5).
Figure 4.41 Sample of outlier case – lateral error (DE5).

Figure 4.42 Sample of outlier case – lane position (DE5).
Figure 4.43 Sample of corner case – safe and relative distance (DE5).

Figure 4.44 Sample of corner case – longitudinal velocity (DE5).
Figure 4.45 Sample of corner case – acceleration and steering (DE5).

Figure 4.46 Sample of corner case – yaw error (DE5).
Figure 4.47 Sample of corner case – lateral error (DE5).

Figure 4.48 Sample of corner case – lane position (DE5).
Figure 4.49 Sample of endurance testing – relative distance (DE5).

Figure 4.50 Sample of endurance testing – safe distance (DE5).
• DE6 - The agent shall successfully perform path-following control, including merging, without collision with a success rate of at least 99.999%.
  
  o Verification method: testing
  
  o Means of compliance: perform the following, including merging, from 40 mph (17.9 m/s) to 60 mph (26.8 m/s).

Table 4.17 DE6 testing parameters and results.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Agent Desired Speed m/s</th>
<th>Number of Cycles</th>
<th>Collision</th>
<th>Number of Violations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Merging</td>
<td>40</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>40</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Merging</td>
<td>42.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>No merging</td>
<td>42.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Merging</td>
<td>45</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>No merging</td>
<td>45</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>Merging</td>
<td>47.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>No merging</td>
<td>47.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Merging</td>
<td>50</td>
<td>10,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>No merging</td>
<td>50</td>
<td>10,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Merging</td>
<td>52.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>No merging</td>
<td>52.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>Merging</td>
<td>55</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>No merging</td>
<td>55</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>Merging</td>
<td>57.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>No merging</td>
<td>57.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>Merging</td>
<td>60</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>No merging</td>
<td>60</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Total # of Cycles</td>
<td>100,000</td>
<td></td>
<td>Violations</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Success Rate</td>
<td>100.00%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 4.51 Sample of reliability testing – relative distance (DE6/DE7/DE8).

Figure 4.52 Sample of reliability testing – safe distance (DE6/DE7/DE8).
• **DE7** - The agent shall successfully perform path-following control, including merging, without a reduction of safety margins with a success rate of at least 99.99%.

  o Verification method: testing

  o Means of compliance: perform the following, including merging, from 40 mph (17.9 m/s) to 60 mph (26.8 m/s).

Table 4.18 DE7 testing parameters and results.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Agent Desired Speed m/s</th>
<th>Number of Cycles</th>
<th>Near Collision</th>
<th>Number of Violations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Merging</td>
<td>40</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>40</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Merging</td>
<td>42.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>No merging</td>
<td>42.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Merging</td>
<td>45</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>No merging</td>
<td>45</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>Merging</td>
<td>47.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>No merging</td>
<td>47.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Merging</td>
<td>50</td>
<td>10,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>No merging</td>
<td>50</td>
<td>10,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Merging</td>
<td>52.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>No merging</td>
<td>52.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>Merging</td>
<td>55</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>No merging</td>
<td>55</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>Merging</td>
<td>57.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>No merging</td>
<td>57.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>Merging</td>
<td>60</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>No merging</td>
<td>60</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
</tbody>
</table>

*Total # of Cycles: 100,000 Violations: 0
Success Rate: 100.00%*

• **DE8** - The agent shall successfully perform path-following control, including merging, without transgression of safe distance with a success rate of at least 99.9%.

  o Verification method: testing
Means of compliance: perform the following, including merging, from 40 mph (17.9 m/s) to 60 mph (26.8 m/s).

Table 4.19 DE8 testing parameters and results.

<table>
<thead>
<tr>
<th>#</th>
<th>Scenario</th>
<th>Agent Desired Speed m/s</th>
<th>Number of Cycles</th>
<th>Safe Distance Transgression</th>
<th>Number of Violations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Merging</td>
<td>40</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>No merging</td>
<td>40</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>Merging</td>
<td>42.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>No merging</td>
<td>42.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>Merging</td>
<td>45</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>No merging</td>
<td>45</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>Merging</td>
<td>47.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>No merging</td>
<td>47.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>Merging</td>
<td>50</td>
<td>10,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>No merging</td>
<td>50</td>
<td>10,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Merging</td>
<td>52.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>No merging</td>
<td>52.5</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>Merging</td>
<td>55</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>No merging</td>
<td>55</td>
<td>5,000</td>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>Merging</td>
<td>57.5</td>
<td>5,000</td>
<td>Yes</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>No merging</td>
<td>57.5</td>
<td>5,000</td>
<td>Yes</td>
<td>88</td>
</tr>
<tr>
<td>17</td>
<td>Merging</td>
<td>60</td>
<td>5,000</td>
<td>Yes</td>
<td>1440</td>
</tr>
<tr>
<td>18</td>
<td>No merging</td>
<td>60</td>
<td>5,000</td>
<td>Yes</td>
<td>1430</td>
</tr>
</tbody>
</table>

Total # of Cycles 100,000 Violations 2979 Success Rate 97.02%

- DE9 - The agent shall not exceed the maximum longitudinal velocity error by 1 m/s.
  - Verification method: testing
  - Means of compliance: DE4 and DE5 parameters

- DE10 - The agent shall not have a longitudinal acceleration error below – 0.1 or above 0.1 m/s$^2$.
  - Verification method: testing
  - Means of compliance: DE4 and DE5 parameters
- DE11 - The agent shall not have a yaw error below – 0.1 or above 0.1 rad.
  - Verification method: testing
  - Means of compliance: DE4 and DE5 parameters

- DE12 - The agent shall not have a lateral error below – 0.1 or above 0.1 m.
  - Verification method: testing
  - Means of compliance: DE4 and DE5 parameters

Once all artifacts have been collected, the model’s safety risk assessment may be performed utilizing the safety risk criteria levels shown below. The determination of the model’s contribution to the system risk requires an analysis of the confidence of the verification activities of each safety-significant requirement and function. Both quantitative and qualitative evidence and judgment are required for the model’s safety risk assessment. Insufficient evidence or evidence of inadequate safety artifacts should be assessed as contribution to the system safety risk.

Table 4.20 Safety case safety risk determination.

<table>
<thead>
<tr>
<th>Risk Levels</th>
<th>Description of Risk Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>High</strong></td>
<td>Loss of functionality or malfunction that upon occurring during normal or credible off-nominal operations or tests:</td>
</tr>
<tr>
<td></td>
<td>- Can lead directly to a catastrophic, or</td>
</tr>
<tr>
<td></td>
<td>- Places the system in a condition where no independent functioning interlocks preclude the potential occurrence of a catastrophic mishap.</td>
</tr>
<tr>
<td><strong>Serious</strong></td>
<td>- Can lead directly to a critical mishap, or</td>
</tr>
<tr>
<td></td>
<td>- Places the system in a condition where only one independent functioning interlock or human action remains to preclude the potential occurrence of a catastrophic mishap.</td>
</tr>
<tr>
<td></td>
<td>- Places the system in a condition where one or no independent functioning interlocks preclude the potential occurrence of a critical mishap.</td>
</tr>
<tr>
<td><strong>Medium</strong></td>
<td>- Can lead directly to a marginal or negligible mishap, or</td>
</tr>
<tr>
<td></td>
<td>- Places the system in a condition where two or more independent functioning interlocks or human action remains to preclude the potential occurrence of a catastrophic mishap.</td>
</tr>
<tr>
<td></td>
<td>- Places the system in a condition where only two or more independent functioning interlocks or human actions remain to preclude the potential occurrence of a critical mishap.</td>
</tr>
<tr>
<td><strong>Low</strong></td>
<td>- Would be a contributor to a marginal or negligible mishap, but two or more independent functioning interlocks or human actions remain.</td>
</tr>
</tbody>
</table>
4.1.10 System/Subsystem Implementation – System Development
The verified RL model may be implemented onto the target platform as a software or firmware item. The software item for this particular safety case includes traditional software developed using a model-based approach in MATLAB, following the DO-178C/DO-331 compliance standards [27][66]. The MATLAB code can generate the C source code from the MATLAB model, which incorporates both RL and traditional software.

One of the objectives of this stage is to ensure model compatibility with the target platform. Given the vast number of test scenarios and data points involved in the model verification, it would be impractical to rerun all the tests on the target platform and subsystem hardware within an appropriate timeframe. When complying with relevant model-based development and verification standards, the process should ensure that the model generated for system/subsystem integration is functionally and structurally equivalent to the model used during model verification. Therefore, confidence can be established in the implementation activities without rerunning all model verification testing. However, it is known that numerical computations on varying platforms may lead to different outputs and, as a result, impact the model’s output. Therefore, a subset of the test dataset that was used during the model verification needs to be retested on the target platform as part of the hardware/software integration.

4.1.11 System Verification & Risk Assessment – System Development
Since RL safety risk measured in a controlled environment often differs from safety risk in operational settings or the real world, continuing the assurance safety test with the actual system in its intended environment is needed. If the test with the actual system in its intended environment is not conducted successfully, it is necessary to deploy the learning algorithm and the policy to the target platform to continue learning with the actual target system.
The data from the assurance safety test and other safety artifacts collected during the life cycle provide evidence that residual safety risk has been eliminated or minimized as low as reasonably practicable. Once all artifacts have been collected, the system safety risk assessment may be performed utilizing system safety risk criteria similar to those in traditional safety standards. The determination of the system’s contribution to a mishap risk requires an analysis of the confidence of the subsystem (model) and system verification activities. Similar to the model’s risk assessment, both quantitative and qualitative evidence and judgment are required for the safety risk assessment. Insufficient evidence or evidence of inadequate safety artifacts should be assessed as system safety risk.
Chapter 5: Discussion and Conclusions

The objective of this research was to ensure high levels of safety confidence in systems employing RL by an approach that utilized safety best practices such as those found in industry-vetted safety guidelines and other novel practices for safe RL model development. The aim was to develop a method not only for developers but also for end-users and acceptance authorities to assess the safety risk associated with RL-based systems and determine their acceptability, providing a metric to measure this acceptability. The research involved developing specific objectives and LOR activities, which could be curated into guidelines for developers or project teams. These guidelines offer recommendations for the rigorous development of RL subsystems, enabling better identification, assessment, and mitigation of safety risks. It is essential to note that these guidelines are not intended to replace traditional system safety standards' LOR activities but rather complement them to enhance the overall system's safety confidence.

These guidelines were showcased in a safety case to address known unsafe and unknown unsafe hazards. For known unsafe scenarios, deterministic analysis was employed, while rigorous development was implemented to ensure safety in unknown unsafe scenarios, ensuring assurance and integrity in safety-critical functions. This approach involved a hazard analysis process to identify mitigations (functional coverage) and LOR activities to ensure high-quality model development (development coverage). The guidelines yielded comprehensive, contextually relevant, and understandable results. The goal was to ensure that the model was robust, accurate, reliable, and able to generalize.

The safety case pertained to a vehicle's path-following control. The vehicle simulation detected other vehicles' positions and directions on a three-lane highway, specifically, I-64 near Washington University in St. Louis.
This paper’s proposed framework for RL composed of functional coverage and development coverage during model development yielded increased safety confidence. The model verification process demonstrated that nearly all requirements related to accuracy, reliability, generalization, and robustness were satisfied.

Nominal and functional requirements-based testing was performed to demonstrate the model’s ability to respond to normal input conditions, verifying that the model complied with accuracy requirements. All the accuracy requirements were successfully met. The accuracy requirements included a maximum longitudinal velocity error of 1 m/s, a longitudinal acceleration error below –0.1 or above 0.1 m/s², a yaw error below –0.1 or above 0.1 rad, and a lateral error below –0.1 or above 0.1 m.

Also, nominal and functional requirements-based testing was performed to demonstrate the model’s ability to respond to normal input conditions, verifying that the model complied with reliability requirements. The reliability requirements were path-following control without collision (catastrophic severity), including merging, at a 99.999% success rate; path-following control without a reduction of safety margins (critical severity), including merging, at a 99.99% success rate; and path-following control without transgression of safe distance (marginal severity), including merging, at a 99.9% success rate. Nearly all of these reliability requirements were met and exceeded. In over 100,000 cycles, not a single collision occurred, resulting in a 100% success rate. In over 100,000 cycles, not a single reduction of safety margins occurred, resulting in a 100% success rate. No transgression of safe distance was maintained for approximately 97.02% of more than 100,000 cycles. This is equivalent to a failure rate of 2.98E-02. This did not meet the 99.9% success criteria and introduced safety risk.

Performance testing was conducted to demonstrate the model’s ability to generalize. This testing assessed the ability of the model to function safely in unseen situations by utilizing the NHTSA’s
“A Framework for Automated Driving System Testable Cases and Scenarios” to simulate real-world operating conditions. All performance tests were passed successfully, including a stress test as part of the performance assessment. While the stress test was passed successfully, it did reveal a safety concern. The success criterion was to operate without vehicle collision. While there were no collisions or near collisions, transgression of safe distance did occur after approximately 30 minutes of operation. In addition, it is worth mentioning that results from nominal and functional requirements-based testing further strengthened the confidence in the model’s ability to generalize.

Robustness testing was performed to demonstrate the model’s ability to function safely even when exposed to abnormal inputs and events. All robustness tests were passed successfully, including a stress test as part of the robustness assessment. No collisions occurred, which was the success criteria of the robustness test. Similar to generalization, results from nominal and functional requirements-based testing further strengthened the confidence in the model’s ability to operate safely even when exposed to abnormal inputs and events.

The results of these assessments helped indicate the contribution to the system-level risk. The simulations during the model verification indicated that the only requirement not met was the transgression of safe distance with a 99.9% success rate in over 100,000 cycles. There was a failure rate of 2.98E-02. This violation is assessed as a “marginal” severity, according to Table 4.1. The CC associated with the path-following control functionality in this safety case is assessed as “Redundant Fault Tolerant” according to Table 4.2. This is a result of the system having two or more independent functioning interlocks to preclude the mishap. With a “marginal” severity and two or more independent functioning interlocks, the model verification results indicate that the contribution to the system-level risk is “low,” according to the safety risk determination developed for this safety case as shown in Table 4.20.
Although the contribution from the RL subsystem to the system-level risk is “low,” it should be noted that this does not necessarily mean the system-level risk is low. These results imply that developers may use this approach to understand the risk and increase the safety confidence in their systems. After completing the system/subsystem integration and system verification, which includes an assurance safety test to evaluate the agent policy with the actual system in its operational environment, a final system safety risk assessment can be conducted. This will allow developers to have sufficient contextual knowledge of the risk associated with the system, which will inform decision-making.

The ability to establish high confidence in RL subsystems unlocks many practical implications for RL implementation. For example, despite the existence of path-following control technology, RL offers many advantages for responding to the unpredictability of real-world driving environments. RL is capable of learning from large driving datasets and adapting to varying operational environments, resulting in generalization to unforeseen scenarios. On the other hand, traditional path-following control technologies often require explicit modeling and have difficulty with unpredictable non-deterministic environments. RL’s ability to make high-level decisions and optimize behavior while prioritizing safety makes it an asset for complementing path-following control functionality, allowing the development of more complex and adaptive autonomous driving systems.
Chapter 6: Future Work

Offering guidelines for developers to increase safety confidence in the RL subsystem, methods for end-users and acceptance authorities to assess the safety risk associated with RL-based subsystems, and a metric to measure the associated risk, this work contributes to the practical implications of achieving high safety confidence in RL subsystems. This work opens several promising directions for future investigations. One way to expand upon this research is to characterize the impacts of each of the LOR activities so that resources may be purposely focused on specific LOR activities that most increase safety confidence. Another avenue of further exploration is the development of specific LOR activities and metrics to build a compelling safety case for online RL.

One key area for future work is characterizing the impact of each LOR activity. It is important to know that not all LOR activities will have the same effectiveness. Some will impact safety confidence more than others, and some activities will pose a greater implementation cost. For future work, it would be beneficial to characterize the impact and limitations of each of the LOR activities so that developers can purposely allocate resources more effectively to achieve a high level of safety confidence in the RL subsystem.

Another focus for future work is the creation of distinct LOR activities and an assessment methodology that utilizes metrics to construct a convincing safety argument for online RL. Although offline RL training can successfully generate a policy before deployment to operate in stochastic environments, online RL has even greater potential due to its real-time adaptability to handle unknown scenarios. This continuous improvement makes online RL attractive to developers. However, a drawback of online learning in terms of ensuring functional safety is the inability to develop and evaluate safety cases in advance. Therefore, online learning should be restricted to non-safety-critical tasks unless a compelling safety case can be built. Moving forward, researchers could examine methods to create a compelling safety case for online RL. This safety case would need to
be based on specific LOR activities that increase the safety confidence of the RL subsystem. The safety case should also be based on a methodology for developers, end-users, and acceptance authorities to assess the safety risk associated with RL-based subsystems and a metric to measure this associated risk.

This work developed guidelines for developers to increase safety confidence in the RL subsystem, methods to assess the safety risk associated with RL-based subsystems, and a metric to measure the associated risk. However, there is still much work to be done in the RL field. Potential directions for future research include the characterization of the impact of each LOR activity to effectively allocate resources to increase safety confidence and developing LOR activities, methodology, and metrics for building a compelling safety case for online RL. In conclusion, this research has provided valuable contributions to the field of RL while also laying the groundwork for further advancements in the field.
References


163


