ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 11, No. 6, 18-27 (2026)
www.astesj.com
Special Issue on Emerging Multidisciplinary Directions in Engineering, Computing, and Applied Sciences

ASTES Journal
ISSN: 2415-6698

A Multi-Perspective Evaluation of Assistance Methods for Specifying Requirements for Reinforcement Learning

Wataru Mori*, Fumiko Harada, Hiromitsu Shimakawa

Graduate School of Information Science and Engineering, Ritsumeikan University, Ibaraki, Osaka ,567-8570, Japan

Email(s): is0639xr@ed.ritsumei.ac.jp (W. Mori), harada@fc.ritsumei.ac.jp (F. Harada), simakawa@cs.ritsumei.ac.jp (H. Shimakawa)

*Corresponding Author: Wataru Mori, Ritsumeikan University, 2-150 Iwakura-cho, Ibaraki, Osaka 567-8570, Japan, Contact No: +81-70-2304-4615 & Email: is0639xr@ed.ritsumei.ac.jp

ARTICLE INFO

Article history:
Received: 13 September, 2026
Revised: 10 November, 2026
Accepted: 14 November, 2026
Online: 23 November, 2026

Keywords:
Reinforcement Learning
State Transition Diagram
Requirement

ABSTRACT

This study proposes a method to assist in the creation of requirement documents for high-quality state transition diagrams, with the aim of simplifying the creation of reinforcement learning environments. It provides templates and precautions to assist inexperienced engineers in refining their requirements. These requirements can be semi-automatically converted into state transition diagrams. The effectiveness of various assistive elements is evaluated by comparing edit distances pre and post-modification. The method also applies the same assistance to LLMs to analyze differences from human performance. Experiments on specifying the card game "Old Maid" confirm the effectiveness of model requirements and precautions. The results reveal the effectiveness of model documents and precautions, but no significant effect with the precaution position. The results showed that model documents and writing precautions reduced textual edit distances, while the position of precautions had no statistically significant effect. LLM-generated requirements also showed relatively low edit distances, although their quality requires further evaluation.

1. Introduction

More and more studies in recent years have tried to achieve goals stably even under conditions where rapid changes would occur [1, 2, 3]. Their typical example is the autonomous driving technology. The study refers to the environment as a "non-stationary environment" hereafter [4]. Reinforcement learning is often used to solve problems in such non-stationary environments [5, 6]. To learn an optimal policy, reinforcement learning requires a simulation environment of the system it tries to control. However, it costs considerable effort because it involves modeling the target task.

Since many existing methods assume that the environment is always in a steady state, it is more difficult to create simulation environments under non-steady-state environment. To solve the problem, we can use a state transition diagram, but it is difficult for inexperienced users to create one.

As one possible approach, let us consider to create a requirement that describes, in natural language, the transitions that occur when specific events take place in each state of the system under control. A good state transition diagram created from a good requirements document leads to a good simulated environment. The assistance of writing good requirements would contribute to the generation of reinforcement learning environments.

The study discusses what kind of advice can help inexperienced users improve the quality of their requirements. It compares the proposed approach with cases presented by LLMs. In this method, participants are asked to write a requirement for a given task. Within the requirement, the method provides participants with several different types of advice. The study investigates which improves the quality of requirements: providing each type of advice individually or their combination. In addition, LLMs such as ChatGPT are treated as virtual participants to make them generate requirements documents for the task, given the same information as the human participants. The comparison of the LLM-generated documents with those written by human participants enables us to analyze the differences in requirements-writing ability between LLMs and humans. As a result, providing model document and writing hints finds to assist the generation of requirements. Furthermore, it is suggested that LLM could produce excellent results in tasks with extensive corpus.

An earlier version of this study was presented at the 2025 International Mobile, Intelligent, and Ubiquitous Computing Confer-

ence (MIUCC) [7]. The conference paper evaluated the effects of model documents and the presentation position of writing precautions using normalized Levenshtein edit distance. It also compared requirements written by human participants with those generated by LLMs.

This journal version extends the previous study by providing an additional analysis of the effectiveness of writing precautions. Specifically, it compares the normalized edit distances of requirement documents created with and without these precautions to examine whether explicit writing guidance helps reduce the amount of textual modification required during expert refinement.

However, the evaluation in this study primarily focuses on textual edit distance, which does not directly measure the correctness or completeness of generated state transition diagrams. Therefore, future work will directly evaluate the quality of state transition diagrams generated from the requirement documents.

2. State Transition Diagram

2.1. State Transition Diagram and Requirement

Efficient reinforcement learning requires defining the agent's actions and properties. Additionally, it is necessary to generate an environment that simulates the rewards the agent receives based on those actions. It is indispensable for a good state transition diagram to make the generation of this simulated environment easier.

A state transition diagram defines the states, transitions, and actions of the controlled system. It visually represents how the states change over time [8]. The complex relationships between states of the controlled system can be understood sequentially by generating a state transition diagram. Creating a simulation environment from the state transition diagram allows for the simulation of the behavior of the controlled system. It is necessary to generate a good state transition diagram for the effective creation of a simulated environment for reinforcement learning. However, it is difficult for inexperienced engineers to describe all the state transitions of the control target.

Instead of immediately drawing a state transition diagram, it is a plausible way to let inexperienced engineers describe the requirements of each state and its transition of the controlled target in natural language. By writing these requirements, we can develop a reinforcement learning environment.

2.2. The definition of a good state transition diagram

This study refers to a good state transition diagram as one that meets two conditions. These conditions are summarized below.

1. The relationships between the various states of the controlled object are appropriately represented
2. Through the state transition diagram, it clarifies the structure and overall picture of the target problem

The first condition appropriately represents the operations on the target object and the transitions between states for each state of the controlled object. The actions and transitions for each state become clear by constructing a simulation environment from the state transition diagram. By doing this, it ensures that the progression follows the flow of the controlled object's behavior. This condition is particularly indispensable for the accurate simulation of complex and diverse control systems.

The second condition is that the state transition diagram must represent the flow of the controlled object's behavior itself.

A state transition diagram visually represents the behavioral flow of the system under control. It should enable users to understand the behavior of a complex system at a glance. State transition diagrams should effectively model the complex relationships between states and the overall structure. They should promote users to grasp the big picture of the problem. A properly designed state transition diagram enables users to easily trace and analyze system behavior and possible scenarios. A state transition diagram that satisfies these conditions enables users to visually understand the structure of the problem, from the overall picture to the details.

2.3. Data generation using LLM

With the development of LLMs such as ChatGPT, some researcher has been studied the use of LLMs for generating experimental data and creating requirement specifications.

In [9], a methodology was proposed for generating diverse test data in formats such as HTTP and JSON using LLMs. This study improves the quality of the generated data by applying the following three approaches: prompt engineering, unknown information completion, and iterative generation. This enhances the model's understanding and improves response accuracy.

In [10], methods for using LLMs to assist the requirements engineering process were explored. Their study demonstrated that effective assistance from LLMs requires appropriate prompt engineering and the supplementation of domain knowledge. Collaboration with experienced engineers was also found to enhance the completeness of requirements. This result demonstrates the effectiveness of utilizing LLMs as a complement to human expertise. Additionally, in [11], an approach for automatically generating formal program specifications using LLMs was proposed. Their method employs an interactive specification-generation process in which error messages from generated program specifications are fed back into the LLM as prompts to improve generation accuracy. The results demonstrated that the quality of the generated specifications could be improved iteratively.

All these studies aim to improve the accuracy of data generation using LLMs. There has been little research comparing the quality of data generated by LLMs with that created by humans.

To overcome the problem, this study provides LLMs with the same information as human participants and tasked with generating data as virtual participants. This method analyzes the generating data using the same method applied to human-generated data. By comparing these results, it becomes possible to evaluate the performance of LLMs.

3. Assist in specifying requirements aiming to generate state transition diagrams

3.1. Assist using templates

The generation of reinforcement learning environments needs good state transition diagrams. However, it is difficult for inexperienced engineers to describe the state transitions of a control object

without omissions. Instead of directly drawing a state transition diagram, a requirement document can first be created to describe, in natural language, how the system under control transitions from one state to another when a specific external input is received in a given state. Good requirements lead to the generation of good state transition diagrams. Furthermore, it leads to the generation of a good reinforcement learning environment. If we assist in the generation of good requirements, users are expected to generate the environment.

The study investigates the effectiveness of two methods to assist in describing requirements to obtain good state transition diagrams. It considers that the degree of understanding varies depending on the way information is given. The following information is presented to inexperienced users who describe requirements.

1. Providing a model document
2. Providing hints for description

Using edit distance measures the improvement in the quality of the requirement by each assist method. The methods proposed in the study help inexperienced users to learn how to write requirements.

The overall diagram of the method is shown in Figure 1. The method provides templates to get users to generate requirements. To prompt them to revise their requirements, the method provides model documents. User requirements similar to their own writing style would have them learn how to write good requirements.

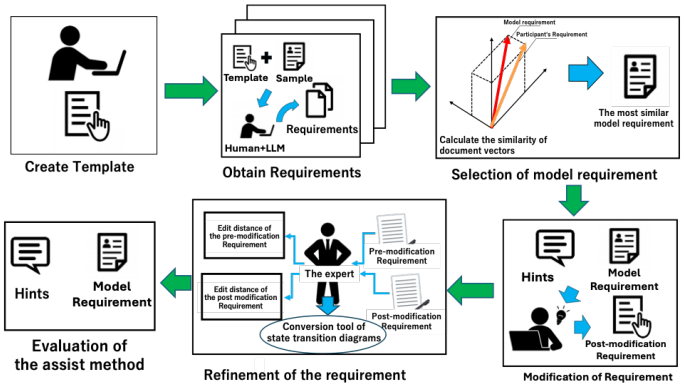


Figure 1: The overall diagram of the method

As the next step, experts refine both the pre-modification and post-modification requirements so that they can be fed to a generation tool of state transition diagrams. The edit distances are calculated for the requirements before and after refinement to examine their values. By analyzing these values, the method can measure the degree to which the user has mastered the writing style of requirements. In other words, the edit distance before and after refinement serves as the benchmark for understanding the writing style of requirements. If the change in edit distance shows a significant decrease before and after refinement, it is likely that the user has mastered the writing style of requirements, indicating the effectiveness of this method.

3.2. Templates for Requirements

The study assists in the generation of requirements for good state transition diagrams using templates. Templates prevent users from including unnecessary information as well as from omitting essential information. Such constraints on the user enable the method to assist the design of the user's state transition diagrams.

The study conducts preliminary experiments to generate templates. The experiment requires participants to specify requirements for generating state transition diagrams for a specific task. The task is the card game, Old Maid. 12 male university students take part in the experiment.

First, the experiment provides a manual containing examples of how to write requirements and an example of a requirement document written for the card game Blackjack. The participants write requirements for generating a state transition diagram for the card game, Old Maid, in free form. The expert refines the requirements so that a good state transition diagram can be created.

During the refinement process, for each participant, the expert summarizes the changes made to the original document and the reasons for those changes in a separate file. The study extracts the commonality among these changes. The study regards it as a characteristic that facilitates the conversion into a good state transition diagram. A list of these characteristics is summarized below.

1. We should use the object of interest (in this case, the cards) as the subject instead of the player
2. We should describe state transitions in the format: ⟨ current state⟩ transitions to ⟨ next state⟩ when ⟨ condition⟩ is met
3. We should describe operations on objects clearly and concisely
4. We should avoid using abbreviated expressions like 'in the above case' or 'otherwise'; namely, we should always describe conditions explicitly
5. We should include at least one state transition description for every state except the stop state
6. Each state description should include two types of information: operations on objects in the state and the transition to the next state.

The study converts these characteristics into a template to add them as notes in the precautions to assist inexperienced engineers in describing requirements to generate good state transition diagrams. Figure 2 shows the precautions based on these characteristics, while Figure 3 shows the template.

Requirement Precautions

1. Describe state transitions in the format: '<current state> transitions to <next state> when <condition> is met.
2. For each state, specify two types of information: operations on objects in the state and state transition.
3. Descript operations from the player's perspective, Descript state transition from the perspective of the cards changing
4. Describe operations on objects clearly and concisely.
5. Do not use abbreviated expressions like 'in the above case' or 'otherwise', i.e., always describe conditions explicitly.
6. Include at least one transition to another state for every state except the stop state

Figure 2: Precautions of Requirement

Requirement Template

1. <state name>
・<subject>becomes<operation>.
・<current state> transitions to <next state> when <condition> is met.
2. <state name>
・<subject>becomes<operation>.
・ <current state> transitions to <next state> when <condition> is met.
・ <current state> transitions to <next state> when <condition> is met.
3. <state name>
・Loss occurs when in the <state>.
4. <state name>
・Gain reward when in the <state>

Figure 3: Template of Requirement

The precautions and templates enable us to describe requirements that satisfy the characteristics in Figure 2.

Figure 4 shows an example of a requirement for "Old Maid" written in accordance with the precautions and template.

1. Initial State
• Cards are dealt to each player.
• Once the hands are determined, it transitions to the pre-processing state.
2. Pre-processing State
• The player checks their cards. if there are any pairs of cards with the same number, those pairs are removed.
• This process is repeated until no pairs remain.
• If no pairs are left in the hand, it transitions to the state of drawing a card.
3. Drawing a Card State
• The player selects one card from the opponent's hand and adds it to their own hand.
• Once the card is added, it transitions to the pair-checking state.
4. Pair-Checking State
• If a pair is formed between the drawn card and the player's hand, the pair is removed from the hand.
• If no pair is formed, the drawn card is added to the player's hand.
• If the player's hand is empty, it transitions to the victory state.
• If there are still cards remaining and no pairs, it transitions to the state of drawing a card.
5. State of Drawing a Card from Opponent
• The opponent draws one card from the player's hand.
• If the opponent still has cards, it transitions to the drawing a card state.
• If the opponent has no cards, it transitions to the defeat state.
6. Defeat State
• The player suffers a loss when in the defeat state.
• It transitions to the end state.
7. Victory State
• The player receives a reward when in the victory state.
• It transitions to the end state.
8. End State
• The game ends.

Figure 4: Requirement using the template and precautions

The example satisfies the six characteristics mentioned above. Fig 5 presents the state transition diagram created from the example.

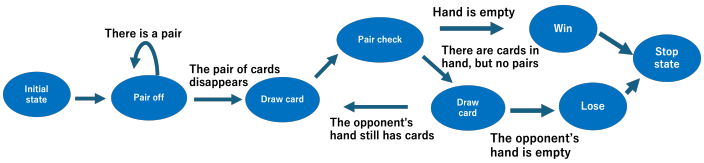


Figure 5: State Transition Diagram

The state transition diagram reveals the structure of operations and state transitions for the given task, Old Maid. Both the example and the diagram appropriately show the operations and state transitions at each state. Based on this, the state transition diagram satisfies the definition of a good state transition diagram outlined in Section 2.1. The sentences constitute a good requirement. The proposed method gets the user to write the requirement following the template and precautions.

3.3. Obtaining Requirements Data Using Templates

This study collects requirements data using the same method as the preliminary experiment to evaluate the extent to which users understand how to write requirements with the assistance of templates and precautions. First, participants are provided with instructions without templates or precautions, and their requirements data are collected to establish a baseline for evaluating the effectiveness of writing guidance.

Next, participants are divided into two groups to investigate the hypothesis that the order in which information is presented may influence their understanding of the instructions. Both groups receive the same template but differ in the placement of the precautions within the instructions. The first group receives instructions with precautions for writing a good state transition diagram placed at the beginning, as shown in Figure 2. In contrast, the second group receives instructions with the same precautions placed at the end. Since readers generally process instructions from top to bottom, information presented earlier may receive greater attention than information presented later. This grouping is designed to examine whether the placement of precautions influences the quality of the resulting requirements.

Additionally, large language models (LLMs), such as ChatGPT, are employed as virtual participants to increase the number of requirements collected and compare their performance with that of inexperienced human participants. Since LLMs are trained on extensive datasets, they are expected to have greater familiarity with requirement-writing conventions than inexperienced users. Similar to the human participants, the virtual participants are divided into two groups and provided with the same templates, precautions, and reference example. The following prompt is given to the LLMs:

> "Please write the document according to the following instructions. You will now write a requirement for a state transition diagram for a given task. The task is 'Old Maid'.
>
> Read and understand the template and the contents of the instruction from top to bottom and write the requirement for the state transition diagram of Old Maid

according to the provided content. For reference, an example of a requirement written for the card game 'Blackjack' is provided. Please use it as a reference for writing style and structure."

Generally, the accuracy of responses generated by LLMs, such as ChatGPT, can be improved by clearly specifying the purpose and background of the task, as well as the desired output format [12]. Accordingly, the prompt explicitly introduces the task and its purpose to guide the generation process. Furthermore, a reference example is provided to clarify the expected writing style and structure, with the aim of improving the consistency and accuracy of the generated requirements.

3.4. Selection of Model Documents

In this method, requirements written by multiple inexperienced users convert into document vectors using Word2Vec[13]. These vectors are then embedded into a specific-dimensional space. Additionally, experts convert these requirements into correct examples, which are also transformed into document vectors and embedded in the same dimensional space. This enables the calculation of similarity between the user's requirement and the correct examples. The correct example with the highest similarity is provided as the model document for that user. Figure 6 shows the method for calculating similarity between documents.

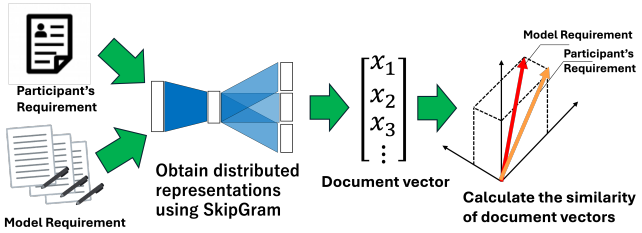


Figure 6: Method for Selecting Model Documents

To determine similarity, MeCab segments the user's requirement and the correct example document into words. Next, SkipGram, a Word2Vec technique, converts each word into a meaningful embedding. Finally, calculating the average of all word vectors in the document defines the document vector. This method calculates document similarity by applying cosine similarity to the calculated document vectors.

Training the Word2Vec model requires a corpus of text documents. Generally, a larger corpus provides more training data, which can improve the quality of learned word embeddings [14, 15]. In this study, the corpus consists of 16 documents: 12 expert-refined requirements obtained from the preliminary experiment and four additional example documents created by the researchers. These four documents were added to increase the amount of training data. All documents were related to the target task, Old Maid, to support the selection of model documents relevant to the experimental requirements.

However, the relatively small corpus size may limit the reliability and stability of the learned word embeddings. Furthermore, the use of task-specific documents may influence the calculated similarities and the selection of model documents. Therefore, the present study does not establish that the Word2Vec-based selection method is optimal or generalizable to other tasks. Further evaluation using larger and more diverse corpora is required to assess the reliability of this selection method.

3.5. Modification of Requirements Using Model Documents

Humans are likely to understand advice better when it is both general and specific. Therefore, by viewing a correct example similar to their own requirement, users can more easily identify the issues in their requirements and understand the characteristics of a good requirement. In other words, by presenting a model document similar to the requirement written by the user, the user is more likely to learn how to write a good requirement and improve the quality of their own requirement.

This method provides the correct example with the highest document vector similarity to the user's requirement as the model documents as explained in section 3.4. The user modifies their own requirement by referencing it. Instead of directly using the sentences from the model document, the user is instructed to identify the parts where the perspective or writing style differs from their own requirement and make changes to those sections. This instruction gets the user to only modify the parts where they have understood the writing style. By comparing the document before and after the changes, it is possible to measure how well the user has learned the effective writing techniques.

3.6. Refinement of Acquired Requirement Documents

This study defines the requirement before modification as the 'pre-modification requirement' and the requirement after modification as the 'post-modification requirement'. The expert refines each requirement for the participants until it can be input into the state transition diagram creation tool. The state transition diagram creation tool automatically generates state transition diagrams from requirements based on the templates and characteristics shown in section 3.2. The tool performs dependency parsing on the input requirement, extracting parts related to specific words as states or transition information. Based on these words, it generates a state transition diagram. If the refined requirement produces a good state transition diagram when input into the tool, it can be considered a requirement that meets the specified characteristics. Therefore, the refinement process continues until the requirement produces a good state transition diagram when input into the tool.

This method calculates the edit distance between both the pre-modification and post-modification requirements and the refined requirement. The degree of change in edit distance can evaluate the effectiveness of the modifications described in Section 3.5.

Edit distance is a metric that measures the similarity between two sentences by applying specific edit operations to transform one sentences into another.This method uses the Levenshtein distance[16], which is defined as the minimum number of insertions, substitutions, and deletions needed to transform one sentence into another. However, the Levenshtein distance increases as the length of the text grows, even if the two sentences are similar. To solve this problem, this study uses a standardized version of the Levenshtein distance. The standardized Levenshtein distance L_s is defined as follows.

$s1, s2$: Two Strings That Calculate Distance

$L(s)$: String Length

$\mathcal{L}(s1, s2)$: Levenshtein distance between two strings

$$L_s = \frac{\mathcal{L}(s1, s2)}{\max(L(s1), L(s2))} \quad (1)$$

A smaller edit distance indicates that fewer textual changes are needed to obtain the expert-refined document. By comparing the normalized edit distances before and after modification using model documents, this study examines the effectiveness of providing model documents. In addition, the effects of writing precautions and their presentation positions are evaluated by comparing the edit distances obtained under different experimental conditions.

However, edit distance measures textual differences rather than semantic quality. Therefore, a reduction in edit distance does not necessarily indicate improvements in the correctness, completeness, or consistency of the requirements. These aspects require complementary evaluations, such as expert assessments and direct evaluation of the generated state transition diagrams.

4. Template-based requirements

4.1. Experimental details

This study experiments with participants writing requirements for a given task to investigate the method's effectiveness in assisting their creation. This experiment analyzed whether revisions based on model documents improved participants' understanding of how to write requirements documents under conditions in which templates and precautions were provided.

In this experiment, participants were first provided with a requirement template and an instruction document without the guidelines created in Section 3.2. Participants were given a sample requirement describing a state transition diagram for the card game Blackjack. They referred to it and wrote a requirement for a state transition diagram of the card game Old Maid.

Next, participants were provided with the template and instruction that included precautions. They write a requirement for the same task. By giving a model requirement, they modify their own requirement. The participants in the experiment were 15 Japanese male college students.

4.2. Experimental Groups and Conditions

This experiment divided participants into two groups to investigate whether the position of writing precautions in the instructions affects their understanding of requirement writing. Group A received an instruction document with the precautions presented at the beginning, as shown in Figure 2, whereas Group B received an instruction document with the precautions presented at the end. Both groups were provided with the same requirement template and example task.

In addition to the human participants, this study used GPT-4o, a large language model developed by OpenAI, to generate requirement documents. The LLM experiment was conducted on November 26, 2024, using the ChatGPT web interface. The input provided to GPT-4o consisted of the prompt described in Section 3.3, the requirement template, and an example requirement document for the card game Blackjack. The position of the writing precautions in the instructions differed between Group A and Group B, as in the human experiment.

A total of 11 requirement documents were generated using GPT-4o, with six assigned to Group A and five assigned to Group B. A new chat session was started for each generation to avoid carrying over previous conversation histories. Since the experiment was conducted through the ChatGPT web interface, generation parameters such as temperature were not explicitly controlled or recorded.

In total, the experiment included 15 human participants and 11 LLM-generated requirement documents. The LLM-generated documents were used as supplementary data to explore differences in edit distance between human-written and LLM-generated requirements.

Furthermore, this experiment imposes the rules of the Old Maid game used in the task. These constraints are as follows:

1. The game is played with two players.
2. The turn order starts with the participant.
3. Low-probability events are not described.

These constraints simplified the structure of the Old Maid game. The constraints allowed them to focus more on creating the state transition diagram to reduce the number of state combinations for participants to consider. simplifying the game structure would help participants concentrate on the task. For this reason, the above constraints were imposed.

The 26 participants, consisting of both human and LLM virtual participants, were divided into Group A with 13 participants and Group B with 13 participants. Among them, Group A included 6 virtual participants, and Group B included 5 virtual participants.

4.3. Request for Requirement Modification Using Model Requirements

This experiment provides model requirement that had the highest similarity in document vectors to participants to aim of assisting requirement creation. Subsequently, participants were asked to modify their own requirements. The requirement written by the participants will be referred to as the pre-modification requirement.

The pre-modification requirement is divided into words, and each word is converted into the embedding using SkipGram, a method of Word2Vec. Subsequently, the embeddings are converted into a document vector. On the other hand, the requirements of some correct examples are also converted into document vectors using the same method. This method calculates the cosine similarity between the pre-modification requirement and each correct example's requirement. The participants receive the requirement of the correct example with the highest similarity.

This experiment instructs the participants to modify the parts of their own requirements that differed in perspective or writing style from the provided model requirement. Participants modified

only the parts of the pre-modification requirement where they recognized differences. The modified requirement is referred to as the post-modification requirement.

4.4. Testing for Significant Differences Based on the Type of Information Presented

Both the pre- and post-modification requirements were refined by an expert until they could be used as input to the state transition diagram creation tool. For each requirement, calculate the edit distance between the requirement after refinement and the original requirement.

We test a statistical test to determine whether there was a significant difference in edit distance across all participants. If there is a significant difference, it can be concluded that it is effective to help participants learn how to write requirements. The difference in edit distance caused by refinement was also calculated for the conditions with and without templates and precautions.

This study tests statistical test to determine whether there was a significant difference between these values. If there is a significant difference, he use of templates and precautions can be considered effective support for helping participants learn how to write requirements. Furthermore, for each group, this study calculates the difference in edit distance between the pre- and post-modification requirements resulting from the refinement. If a statistical test applied to this difference reveals a significant difference, it can be said that describing important matters earlier rather than later contributes more to human understanding.

5. Evaluation of the methods

5.1. The effect of providing a model document for changes

This study evaluates whether presenting model documents effectively assists in describing requirements. This study calculated the edit distance between the pre- and post-modification requirements and analyzed the difference. If the edit distance decreases after modification, presenting a model document can be considered effective in helping participants learn how to write requirements.

Figure 7 shows the histogram of edit distances pre and post-modification.

This result was calculated using only the data obtained from human participants, excluding LLMs. To verify whether there was a difference between the edit distances before and after the change, this study used the Wilcoxon signed-rank test. For the 15 human participants, the mean normalized edit distance decreased from 0.560 to 0.404, corresponding to a mean reduction of 0.156. The Wilcoxon signed-rank test indicated a statistically significant reduction (W = 1.0, p = 0.0012, n = 15). The matched-pairs rank-biserial correlation was 0.981, and the 95% bootstrap confidence interval for the mean reduction was [0.100, 0.223]. This difference in edit distance shows that presenting the model document led to a reduction in the edit distance. Among the 15 participants who were recruited to modify a requirement using a model document as a reference, the edit distance of the post-modification requirement was smaller than that of the pre-modification requirement for 14 of them (93%).

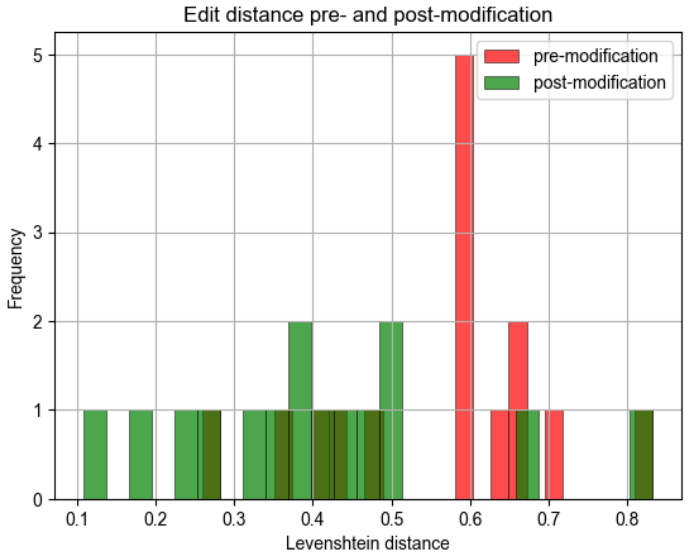


Figure 7: The histogram of edit distances pre- and post-modification

However, for one participant's requirement (7%), the edit distance was higher after modification. This suggests that the participants may have misunderstood the writing style or perspective of the model document presented to them. This study presented only the model document without additional explanations. This left considerable room for interpretation. As a result, while presenting the model document proved effective in assisting requirement creation, However, it suggests supplementary explanations need for to prevent misunderstandings.

This study analyzed edit distances not only for human participants but also for requirements generated by LLMs. Figure 8 shows a histogram of the 11 requirements generated by the LLM, combined with the results of the human experiment.

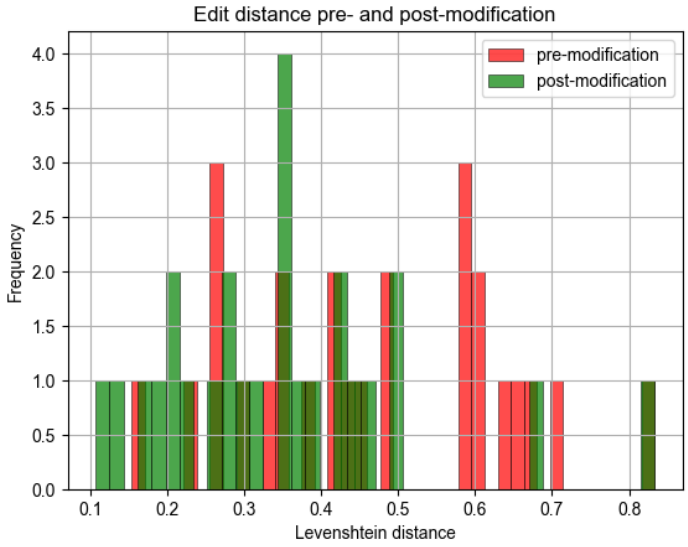


Figure 8: Disribution of Edit Distance for LLM and Human

The Wilcoxon signed-rank test showed a significant difference. This result follows the same trend as the findings for human participants. In addition, the average edit distance was 0.463 pre-

modification and 0.351 post-modification. Including LLMs further became even smaller than when only human data was used. This suggests that LLMs, when provided with appropriate advice or templates, may achieve better results in requirement creation than inexperienced users. LLMs are trained using large corpora of text from the internet. Especially, since numerous corpuses on the rules of Old Maid exist online. LLMs were likely able to generate appropriate documents with only partial instructions. As a result, LLMs demonstrated the potential to generate documents as intended, at a level comparable to or even surpassing humans, in domains with a large corpus available.

5.2. Impact of Precautions on Requirements Writing Assist

An expert refined the requirements created under two conditions: instructions with only the template and instructions with both the template and precautions. This study evaluated the effectiveness of providing precautions based on the edit distance from the refined requirements for both the pre- and post-modification versions. If the edit distance of the requirements written with the precautions is smaller than that of the requirements written without them, the writing hints can be considered effective for helping participants learn how to write requirements.

Figure 9 shows the histogram of edit distances with and without precautions.

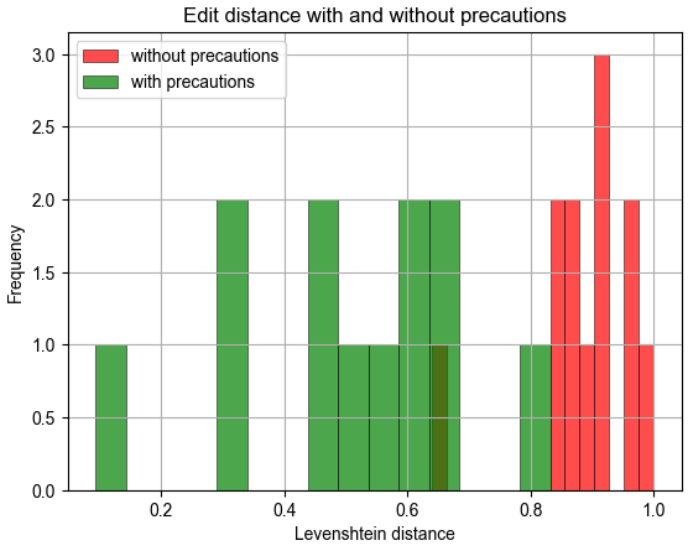


Figure 9: Histogram of Edit Distance With and Without Precautions

As shown in Figure 9, the edit distances differed substantially depending on the presence or absence of the precautions. The requirements written with precautions showed smaller edit distances than those written without it. The Wilcoxon signed-rank test confirmed that this difference was statistically significant. ($p = 9 \times 10^{-4}$) Based on the above results, it can be concluded that providing descriptive precautions is effective in supporting the creation of requirements.

The edit distances were generally larger for the requirements written without precautions, likely because participants had greater freedom in their expressions. Many of the precautions developed in Section 3.2 restrict the content that participants should describe. Therefore, the requirements written by participants who were provided with the precautions showed smaller edit distances after refinement. In contrast, when the precautions were not provided, participants tended to use ambiguous or unnecessary expressions, which likely increased the edit distance. However, because all participants completed the conditions in the same order, the observed reduction in edit distance may also have been influenced by familiarity with the task and repeated practice. Therefore, the effect cannot be attributed solely to the writing precautions.

5.3. Impact of Information Presentation Position

Section 5.2 showed that providing writing precautions was effective as a form of support for requirements writing. This may be because people tend to read documents from the beginning and place greater importance on information presented earlier. Therefore, placing important information, such as writing precautions, near the beginning of the instructions may contribute more effectively to participants' understanding of how to write requirements. Group A received an instruction manual with hints placed at the beginning, while Group B received one with hints placed at the end. This study measured the changes in edit distance for the requirements created by each group using the method described above.

Figure 10 shows the changes in the average edit distances between the pre- and post-modification requirements for both groups.

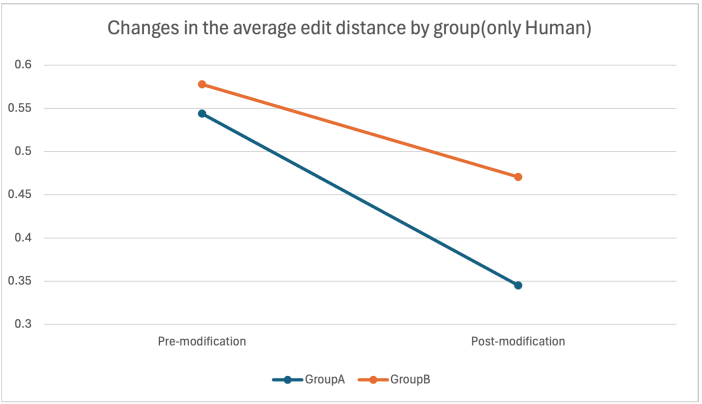


Figure 10: Changes in the Average Edit Distance by Group(only human)

Figure 10 shows that the edit distance decreased for both groups following the presentation of the model document.For the human participants, in Group A(n=7), the average edit distance decreased from 0.54 for the pre-modification requirements to 0.345 for the post-modification requirements. In contrast, for Group B(n=8), the average edit distance was 0.578 pre-modification and 0.471 post-modification. Thus, the edit distance decreased by approximately 0.2 in Group A, whereas it decreased by only approximately 0.1 in Group B. Based on Figure 10 and these results, there appeared to be a difference between the two groups in the change in edit distance from the pre-modification requirements to the post-modification requirements. To examine whether this difference was statistically significant, this method tested the Mann–Whitney U test. However, the test yielded a p-value of 0.232 at a significance level of 0.05 (U = 39.0).The rank-biserial correlation was 0.393, and the 95% bootstrap confidence interval for the difference in mean reductions

between the groups was [-0.013, 0.223]. This result indicates that there was no statistically significant difference in the changes in edit distance between the two groups. This indicates that there was no statistically significant difference between the two groups in the degree of change in edit distance.

Next, this study adds the virtual participants generated by LLM to the experiment. The results are then reanalyzed.Figure 11 shows the changes in the average edit distances between the pre- and post-modification requirements for both groups adding virtual participants generated by the LLM.

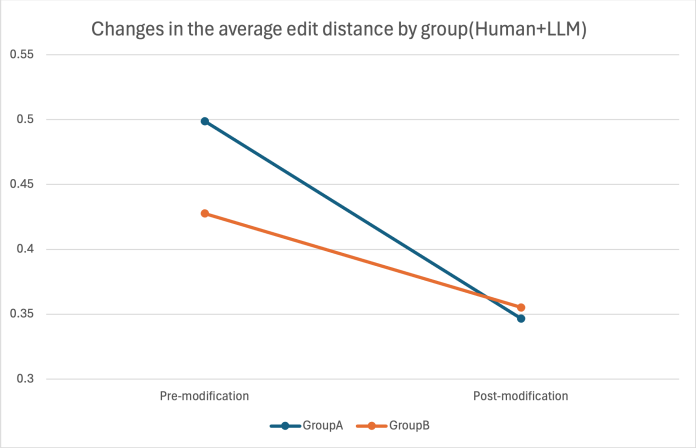


Figure 11: Changes in the Average Edit Distance by Group(LLM add)

As shown in Figure 11 even when the virtual participants were included, the edit distances of the post-modification requirements decreased, as in Figure 10. In Group A(n=13), the average edit distance was 0.499 for the pre-modification requirements and 0.347 for the post-modification requirements. In Group B(n=13), the average edit distance was 0.427 pre-modification and 0.355 post-modification. As a result, the decrease in edit distance for Group A remained larger than that for Group B. This trend was consistent with the experiment conducted with human participants only. However, the statistical test results showed a p-value of 0.106 (U = 116.5). The rank-biserial correlation was 0.379, and the 95% bootstrap confidence interval for the difference in mean reductions was [-0.006, 0.174]. This indicates that a significant difference was still not observed.

Therefore, based on the data obtained in this study, it is not possible to determine whether placing important information at the beginning is effective for helping participants learn how to write requirements.Further evaluation with a larger and more diverse sample is needed to examine this effect. Furthermore, the combined analysis should be interpreted as exploratory because human-written and LLM-generated requirements have different data-generating processes.

5.4. Limitations of the Study

This study has several limitations that should be considered when interpreting the experimental results.

First, the experiment involved only 15 male university students and focused on a single task, the card game Old Maid. Therefore, the generalizability of the findings to professional engineers and more complex reinforcement learning tasks remains uncertain.

Second, the experimental conditions were presented in a fixed order. Consequently, the observed reductions in edit distance may have been influenced by participants' increasing familiarity with the task and repeated practice. Thus, the improvements cannot be attributed solely to the proposed assistance methods.

Third, the LLM-generated requirements were compared with those written by human participants. However, GPT-4o may have acquired prior knowledge of Old Maid through pretraining, whereas human participants had different levels of prior knowledge. Therefore, this comparison should be interpreted as exploratory rather than as direct evidence of the superiority of LLMs.

Finally, the evaluation primarily relied on normalized Levenshtein edit distance, which does not directly measure the correctness or completeness of requirements. In addition, the Word2Vec model was trained using only 16 task-specific documents, potentially limiting the reliability of model document selection. Future work will address these limitations by evaluating the quality of generated state transition diagrams and examining the method using more diverse participants and tasks.

6. Conclusion

This study proposes a method for writing requirements for good state transition diagrams to assist in building reinforcement learning environments. The participants write requirements for the task using templates and instructions. Afterward, a model document is presented to them to modify. The expert refines the pre- and post-modification requirements until they could be input into a state transition diagram creation tool. This method calculates the edit distance between pre- and post-modification requirements and the refined requirements.

In this way, this study validates the effectiveness of this method. To verify the effectiveness of the proposed method, we calculated the edit distances between each requirement modified with reference to a model document and its corresponding refined version.

In addition, we calculated the edit distances between the requirements documents written by users who were provided with the precautions and those written by users who were not. The experimental results confirmed that presenting model document and precautions is effective for writing requirement documents. An additional analysis also showed lower edit distances for requirements written with precautions than for those written without them. However, changing the order in which the precautions were presented did not produce a statistically significant difference. Additionally, when adding the data generated by LLM, the edit distance resulted in a lower value compared to the human data. These results suggest that LLMs can generate requirement documents with relatively low edit distances when provided with guidance such as precautions and templates, although this does not necessarily indicate superiority over human participants. However, this study has several limitations, including the small number of human participants, the use of a single task, and the reliance on textual edit distance. In particular, a reduction in edit distance does not necessarily indicate improvements in the correctness or completeness of generated state transition diagrams.

In this method, the original text of a model document was

presented to the participants when they modified their requirements. To make the presentation of model documents more effective, adding explanations of the writing characteristics of the model documents may further reduce the edit distance. Future work will directly evaluate the correctness and completeness of state transition diagrams generated from the requirements to examine whether reductions in edit distance correspond to improvements in diagram generation.In addition, it will be necessary to examine the effectiveness of this support by providing participants with both a model document and explanations of its writing characteristics when they modified their requirements. The results should then be compared with those obtained when only the original text is provided.

Also, the method proposed in this study can be applied not only to Japanese but also to other languages. Furthermore, the method has the potential to facilitate the construction of environments for complex RL tasks, such as autonomous driving.

References

[1] M. Masmoudi, H. Friji, H. Ghazzai, Y. Massoud, "A Reinforcement Learning Framework for Video Frame-Based Autonomous Car-Following," IEEE Open Journal of Intelligent Transportation Systems, **2**, 111–127, 2021, doi:10.1109/OJITS.2021.3083201.

[2] P. Cai, H. Wang, H. Huang, Y. Liu, M. Liu, "Vision-Based Autonomous Car Racing Using Deep Imitative Reinforcement Learning," IEEE Robotics and Automation Letters, **6**(4), 7262–7269, 2021, doi:10.1109/LRA.2021.3097345.

[3] P. Cai, H. Wang, Y. Sun, M. Liu, "DQ-GAT: Towards Safe and Efficient Autonomous Driving with Deep Q-Learning and Graph Attention Networks," IEEE Transactions on Intelligent Transportation Systems, **23**(11), 21102–21112, 2022, doi:10.1109/TITS.2022.3184990.

[4] S. P. M. Choi, D.-Y. Yeung, N. L. Zhang, "An Environment Model for Nonstationary Reinforcement Learning," in S. A. Solla, T. K. Leen, K.-R. Müller, editors, Advances in Neural Information Processing Systems, volume 12, 307–313, MIT Press, 1999.

[5] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, USA, 2nd edition, 2018.

[6] Z. Liu, J. Lu, J. Xuan, G. Zhang, "Deep Reinforcement Learning in Nonstationary Environments With Unknown Change Points," IEEE Transactions on Cybernetics, **54**(9), 5191–5204, 2024, doi:10.1109/TCYB.2024.3356981.

[7] W. Mori, F. Harada, H. Shimakawa, "Evaluating Assists in Specifying Requirements for Reinforcement Learning," in 2025 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC), 34–39, 2025, doi:10.1109/MIUCC66482.2025.11196869.

[8] D. R. Windle, L. R. Abreo, Software Requirements Using the Unified Process: A Practical Approach, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2002.

[9] T. Tóthfalusi, Z. Csiszár, P. Varga, "Utilizing Generative AI for Test Data Generation – Use-Cases for IoT and 5G Core Signaling," in 2024 IEEE Network Operations and Management Symposium (NOMS), 1–6, 2024, doi:10.1109/NOMS59830.2024.10574974.

[10] C. Arora, J. Grundy, M. Abdelrazek, "Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs," in A. Nguyen-Duc, P. Abrahamsson, F. Khomh, editors, Generative AI for Effective Software Development, 129–148, Springer, 2024, doi:10.1007/978-3-031-55642-5_6.

[11] L. Ma, S. Liu, Y. Li, X. Xie, L. Bu, "SpecGen: Automated Generation of Formal Program Specifications via Large Language Models," in Proceedings of the 47th IEEE/ACM International Conference on Software Engineering (ICSE), 16–28, 2025, doi:10.1109/ICSE55347.2025.00129.

[12] J. White, S. Hays, Q. Fu, J. Spencer-Smith, D. C. Schmidt, "ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design," in A. Nguyen-Duc, P. Abrahamsson, F. Khomh, editors, Generative AI for Effective Software Development, 71–108, Springer, 2024, doi:10.1007/978-3-031-55642-5_4.

[13] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2nd edition, 2019.

[14] K. Saito, Deep Learning from Scratch 2, O'Reilly Japan, 2018.

[15] M. Sahlgren, A. Lenci, "The Effects of Data Size and Frequency Range on Distributional Semantic Models," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 975–980, Association for Computational Linguistics, Austin, Texas, 2016, doi:10.18653/v1/D16-1099.

[16] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," Soviet Physics Doklady, **10**(8), 707–710, 1966, english translation of the original article published in Doklady Akademii Nauk SSSR, vol. 163, no. 4, pp. 845–848 (1965).

Copyright: This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-SA) license (https://creativecommons.org/licenses/by-sa/4.0/).