Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023)
Edited by Mário P. Brito, Terje Aven, Piero Baraldi, Marko Čepin and Enrico Zio
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-18-8071-1_P375-cd

Adversarial Multi-Agent Reinforcement Learning for Fault-Tolerant Design of Complex Systems

Joachim Grimstad

Institute of Automation and software systems, University of Stuttgart, Germany.
E-mail: Joachim.Grimstad@ias.uni-stuttgart.de

Andrey Morozov

Institute of Automation and software systems, University of Stuttgart, Germany.
E-mail: Andrey.Morozov@ias.uni-stuttgart.de

With the increasing complexity of systems, the design and operation of these systems have become increasingly challenging, necessitating greater expertise from engineers and operators. Ensuring the safety and reliability of such systems is crucial, however, traditional design methodologies may not prove adequate to manage the growing complexity of systems. This paper proposes a conceptual approach to designing fault-tolerant complex systems. The approach extends Model-Based System Engineering (MBSE) with zero-sum game models. These models allow Adversarial Multi-Agent Reinforcement Learning (Adv-MARL) techniques to explore various strategies, and assess outcomes. They also have the potential to identify vulnerabilities that can be addressed by iteratively refining the system design.

Keywords: Model-Based System Engineering, Adversarial Multi-Agent Reinforcement Learning, Fault-Tolerant Design, Zero-Sum Game

1. Introduction

Over the past few decades, there has been a rapid increase in the complexity of systems across many industries, driven by the integration of new technologies, and the demand for interoperability and interconnectedness. This tendency is especially noticeable in safety-critical systems, such as autonomous systems or functional safety systems, where intricate networks of components and sensors are essential for the system to function as required. However, as systems become more complex, they also become harder to design and operate, posing significant challenges for engineers and operators. This is further amplified by the need to remain compliant with both existing and emerging regulations and standards. Ensuring the safety and reliability of these systems is crucial, but existing design methodologies may not be sufficient to cope with the growing complexity.

Games have long been used as a tool to develop and demonstrate strategic thinking and problem-solving capabilities. For example, chess has been played for centuries and has been used as a training tool for military strategists, court officials, and rulers (Murray, 1913). It has been shown that games can be an effective way in educational and training contexts to teach problem-solving heuristics (Trinchero and Sala., 2016) and have positive effects on cognitive functioning, including attention, memory, and decision-making (de Aguilera and Mendiz, 2003). Moreover, games provide a safe and controlled environment in which learners can experiment with different strategies and learn from their mistakes without the risk of real-world consequences.

This paper proposes a conceptual approach that extends Model-Based System Engineering (MBSE) with zero-sum game models, where Adversarial Multi-Agent Reinforcement Learning (Adv-MARL) can be applied and evaluated using an existing framework. By integrating the power of MBSE with Adv-MARL, we aim to provide a conceptual framework that enables the development of advanced and resilient systems capable of meeting the demands of modern industries.

2. State of the Art

Games have long been a popular arena for the development and demonstration of AI and Reinforcement Learning (RL) techniques, mainly due to well-defined environments and clear objectives. These factors make it easier to compare the performance of AI agents to human players. A landmark moment in AI history was the 1997 chess match between IBM's Deep Blue and Grandmaster Garry Kasparov, in which a machine defeated a human world champion for the first time in a traditional chess game. However, Deep Blue was extremely resource intensive and highly specialized (Campbell et al., 2002).

In a more recent example, DeepMind's AlphaGo defeated Go world champion Lee Sedol in a five-game match in 2016. AlphaGo utilized an actor-critic RL technique, here value networks were trained to evaluate positions, and a policy network was trained to select moves (Silver et al., 2016). Silver et al. (2018), proposed a general-purpose RL algorithm that learns games solely through self-play. Research has shown that RL algorithms enable agents to learn interactions, coordination, and tool use through self-play in complex games characterized by long and continuous temporal dimensions, partial observability, and vast action and observation spaces (Vinyals et al., 2019; Baker et al., 2019). Furthermore, OpenAI et al. (2019) has demonstrated an agent's ability to retain and transfer learning from one game version to its subsequent iteration.

Despite the potential benefits of using AI in MBSE, there is currently a lack of consistent and well-defined frameworks to enable the application of AI in MBSE (Chami et al., 2022). However, some attempts have been made to use Natural Language Processing (NLP) to automatically generate sysML models (Chami et al., 2019; Chen and Bhada, 2022; Zhong et al., 2023).

3. Concept

This paper proposes to extend the MBSE approach by leveraging the strengths of both MBSE and Adv-MARL to enable the design of exceedingly complex and fault-tolerant systems. To achieve this, the approach incorporates the concept of zero-sum game models, which allows adversarial agents to explore various strategies, assess outcomes, and identify vulnerabilities through self-play that can be addressed by refining the system design. A significant criticism of AI lies in the difficulty of interpreting its decision-making process, which can often be opaque and challenging for humans to comprehend. By adopting a zero-sum game paradigm, this issue can be mitigated by providing a framework for analyzing AI decisions. Games typically showcase clear rules and objectives, are grounded in an established mathematical framework that can be described with game theory, and provides a more intuitive context that resonates more with humans.

The proposed approach can be applied iteratively to meet system requirements effectively, enhancing the design and operation of complex systems. The approach also has the potential to explore system behavior and fault tolerance in a structured and safe environment. The overall framework and its fundamental concepts are depicted in Fig. 1, providing a visual overview of the approach.

3.1. *Model-Based Systems Engineering*

As systems have become increasingly more complex and interconnected, traditional Document-Based Systems Engineering (DBSE) approaches have often proven insufficient and MBSE approaches have become increasingly popular. MBSE is an approach to system engineering that relies on models rather than extensive documentation to capture and convey information about systems from the design and conception phase and throughout the life cycle (Akundi and Lopez, 2021). However, for large, complex multidisciplinary systems, MBSE approaches require substantial time and effort to develop and maintain. in this regard, AI can be a powerful tool by enabling automated model generation, validation, and optimization, or by assisting in the decision-making process by processing large volumes of information.

To present the proposed approach, a simple recurring reference example as proposed by Ruijters and Stoelinga (2015) is introduced. The system

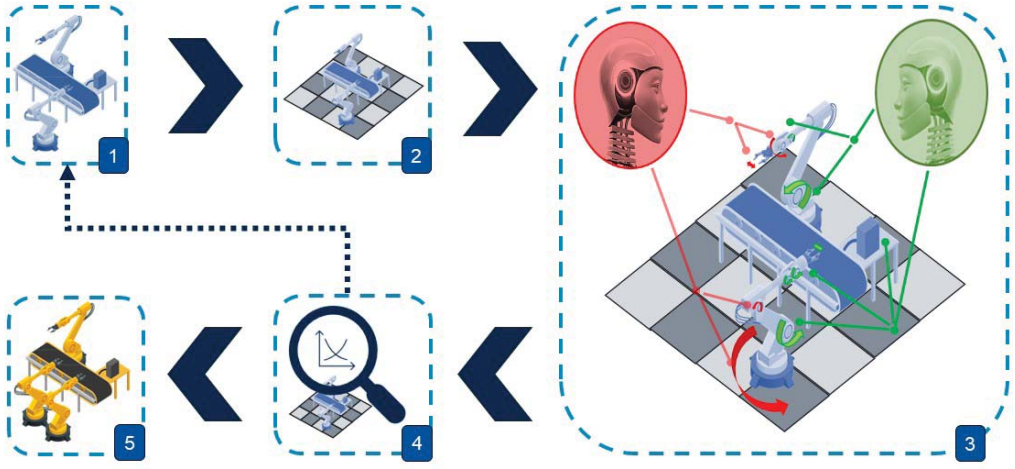

Fig. 1. The workflow of the proposed approach consists of five steps: 1) MBSE approach, 2) Zero-sum Game Design, 3) Adversarial Multi-Agent Reinforcement Learning, 4) Evaluation, and 5) Realization.

is a computer system represented by a single dynamic fault tree as shown in Fig. 2 The system consists of a power supply (PS), two processing units (C_i), and three memory units (M_i). The system is functionally dependent on the power supply and the processing units share the spare memory unit M_3.

3.2. *Game Design*

Designing a zero-sum game that accurately represents the behavior of a complex system while remaining abstract enough to apply the proposed methodology is a significant hurdle that must be overcome in this approach. However, given the recent impressive advances in RL and increased availability of open-source tools, the concept is believed to show promise.

The reference example has been modeled as a simple Agent Environment Cycle (AEC) (Terry et al., 2020). In Multi-Agent Reinforcement Learning (MARL), the most prevalent game model is Partially Observable Stochastic Games (POSGs). Within POSGs, all agents execute actions simultaneously, after which the environment reacts and dispenses rewards to each individual agent. However, in AEC games agents can act sequentially and it can be shown that POSG can be converted to an AEC game and *vice versa* (Terry et al., 2020). The following section outlines several crucial considerations for the proposed approach.

3.2.1. *Time*

The introduction of a temporal dimension is an important aspect from a game, MBSE, and RL perspective as it greatly influences the complexity and difficulty of the proposed methodology. In the case of games, time can act as both a resource and a constraint that directly influences how a game is played. While in MBSE, it may be necessary to consider time-dependent factors or analyze the steady state behavior of a system. For RL algorithms long time horizons are particularly challenging (OpenAI et al., 2019) as it increases the number of cycles per game, making RL exponentially resource intensive.

In the reference example, the temporal dimension is represented as discrete time steps with a fixed interval Δt, such that time is a function of the total number of actions taken $t_n = n\Delta t$. In addition, the time horizon is constrained by introducing a maximum limit of 50 actions per game, which corresponds to the final time step, T.

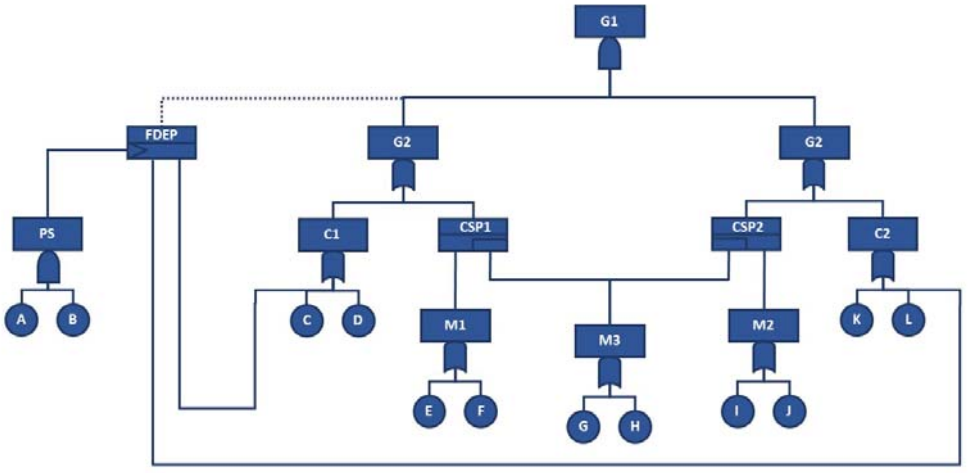

Fig. 2. Recurring example dynamic fault tree, adapted from (Ruijters and Stoelinga, 2015).

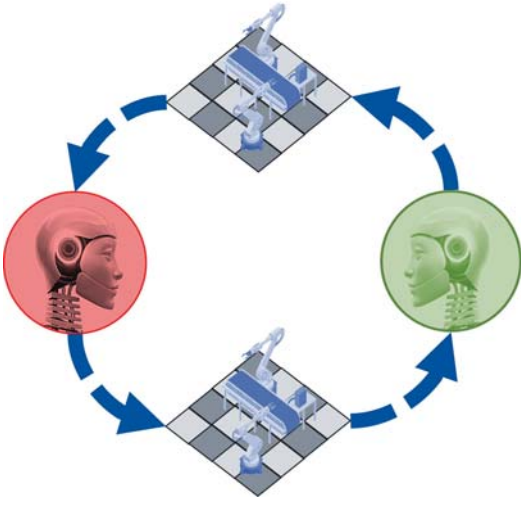

Fig. 3. AEC diagram.

3.2.2. *Observations*

Observations refer to the information an agent receives from the environment or other agents and is used to inform actions taken. Large or partially observable observation spaces also introduce challenges to the proposed methodology. Partial observations force agents to make suppositions (OpenAI et al., 2019) about the environment and adversaries, while large observation spaces complicate the process of identifying relevant observations and relationships.

In the reference example, the observations are represented as a vector $\vec{O}(t_n) = (A, B, C, ..., L)$ where each binary component corresponds to a basic event shown in Fig. 2 and indicates whether the event is active or inactive at the specified time.

3.2.3. *Actions*

Actions refer to the decision made by the agents in response to observations at a specific time. Similarly to observation spaces, large action spaces complicate the training process. In some games, actions may come with a cost of some finite resource, forcing the agent to consider resource management when prescribing actions.

In the reference example, the *red* agents' actions activate basic events, while the *green* agents' actions deactivate them. In other words, *red* agent can be considered a fault instantiator while the *green* acts as a fault mitigator. Actions are also limited by a finite resource, the cost associated with each action, and the current state of the game. For the *red* agent, the cost of activating

basic events is inversely proportional to the basic event probability, while the cost for the *green* agent's action is directly proportional to the cost of restoring the basic event. The basic premise is to limit the *red* agents repeated prescription of low-probability high-consequence events and explore combinations of basic events that would otherwise have been difficult to identify. While the *green* agent is forced to identify and prioritize vulnerable areas of the design. There is also included a latency between when the *green* agent applies an action and when the basic event has been inactivated. The state of the game also influences what actions are possible, for instance, a basic event currently being inactivated, is not a valid action for agents until it is in an active or inactive state.

Thus, the possible actions at a specified time step for each agent i are a function of the state and resources, denoted as $A_i(X(t_n), r)$. Additionally, the state of the system at a future time step is dependent on the previous time step and the chosen action, expressed as $X(t_{n+1}) = f(X(t_n), A_i(t_n))$.

3.2.4. *Rules*

Rules in a game define the set of constraints and conditions that govern the interactions between agents and the environment or win-and-lose conditions for the agents. If agents act sequentially, the order in which they prescribe actions should be clarified, or if actions are prescribed simultaneously, a method for resolving conflicts or ties may be necessary. Termination conditions are essential to define when the game ends. These can include reaching a particular game state, a time limit, or a predetermined number of rounds.

In the reference example, the *red* agent will always start and wins if the top event is activated, while the *green* agent wins if the top event is inactive for the duration of the game.

3.2.5. *Objectives*

In RL, the objective of an agent is typically to maximize a numerical value by learning the relationship between its actions and the associated rewards or penalties. This learning process is guided by a specific algorithm and defined reward functions. Balancing exploration and exploitation is a critical aspect of RL, as agents must decide when to explore the prescription of new actions to discover potentially higher rewards or exploit known actions with expected high rewards (Ishii et al., 2002). Conceptually, this balance can be likened to a Depth-First Search or Breadth-First Search in the exploration of an optimal solution.

In the reference example, the *red* agent is rewarded when the top event is activated and the game concludes. The *green* agent gets rewarded for each cycle the top event is inactive. If an agent wins, the remaining resources get added as a reward.

3.3. *Training*

The training process is essential for enabling agents to learn the connections between prescribed actions, observations, and rewards thus developing their decision-making capabilities. During the training phase, agents interact with the environment by prescribing actions and observing the resulting observations and rewards in a cycle as shown in Fig. 3. The exact learning process is dependent on the algorithm and hyperparameters. A number of different techniques can be used to improve the training process such as batching (Hafner et al., 2017), curriculum learning (Wang et al., 2022), or reward shaping (Hu et al., 2020).

4. Discussion

Evaluation of the proposed approach is highly dependent on implementation and will require an extensive and collaborative effort to demonstrate its effectiveness compared to existing design methodologies and advance the state of the art. As such, this evaluation is limited to a qualitative assessment of its potential benefits.

The proposed approach has the potential to address some of the challenges presented by increasingly complex systems. The inclusion of a zero-sum game paradigm allows for effective, safe, and controlled exploration of system fault tolerance and behavior, potentially leading to more robust, efficient, and reliable system designs. The zero-sum game paradigm also increases transparency and provides a framework for the evaluation of

game outcomes for trained agents. Such as determining if the design is biased towards specific agent objectives using established game theory principles such as the identification of Nash-equilibrium points. The approach might also synergize with the use of AI techniques in other areas of MBSE, potentially enhancing various aspects of the system engineering process and further improving the overall design and operation.

Although the proposed approach may have significant potential, it is important to note that it also has limitations. Similar to models of real-world systems, the zero-sum game paradigm employed in this approach is an abstraction that may not capture the complete system behavior. Additionally, while the use of a zero-sum game paradigm may increase transparency, the inner workings of an agent's trained model remain poorly understood. The Adv-MARL approach may also be more suited to evaluate undesired events caused by a malicious third party than stochastic undesired events.

5. Further Work

The proposed approach presents numerous interesting opportunities for future work. One avenue is an automated solution to translate large sets of models representing a complex system into a zero-sum game. Evaluation and redesign of the system based on game outcomes could also be a direction. Input models, game design, learning algorithms, and neural network architectures are assumed to be highly correlated with applicability and are another interesting direction. Additionally, investigating more complex input models with larger observation and action spaces and expanding the method's applicability to a wider range of system design problems or MBSE-related topics.

Acknowledgement

The authors would like to thank the Ministry of Science, Research and Arts of the Federal State of Baden-Württemberg for the financial support of the projects within the InnovationCampus Future Mobility (ICM).

References

Akundi, A. and V. Lopez (2021). A review on application of model based systems engineering to manufacturing and production engineering systems. *Procedia Computer Science 185*, 101–108. Big Data, IoT, and AI for a Smarter Future.

Baker, B., I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch (2019). Emergent tool use from multi-agent autocurricula.

Campbell, M., A. Hoane, and F. hsiung Hsu (2002). Deep blue. *Artificial Intelligence 134*(1), 57–83.

Chami, M., N. Abdoun, and J.-M. Bruel (2022). Artificial intelligence capabilities for effective model-based systems engineering: A vision paper. *INCOSE International Symposium 32*(1), 1160–1174.

Chami, M., C. Zoghbi, and J.-M. Bruel (2019). A first step towards ai for mbse: Generating a part of sysml models from text using ai. *A First Step towards AI*.

Chen, M. and S. V. Bhada (2022). Converting natural language policy article into mbse model. In *INCOSE International Symposium*, Volume 32, pp. 73–81. Wiley Online Library.

de Aguilera, M. and A. Mendiz (2003, oct). Video games and education: (education in the face of a "parallel school"). *Comput. Entertain. 1*(1).

Hafner, D., J. Davidson, and V. Vanhoucke (2017). Tensorflow agents: Efficient batched reinforcement learning in tensorflow. *CoRR abs/1709.02878*.

Hu, Y., W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan (2020). Learning to utilize shaping rewards: A new approach of reward shaping. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems*, Volume 33, pp. 15931–15941. Curran Associates, Inc.

Ishii, S., W. Yoshida, and J. Yoshimoto (2002). Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural Networks 15*(4), 665–687.

Murray, H. J. R. (1913). *A History of Chess* (1st ed.). London: Humphrey Milford.

OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang (2019). Dota 2 with large scale deep reinforcement learning.

Ruijters, E. and M. Stoelinga (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review 15-16*, 29–62.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis (2016, January). Mas-

tering the game of go with deep neural networks and tree search. *Nature 529*, 484–489.

Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science 362*(6419), 1140–1144.

Terry, J. K., B. Black, A. Hari, L. S. Santos, C. Dieffendahl, N. L. Williams, Y. Lokesh, C. Horsch, and P. Ravi (2020). Pettingzoo: Gym for multi-agent reinforcement learning. *CoRR abs/2009.14471*.

Terry, J. K., N. Grammel, B. Black, A. Hari, C. Horsch, and L. S. Santos (2020). Agent environment cycle games. *CoRR abs/2009.13051*.

Trinchero, R. and G. Sala. (2016). Chess training and mathematical problem-solving: The role of teaching heuristics in transfer of learning. *Eurasia Journal of Mathematics, Science and Technology Education 12*(3), 655–668.

Vinyals, O., I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature 575*(7782), 350–354.

Wang, X., Y. Chen, and W. Zhu (2022). A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*(9), 4555–4576.

Zhong, S., A. Scarinci, and A. Cicirello (2023). Natural language processing for systems engineering: Automatic generation of systems modelling language diagrams. *Knowledge-Based Systems 259*, 110071.