https://doi.org/10.31891/csit-2025-1-13
UDC 004.75

Yevhenii VOVK, Juliya POLUPAN
National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute»

A NEW APPROACH FOR CREATING CHATBOTS BASED ON THE USE OF FINITE AUTOMATA THEORY

Nowadays, the era of waiting in lines, writing official letters, and direct contact with employees of institutions, establishments, and companies is gradually becoming a thing of the past. Instead, the problem of creating tools that ensure the development, implementation, and implementation of chatbots and agents, their support, and expansion of functionality, and scalability, arises.

The main subject of this article is precisely the representation of a chatbot in the form of a state diagram. This technology, together with the technology of analysis and synthesis of formal chatbot models, constitute important components of the platform and information systems as a whole for institutions, establishments, and companies of various levels. An analysis of the possibilities of automata theory has shown the feasibility of using transitional systems and finite automata such as X-automata, Mealy and Moore automata as chatbot models.

The article describes a general approach to the effective representation of a chatbot in the form of a state diagram, implemented within the framework of a platform for the development, accumulation, and use of chatbots. As a formal model of a chatbot, it is proposed to use finite automata of Mealy and Moore, and the transformation of a regular expression, which is based on the input and output alphabets of the system, to a certain graphic configuration is proposed to be carried out according to known algorithms for the synthesis of X-automata.

In the case of a formal description of the business process, the corresponding transition system or automaton is formed on the basis of a decision tree containing pairs <initial state, final state>. If there is no description of the business process, then an algorithm for synthesizing the corresponding automaton based on a set of necessary lines of behavior/scenarios represented by regular expressions is proposed.

Based on the synthesis, a ready-made solution for a telegram bot was formed, on the basis of which a telegram bot was created using the existing messenger software interface and the execution time of a particular line of behavior/scenario for a specific task "Taxi Ordering". Taking into account the time and sequence of message and response creation, an approach was also proposed to calculate the chatbot operation time for different scenarios. It was determined that for standard scenarios T1= 204 (s), T2=324 (s), T3=467 (s), T4=80 (s) provided that the response from the data source (web service) on the available car types and the actual availability of the selected car is received in less than 20 seconds.

Keywords: finite state machine, messenger, chatbot, Telegram, transition system, Mealy automaton, Moore automaton, X-automaton, synthesis of formal models, synthesis algorithm, natural language, natural language interface.

Євгеній ВОВК, Юлія ПОЛУПАН
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

НОВИЙ ПІДХІД ДО СТВОРЕННЯ ЧАТ-БОТІВ НА ОСНОВІ ВИКОРИСТАННЯ ТЕОРІЇ СКІНЧЕННИХ АВТОМАТІВ

Нині, епоха очікування в чергах, написання офіційних листів, безпосереднього контактування з працівниками установ, закладів і компаній поступово відходить в минуле. Натомість постає проблема створення інструментальних засобів, що забезпечують розроблення, реалізацію і впровадження чат-ботів та агентів, їх підтримку і розширення функціональності та масштабування.

Головним предметом цієї статті є саме представлення чат-бота у вигляді діаграми станів. Ця технологія разом з технологією аналізу і синтезу формальних моделей чат-бота становлять важливі складові платформи та інформаційних систем в цілому для установ, закладів і компаній різних рівнів. Аналіз можливостей теорії автоматів показав доцільність використання у якості моделей чат-ботів транзиційних систем та скінченних автоматів, таких як Х-автомати, автомати Мілі та Мура.

У статті описано загальний підхід до ефективного представлення чат-бота у вигляді діаграми станів, реалізованої в рамках платформи розроблення, накопичення і використання чат-ботів. У якості формальної моделі чат-боту запропоновано використання скінченних автоматів Мілі та Мура, а трансформацію регулярного виразу, в основі якого лежать вхідний та вихідний алфавіти системи, до певної графічної конфігурації пропонується проводити за відомими алгоритмами синтезу Х-автоматів. У випадку наявності формального опису бізнес-процесу відповідна транзиційна система або автомат формується на основі дерева виведення рішення, що містить пари<початковий стан,кінцевий стан>. Якщо опису бізнес-процесу не існує, то запропоновано алгоритм синтезу відповідного автомата на основі множини потрібних ліній поведінки/сценаріїв, представлених регулярними виразами.

На основі проведеного синтезу було сформоване готове рішення для телеграм-боту, на основі якого створено телеграм-бот шляхом використання існуючого програмного інтерфейсу меседжеру і визначено час виконання тої чи іншої лінії поведінки/сценарію для конкретної задачі «Замовлення таксі». Врахувавши час та послідовність створення повідомлень та відповідей, також був запропонований підхід для розрахунку часу роботи чат-бота для різних сценаріїв. Визначено, що для стандартних сценаріїв Т1= 204 (с), Т2=324 (с), Т3=467 (с), Т4=80 (с) за умови отримання відповіді від джерела даних (веб сервісу) по наявним типам автомобілів та фактичній наявності обраного автомобілю менше ніж за 20 секунд.

Ключові слова: скінченний автомат, мессенджер, чат-бот, Telegram, транзиційна система, автомат Мілі, автомат Мура, Х-автомат, синтез формальних моделей, алгоритм синтезу, природна мова, природно-мовний інтерфейс.

Introduction

The modern level of information technology (IT) development has created conditions for improving human life in all areas of activity. This is particularly evident in customer service within government institutions, banks, healthcare facilities, retail companies, and other sectors. To meet the growing needs and expectations of customers, these institutions and companies often describe their service processes in the form of business processes. This structured, algorithmic approach allows them to streamline and monitor service delivery, making it faster and more convenient for citizens. At the same time, it facilitates the informatization and automation of customer service operations, minimizing the costs associated with training and maintaining staff who assist clients in fulfilling their needs. One of the key outcomes of this informatization and automation process is the widespread adoption of chatbots.

In [1], the authors proposed a platform that integrates the development, accumulation, and utilization of chatbots. This solution unifies user interactions across various messengers, social networks, and information systems in terms of both content and functional capabilities. The developed platform could be particularly beneficial for Software as a Service (SaaS) providers and Platform as a Service (PaaS) providers by addressing the shortcomings of existing decentralized solutions and enhancing the quality of user service.

However, developing such a platform presents a number of intriguing yet complex subproblems that must be addressed.

Firstly, to enable the rapid development, maintenance, and enhancement of effective and useful chatbots, it is necessary to create formal models of chatbots along with methods for their analysis and synthesis.

Secondly, given the broad scope of operations in government institutions, banks, healthcare facilities, and retail companies, chatbots should be viewed as part of a larger information system within these organizations. These chatbots should be integrated with other system components to define complex behaviors aimed at achieving specific goals. This requires corresponding formal models and methodologies.

For the third time, for a chatbot to function as an effective online assistant, it must support natural language communication, understand customer needs, respond to their queries, and provide appropriate services to fulfill their requests.

Then, the platform should deliver chatbots that are accessible to users accustomed to modern IT environments, leveraging well-established and widely adopted technologies, such as messaging platforms. Consequently, an important challenge is to represent chatbots, synthesized as formal models, using BPMN diagrams that are compatible with well-known messaging systems.

Fifthly, since chatbots often need to perform specific functions—such as data analysis, execution of actions, monitoring, transmission, reception, storage, and retrieval of information—an essential challenge is the automated implementation of these functionalities.

And finally, to ensure that the platform retains and enhances the advantages of machine-based systems—such as ease of deployment, scalability, cost-effectiveness, low latency in customer service, and multilingual support—it is crucial to address challenges related to automatic code generation based on chatbot activity templates.

Various models, methods, and technologies for solving most of the above challenges have been explored by the authors in previous studies [2,3], which focus on the development of a platform for chatbot creation, accumulation, and utilization. This platform represents a complex research object. The primary focus of this paper is the fourth subproblem—representing a chatbot using BPMN diagrams that are suitable for integration with widely used messaging platforms. Since this transformation involves the conversion of a state diagram, derived from a given chatbot description, into a BPMN representation, we will also address the first subproblem by proposing formal models of chatbots and corresponding methods for their analysis and synthesis.

The rise of messaging platforms and chatbots has introduced new opportunities while simultaneously creating challenges in representing chatbots as formal models based on states and transitions. The corresponding technology for chatbot representation using state diagrams, combined with formal model analysis and synthesis methods, constitutes a crucial component of the chatbot platform and the information systems of organizations. However, developing such technology requires adequate chatbot models, effective analysis and synthesis methods, and seamless transformation of state diagrams into BPMN diagrams, ultimately leading to the creation of efficient applications.

Analysis of recent research and publications

Natural language possesses several characteristics that any analytical system must take into account. One of the most significant challenges is ambiguity, which includes polysemy (words having multiple meanings) and homonymy (words that look or sound the same but have different meanings). A single idea can be expressed in multiple ways, which makes text processing for machine understanding highly complex.

Since text inherently has a complex internal structure, its effective processing requires a mathematical model. The objective of our research was to develop a robust framework for text processing and its representation in a structured internal format that ensures clear and accurate interpretation. This challenge has been partially addressed in prior studies [3,4].

From both theoretical and implementation perspectives, one of the most challenging aspects of chatbot

development is designing a natural language interface. This complexity arises from the need to convert natural language texts into an internal representation and vice versa.

Chatbots are generally classified into declarative and predictive types. Predictive chatbots, or selflearning bots, are built using a combination of artificial intelligence (AI), machine learning (ML), and natural language processing (NLP) technologies. In such cases, ontological entities are generated by AI and used within the bot's operational scenarios.

The simplest to implement are declarative chatbots, which rely on predefined conversational patterns. However, creating high-quality conversational scenarios is far from trivial. Ontological entities for such scenarios can be defined either manually by the developer, considering the bot's specific functionality, or automatically generated using AI. In the latter case, a sufficiently comprehensive dataset is required for training the AI model to extract relevant ontological entities for scenario construction. Question-and-answer datasets are particularly useful for chatbot training, and such datasets can be found on platforms like Kaggle or OpenML.

Chatbot development can be approached using Python, along with two key libraries:

• NLTK (Natural Language Toolkit) for natural language processing,

• TensorFlow for building and training machine learning models [5].

Another approach is to use pre-built frameworks and toolkits that facilitate chatbot creation, testing, deployment, and management. For instance, Bot Framework [6] provides a modular and extensible SDK for building bots with AI-powered features such as speech recognition, natural language understanding, and question-answering capabilities.

Recently, libraries that visualize user-agent interaction through graph-based structures have gained popularity [7,8]. One example is LangGraph, a library from the LangChain ecosystem, designed for agentbased system development. It helps determine the appropriate actions in response to user requests and enables the creation of graph-based applications that can be easily modified and expanded for various text analysis tasks.

An effective chatbot design must account for its behavioral aspects, including interactions with both users and other chatbots. These aspects can be conveniently modeled and implemented using transition systems (TS) [9]. A transition system is formally defined as follows [9, 10]:

$$A = (S, T, \alpha, \beta, s_0) \quad (1)$$

where S – a finite or infinite set of states,

T – a finite or infinite set of transitions,

α,β – mappings from T to S, assigning each transition t a starting state α(t) and an ending state β(t),

s_0 – the initial state of the transition system.

Within the scope of our research, given the applicability of transition systems, we aim to define a methodology for developing graphical schemes that facilitate the construction of chatbot applications. This methodology is fundamentally based on finite state machine (FSM) theory, enabling structured chatbot design and implementation.

Purposes of the article

The problem of effectively representing a chatbot as a state diagram must be addressed within a platform for chatbot development, accumulation, and utilization. As mentioned earlier, a transition system model can serve as a formal representation of a chatbot. However, this study proposes an approach based on finite-state automata (FSA), specifically X-automata, Mealy machines, and Moore machines. The choice of this approach is driven by the need to synthesize a finite-state automaton (FSA) that ensures the required chatbot behavior.Two main cases must be considered:

In the first case, when formal business process descriptions exist, the corresponding FSA is constructed based on a decision derivation tree that consists of <initial state, final state> pairs. In this scenario, the models and methods proposed in [2,3] can be applied. The main tasks are:

1) Transforming the decision derivation tree into a corresponding automaton.

2) Integrating multiple automata, if necessary, when they share identical initial and/or final states.

In the second case, when formal business process descriptions are not available, the FSA is synthesized based on predefined chatbot behaviors, which are specified using a set of regular expressions.

Presentation of the main material

General Approach to Representing a Chatbot as a State Diagram

The general idea is to identify the user's intent during the interaction with the chatbot. An intent is treated as a target state of the automaton, which, together with the current state, defines a problem that the chatbot will help solve. This is achieved by formulating a sequence of questions and analyzing user responses [4]. In other words: based on the user-provided input, the system must determine the current state of the automaton, which represents the user's current situation. It must also identify the target state, which represents the user's desired situation.This means locating the user's position within the finite-state automaton, which is defined by the business process, and

determining the target position that the user wants to reach.

Additionally, the approach must take into account a hierarchical structure of process descriptions, following the process → subprocess → operation paradigm.

Hence, there are descriptions of business processes that include descriptions of subprocesses, which, in turn, include descriptions of operations. In addition to hierarchical connections at each level, they are linked according to the order of their execution. There are also inputs and outputs of processes, subprocesses, and operations, conditions, and other components.

An operation, subprocess, and business process will be associated with a state in the automaton. The implementation of this task will include two key approaches: analysis of transition conditions and analysis of results. Since the transition between each element in the business process and the automaton is characterized by a certain condition, as well as the result of execution and a post-condition, they can be analyzed to determine the necessary position of the user in the graph. It is important to determine the position in the graph that the user wants to reach and analyze the data received from them to identify which data is missing and within which blocks or child business processes it can be obtained.

The text in the name of an operation, subprocess, or process defines the action that transfers the automaton from one state to another. Additionally, the text in the condition name may indicate the user's intent.

The text in the name of the input and output may also indicate the user's intent. In general, solving the problem of natural language interaction between the user and the bot includes two separate subproblems:

1. Recognition of text and its analysis to determine the user's current state.
2. Recognition of voice, its transformation into text, and subsequently, similarly to point 1, the determination of the user's current state.

The task of determining the user's current state can be solved by identifying key elements that are specific to a particular vertex in the automaton graph and transition criteria.

When creating the bot itself, we can require the user to provide a description and keywords that will characterize the current vertex.

Example 1:

User request for ordering a taxi.

To generate a bot, there is a business process defined as follows:

1. Request for entering the starting point.
2. Request for entering the destination point.
3. Request for selecting the car class.
4. Displaying trip information.
5. Payment for the trip.

In the case of points 1-4, the user can provide this information in a single text message. For points 1-4, keywords, characteristics, or, for example, variable names and their descriptions that need to be filled in can be added.

A more complex example is a banking system.

As a user, I want to receive a statement for a credit account. Accordingly, as a user, I provide certain information in the form of a text message or a voice message. Next, it is necessary to determine the vertex whose result will be obtaining the account statement and the vertex responsible for retrieving account information, identify the completed and uncompleted steps, and generate the necessary questions to complete the business process.

For chatbots, Telegram is usually used. It is necessary to work with the names of all elements of the operation, subprocess, and process descriptions: action, result, input, condition.

The classical order:

1. Conduct morphological and syntactic text analysis;
2. Identify the entities operated by the text;
3. Map the text to ontological entities;
4. Depending on the context, create an instance of the ontological element;
5. Represent the entities interacting with each other.

However, in the context of chatbot interaction, a different approach is required, focused on utilizing accumulated knowledge and training the system.

Synthesis of formal models for chatbot representation

It is evident that the most crucial aspects from both theoretical and implementation perspectives include the selection of a formal chatbot model, the development of efficient algorithms for chatbot model synthesis, and the creation of algorithms for representing synthesized models within well-known messaging platforms.

When addressing this problem, it is essential to consider several widely recognized concepts, primarily well-established automata theory models, methods for their analysis and synthesis, and their connection to formal languages. Automata theory models and their synthesis and analysis methods, which are widely used for describing system behavior, can provide insights for solving our problem.

An analysis of automata theory capabilities has demonstrated the feasibility of using transition systems as

chatbot models. A similar problem can also be solved by applying finite-state automata (FSA) analysis and synthesis algorithms, where the synthesis algorithm utilizes a graphical interpretation of a regular expression.

Algorithm for synthesizing a chatbot based on desired behavior lines

Let L be a given set of regular expressions that define behavior lines, or in the context of a chatbot, scenarios.

It is necessary to synthesize a Mealy automaton, that is, to establish a one-to-one correspondence between the linear notation of a regular expression and a specific graphical configuration [11]. To achieve this, we define graphical interpretations for the fundamental operations in the input language L, which in our case represents regular expressions defining the desired behavior lines for the automaton or the desired chatbot scenarios.

1. We define the graphical interpretation for a single-letter language.

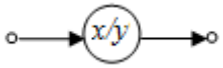

Fig. 1. Configuration for a single-letter language

2. We define the graphical interpretation for the empty word.

Fig. 2. Configuration for the empty word

3. We define the graphical interpretation for the disjunction operation of two languages $L_1 \cup L_2$ and two single-letter languages.

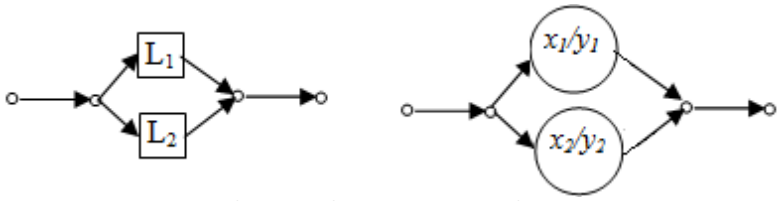

Fig. 3. Configuration for the disjunction of two languages

4. We define the graphical interpretation for the concatenation operation (multiplication) of two languages L_1 and L_2.

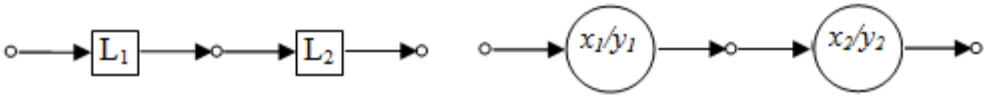

Fig. 4. Configuration for the concatenation of two languages

5. We define the graphical interpretation for the iteration of language L. The iteration of language L is represented as L*.

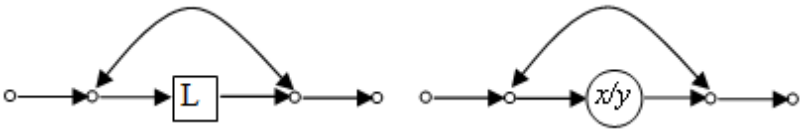

Fig. 5. Configuration for the iteration of a language

Using steps 1-5, we define the steps for transitioning from the graphical configuration of the input language L to the automaton with its states and transitions.

Step 1. Construct a graphical configuration for the expression $0Le_0$, where 0 and e_0 are not alphabet symbols.

Step 2. Assign natural numbers to all vertices labeled with alphabet symbols from the expression L.

Step 3. The states of the automaton correspond to the vertices that have been assigned numbers, as well as 0 and e0.

Step 4. All edges in the graph that connect states are labeled with the symbols corresponding to the respective vertex labels. The arcs and paths in the graph define the transition function. Construct the automaton for the graph obtained in Step 1.

Step 5. Determinize the automaton obtained in this way.

As shown in [11], if a set of regular expressions is given by equation (2):

$$L = \{L_1, L_2, \dots, L_n\} \quad (2)$$

in the same alphabet X, then the process of synthesizing an automaton that accepts the language defined by these expressions can be reduced to a single regular expression, and Steps 1-5 can be applied to obtain the automaton.

Algorithms for constructing a finite-state automaton for implementing chatbots based on well-known messengers

Let a set of regular expressions (2) for the chatbot be given. It is necessary to transform a certain set of regular expressions L into a state machine representation for the Telegram messenger.

It is necessary to define and specify the key alphabet of input and output words.

Set of input words: i0 – i10.

• i0: Opening the chatbot.
• i1: Selecting the departure location.
• i2: Selecting the destination location.
• i3: Selecting the car type.
• i4: Confirming the order.
• i5: Searching for a car.
• i6: Confirming the found car.
• i7: Completing the order.
• i8: Payment not available.
• i9: Changing the final destination.
• i10: Order canceled.

Set of output words: o0 – o10.

• o0: Chatbot opened.
• o1: Selected departure location saved.
• o2: Selected destination location saved.
• o3: Selected car type saved.
• o4: Confirmation received.
• o5: Car search started.
• o6: Car found and confirmed.
• o7: Order completed.
• o8: Notification of payment unavailability.
• o9: Need to change the final destination.
• o10: Trip canceled.

1. Alphabet definition

The alphabet consists of a set of possible input signals (input words) and corresponding output signals (output words). They are denoted as:

• I = {i0, i1, i2, i3, i4, i5, i6, i7, i8, i9, i10} — set of input signals.
• O = {o0, o1, o2, o3, o4, o5, o6, o7, o8, o9, o10} — set of output signals.

2. Definition of the main process

The main process can be described as a sequence of regular expressions corresponding to the correct user actions:

L_1 = i1·o1·i2·o2·i3·o3·i4·o4·i5·o5·i6·o6·i7·o7

This expression describes the main sequence of actions: selecting the departure location, destination, car type, confirming the order, searching for and confirming the car, and completing the order.

3. Adding negative scenarios

To account for negative scenarios, we add alternative expressions:

1. Payment unavailability:

L_2 = i4·o4·i8·o8·i3·o3

This expression describes a scenario where, after confirming the order, a payment issue arises, and the user returns to selecting the car type.

2. Changing the final destination:

L_3 = i5·o5·i9·o9·i2·o2

This expression describes a scenario where, during the car search, the user changes the final destination and returns to selecting the destination location.

3. Trip cancellation:

L_4 = (i4·o4·i10·o10) ∨(i5·o5·i10·o10) ∨(i6·o6·i10·o10) ∨(i7·o7·i10·o10)

This expression describes scenarios where the user cancels the order at any stage after confirming the

order, searching for a car, or confirming the car.

4. Combination of all scenarios

The general regular expression describing all possible scenarios, including both the main and negative ones, can be represented as:

$$L = L_1 \vee L_2 \vee L_3 \vee L_4$$

This expression covers all possible variations of the taxi ordering process, including the standard scenario and negative scenarios with returns to previous stages.

Main scenarios:

1. Selecting the departure location:
(i1, f(i1)) = (i1, S1)
After entering the departure location p = i1, the automaton transitions to state S1.
2. Selecting the destination location:
(i2, f(i2)) = (i2, S2)
After entering the destination location p = i2, the automaton transitions to state S2.
3. Selecting the car type:
(i3, f(i3)) = (i3, S3)
After selecting the car type p = i3, the automaton transitions to state S3.
4. Confirming the order:
(i4, f(i4)) = (i4, S4)
After confirming the order p = i4, the automaton transitions to state S4.
5. Searching for a car:
(i5, f(i5)) = (i5, S5)
After initiating the car search p = i5, the automaton transitions to state S5.
6. Confirming the found car:
i6, f(i6)) = (i6, S6)
After confirming the found car p = i6, the automaton transitions to state S6.
7. Completing the order:
(i7, f(i7)) = (i7, S7)
After completing the order p = i7, the automaton transitions to state S7.

Negative Scenarios:

1. Payment unavailability or selecting a different car type:
(i8, f(i8)) = (i8, S3)
If payment is unavailable p = i8, the automaton returns to state S3 to select a different car type or payment method.
2. Changing the final destination:
(i9, f(i9)) = (i9, S2)
If the user changes the final destination p = i_9, the automaton returns to state S_2 to update the destination information.
3. Trip cancellation:
(i10, f(i10)) = (i10, S1)
If the user cancels the trip p = i10, the automaton returns to state S1.
Thus, we obtain the combined transition and output table for the Mealy automaton (Table 1).

Table 1

Table of transitions and outputs of the Milli automaton

State	Intup signal	Next state	Output signal
S0	i0	S1	o0
S1	i1	S2	o1
S2	i2	S3	o2
S3	i3	S4	o3
S4	i4	S5	o4
S4	i8	S3	o8
S4	i10	S1	o10
S5	i5	S6	o5
S5	i10	S1	o10
S5	i9	S2	o9
S6	i6	S7	o6
S6	i10	S1	o10
S7	i7	End Event	o7
S7	i10	S1	o10

Graphical result of the synthesis (Fig. 6).

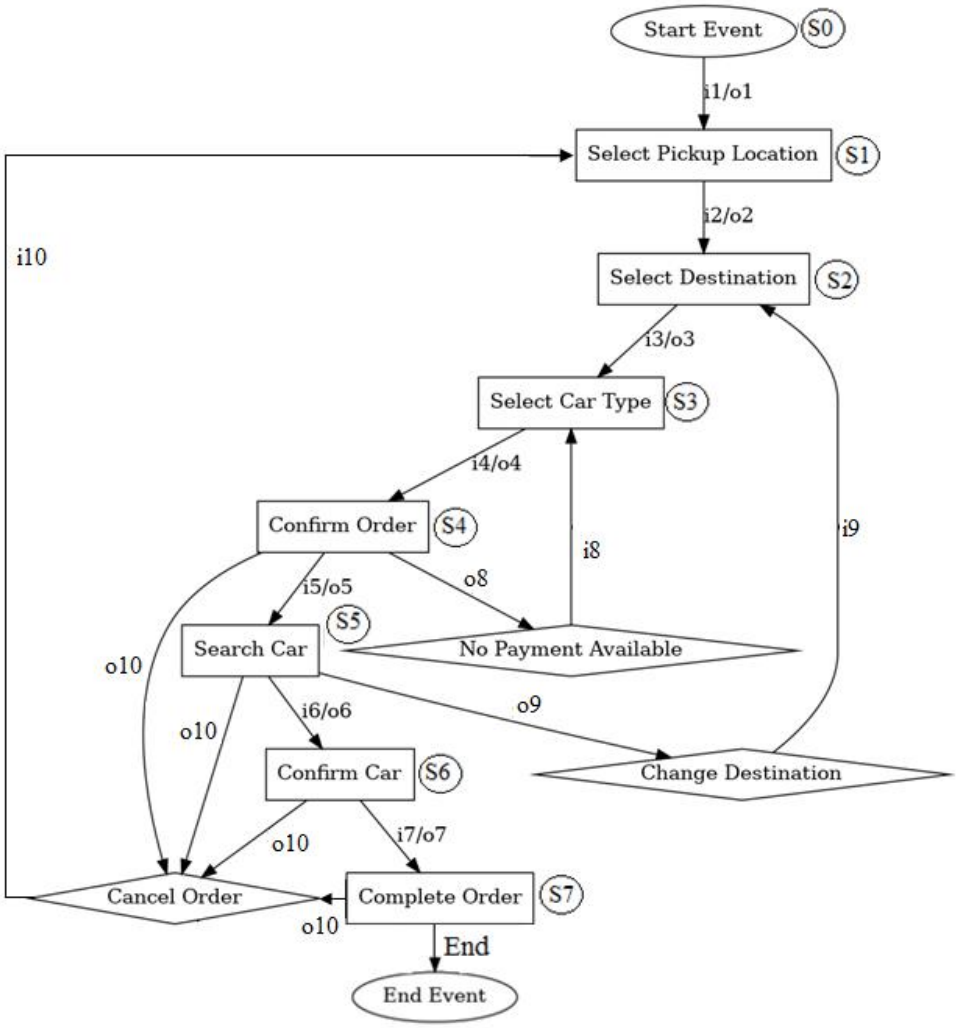

Fig. 6. State Transition diagram of the Mealy automaton for the "Taxi ordering" task

Features of using the Mealy automaton and considering time constraints

Since all events are inherently linked to the time at which they occur, interact within a single scenario, exchange messages, and return responses, the model can be extended by incorporating time and the sequence of message generation and responses. The time tis required for the system to generate messages for the user ii must also be considered, as it is directly related to the interaction of the system with thirdparty web services that the application needs to operate with. By taking Fig. 6 as the basis and applying the fundamental concepts of sequence diagrams, we obtain the Time Sequence Diagram for the "Taxi Ordering" Task (Fig. 7).

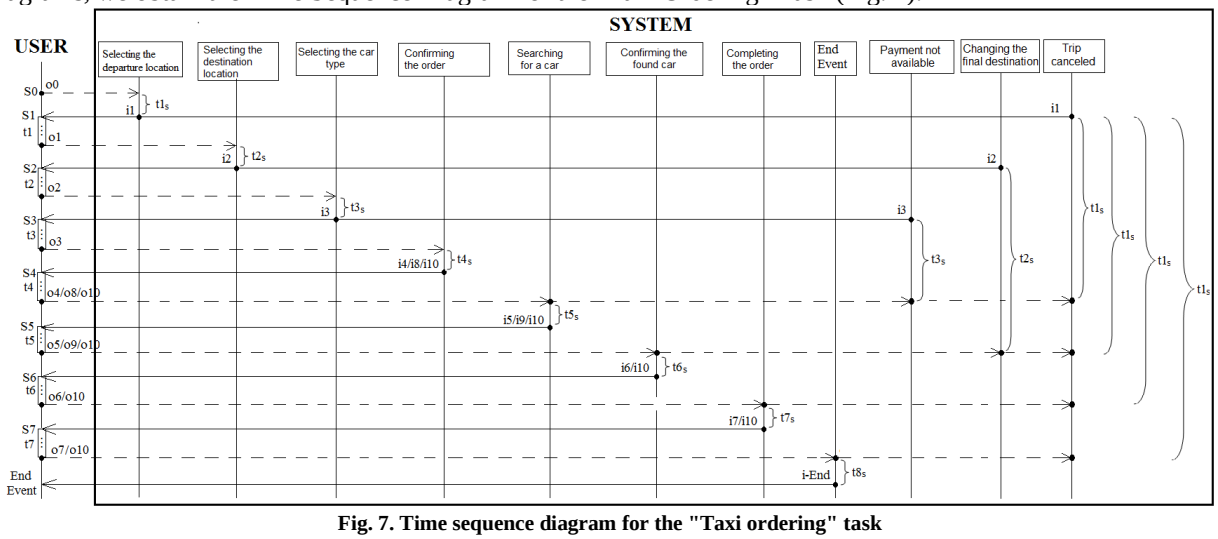

Fig. 7. Time sequence diagram for the "Taxi ordering" task

Now, for any scenario described in the Mealy diagram (Fig. 6), its execution time can be determined. For example, taking the standard scenario, its execution time will be: T1 = t1 + t2 + t3 + t4 + t5 + t6 + t7 +t1s + t2s + t3s + t4s + t5s + t6s + t7s + t8 (s). If the user wants to change the car type in the scenario, the execution time will be: T2 = t1 + t2 + t3 + t4 + t3 + t4 + t5 + t6 + t7 + t1s + t2s + t3s + t4s + t3s + t4s + t5s +t6s + t7s + t8 (s). A scenario where the final destination is changed can be executed in: T3=t1+t2+t3+t4+t5+t2+t3+t4+t5+t6+t7+t1s+t2s+t3s+t4s+t5s+t2s+t3s+t4s+t5s+t6s+t7s+t8 (s). The "Trip Cancellation" scenario at the order confirmation stage will execute in: T4=t1+t2+t3+t4+t1s+t2s+t3s+t4 (s).

Results of the experimental study

Based on the conducted synthesis, a ready-made solution was developed in the form of a Mealy automaton and a BPMN diagram for a Telegram bot (Fig. 8). This was then used to create a Telegram bot by leveraging the existing messenger API and determining the execution time for various scenarios in the "Taxi Ordering" task (see Figs. 9-12).

Since bot construction involves not only states and transition conditions but also interactions with users and third-party web services, the existing chatbot generation system employs BPMN diagrams, as described in [2]. For the "Taxi Ordering" task, the process model diagram, created by the user to solve the current problem within the bot, will have the following structure (Fig. 8).

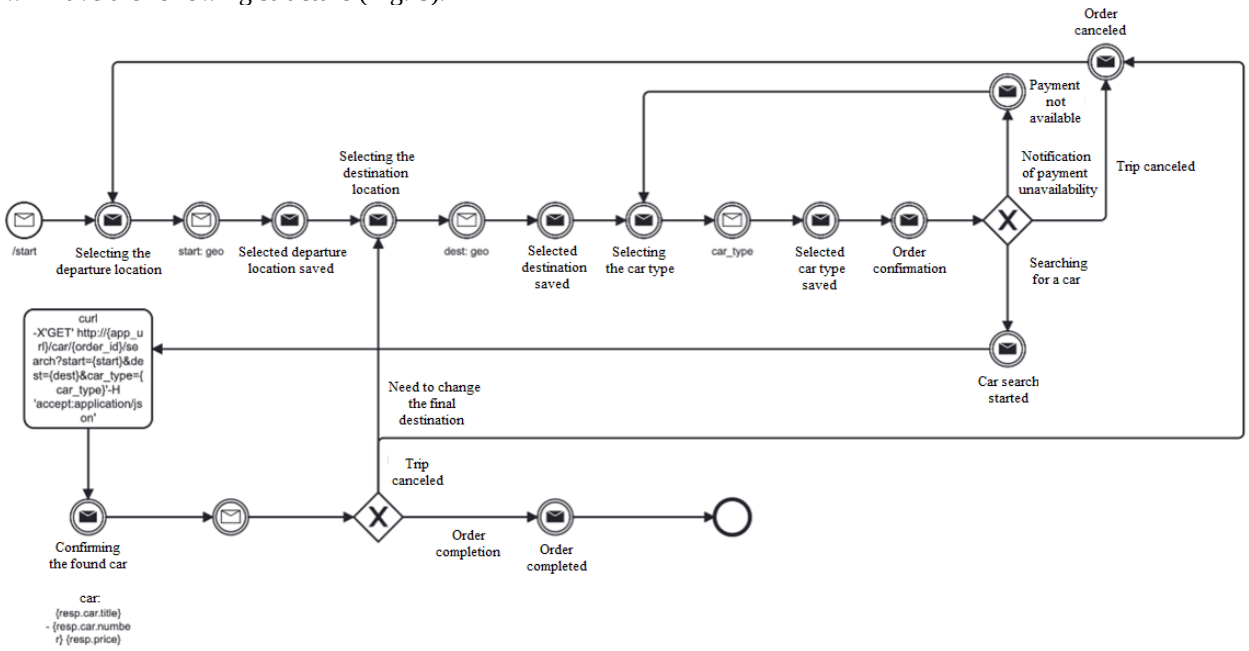

Fig. 8. BPMN - diagram for chatbot creation

A key aspect of the current work is the synthesis of states based on input and output words, which ensures the construction of a draft BPMN diagram with all exits and states. However, as of now, the user creating the chatbot will need to manually define the text of the request sent to the client, determine data sources for retrieving information on the user's geolocation, available car classes, and route cost calculation. This is because the Telegram bot serves as an interaction element with the user and must integrate with existing web services within the system architecture.

In Fig. 9-12, the time was determined by receiving a response from the data source (web service) on the available car types and the actual availability of the selected car in less than 20 seconds.

Thus, for the taxi-ordering Telegram bot, the following external information is required: the starting and ending geolocation of the route, available car types for ordering and searching, trip booking, and payment processing. These elements should be referred to as metadata for Telegram bot synthesis. Metadata cannot be synthesized or automatically assigned. As a result, it must be provided manually by the user utilizing the software solution for chatbot automation. A similar situation applies to messages used in the chatbot's communication with end users. The chatbot synthesis process may determine that at a given stage, a question needs to be asked to the end user, and a response needs to be received. However, the actual text of the question will require manual adjustment. Additionally, certain scenarios require that the user's response undergo further processing within the system with which the final Telegram bot is integrated. Such elements include: generating a payment system link for order payment; receiving a message from the payment system confirming whether the order was successfully paid or not; changing the final destination of the route. In other words, the need for metadata in Telegram bot generation necessitates the inclusion of a BPMN diagram as a fundamental component of the Telegram bot synthesis process in the proposed software solution.

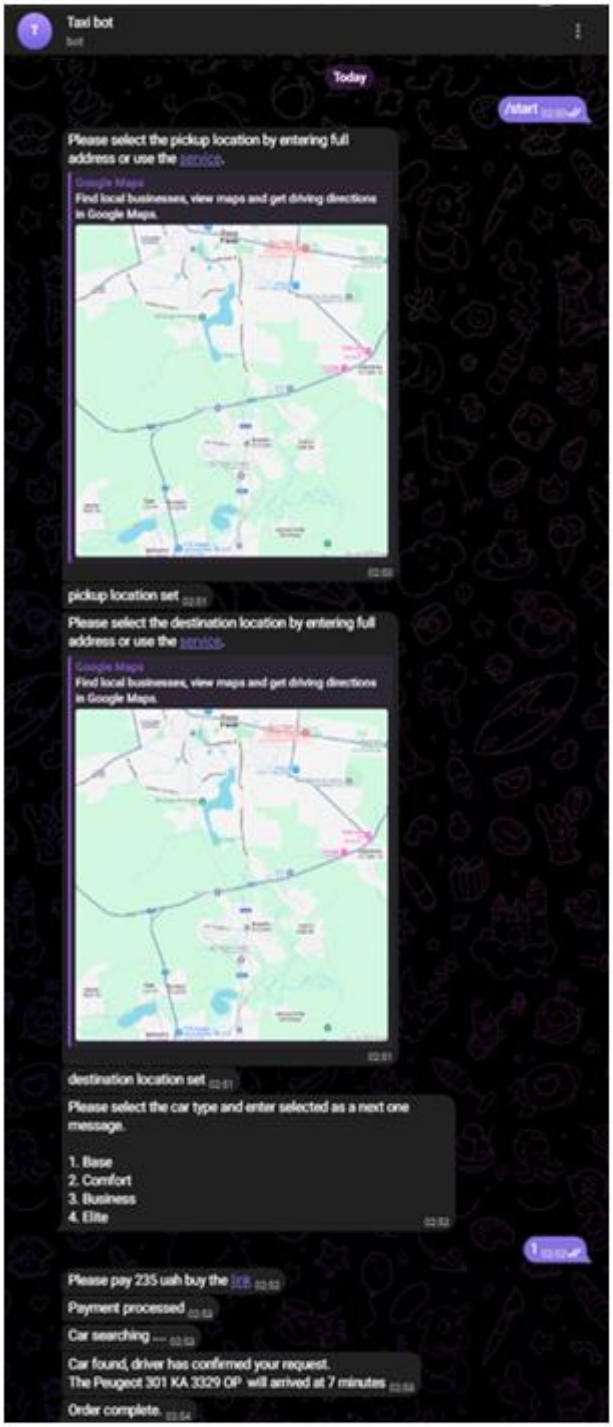

Fig. 9. Standard scenario with execution time T1 =204(s)

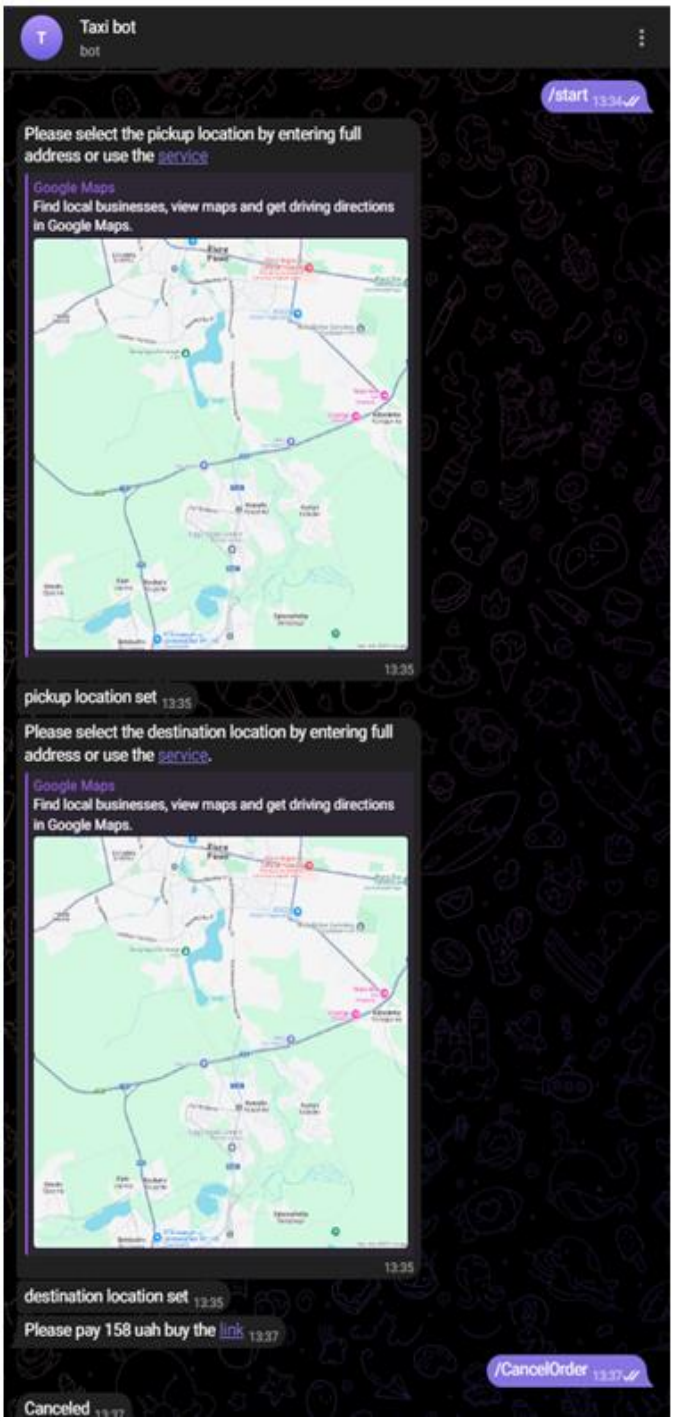

Fig. 10. "Trip Cancellation" scenario. T4 = 80(s)

If we take a closer look at the system's algorithm for synthesizing Telegram bots, it consists of the following steps:

1. defining the sets of input and output words;
2. synthesizing a Mealy-Moore automaton based on the sets of input and output words;
3. generating a BPMN diagram with all states and transitions;
4. providing metadata for retrieving information from the system and the user;
5. creating the Telegram bot.

Let’s examine in more detail the transitions between steps 2-3, 3-4, and 4-5.

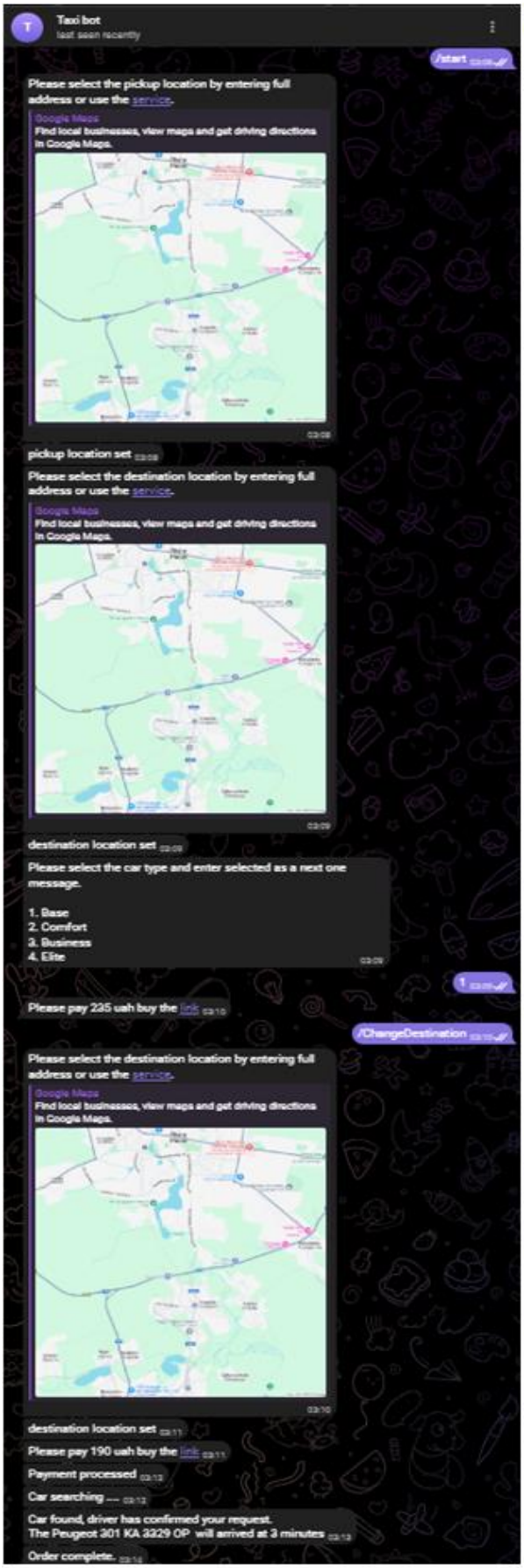

Fig. 11. Scenario with a change in the final destination with execution time T3 = 467(s)

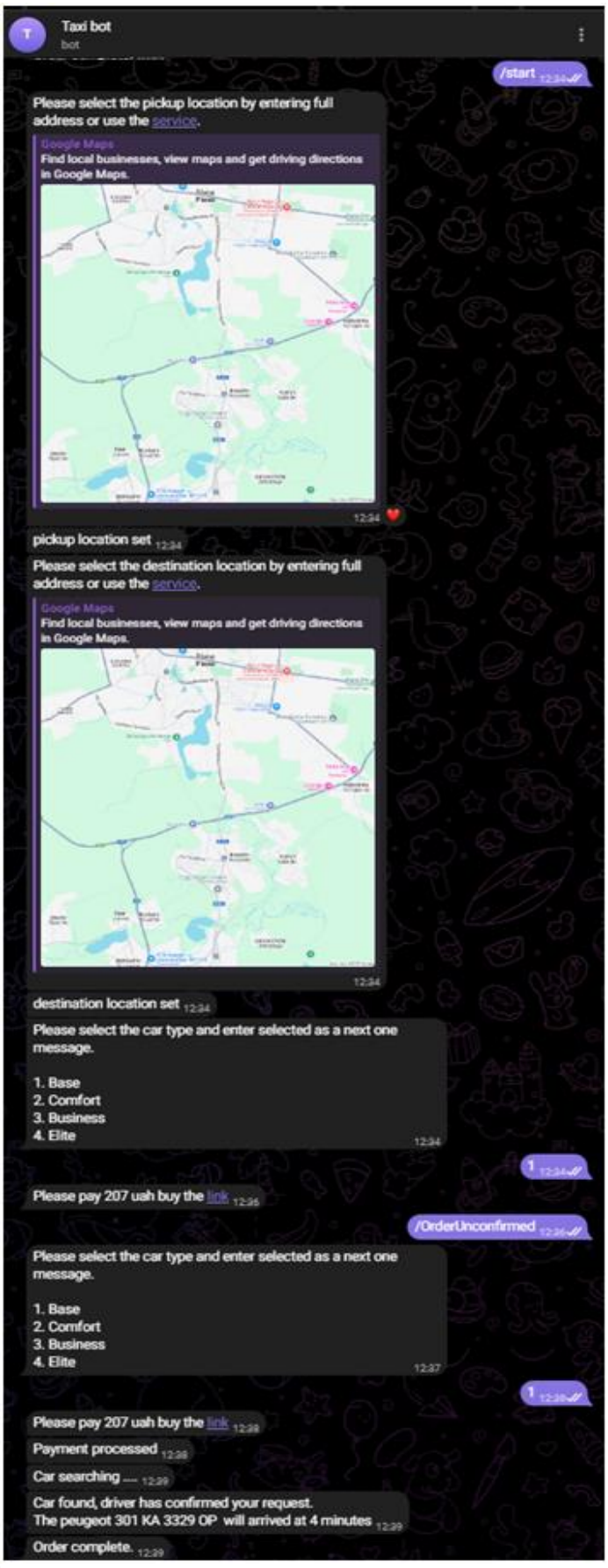

Fig. 12. Scenario with a change in car type with execution time T2 = 324(s)

After synthesizing the Mealy-Moore automaton based on the sets of input and output words, as described in this paper, the system obtains this automaton in the form of a .json file, which contains vertices and a description of transitions as an array, specifying the transition, transition condition, and the destination vertex. This automaton defines all necessary states, points of communication with the user, and corresponding transition conditions. However, as mentioned earlier, this automaton does not include metadata about the system with which it will interact. This aspect is the key reason why the automaton is converted into a BPMN diagram using system tools.

During the transition from step 3 to 4, the obtained BPMN diagram contains all transition conditions and states but requires user intervention to add metadata. For example, in the case of car search, executing this scenario requires making an API request, receiving a response from the API, processing it, and then deciding whether to transition to the next state. The result of step 4 is the completion of all metadata, which is necessary for the chatbot to function correctly within the existing system, in this case, the taxi system.

As a result of the 4-5 transition, we obtain a fully generated chatbot, created by our system, based on the synthesized Mealy-Moore automaton, which was initially automatically converted into a BPMN diagram, with added metadata describing the system with which the Telegram bot will interact.

Conclusions

The conducted research has led to several conclusions that are useful for systematizing studies in this fascinating field.

First, this paper proposes and implements a new approach to building chatbot applications as part of a large information system. The approach is based on the use of automata theory, methods for their analysis and synthesis, and their connection with formal languages. Based on this, a state diagram was developed, implementing the operation of a Mealy finite-state automaton, obtained as a result of transforming a set of regular expressions L, which define the desired system behavior. Subsequently, using a third-party library, the Mealy automaton was transformed into a BPMN diagram, which was used to create a chatbot application that visualizes the user's interaction with the system. By taking into account the time and sequence of message creation and responses, an approach for calculating chatbot execution time for different scenarios was also proposed. It was determined that for standard scenarios, the execution times are: T1 = 204 (s), T2 = 324 (s), T3 = 467 (s), T3 = 80 (s).

Second, the use of this approach allows chatbot developers to modify, generalize, and refine the chatbot system model during system operation by adjusting the internal representation of information. This is achieved by modifying the key alphabet of input and output words, which inevitably leads to changes in the set of regular expressions L and the state diagram that serves as the foundation for chatbot generation.

Third, the proposed approach is characterized by its relative ease of adaptation to other languages.

Among the drawbacks, the following can be noted: the processing algorithm becomes more complex, and there is a need for a developed ontology.

References

1. Chymshyr V., Telenyk S., Rolik O., Zharikov E. The platform for supporting the lifecycle of services in the information systems of information and communication service providers. *Adaptive systems of automatic control*. 2023. Vol. 1, № 42. P. 205–226. https://doi.org/10.20535/1560-8956.42.2023.279172.
2. Telenyk S., Novakovskyi H., Vovk Ye., Anosov A. Rozvytok i realizatsiia tekhnolohii stvorennia shyrokoho klasu zastosuvan na zrazok chat-botiv na osnovi formalnykh modelei. *Naukovi zapysky NaUKMA. Kompiuterni nauky*. 2022. - T. 5. - S. 97-107. - https://doi.org/10.18523/2617-3808.2022.5.97-107.
3. Telenyk S., Nowakowski G., Vovk Y. Conceptual Foundations of the Use of Formal Models and Methods for the Rapid Creation of Web Applications. *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS2019)*. Metz, France, 2019. P. 305–313.
4. Telenyk S. F., Pohorilyi S. D., Kramov A.A., Vovk Ye. A. Vyiavlennia namiriv korystuvacha pry spilkuvanni z botom. *Reiestratsiia, zberihannia i obrobka danykh*. 2023, T. 25, № 2. Stor. 10-26. ISSN 1560-9189.
5. Hobson L., Hannes H., Howard C. Natural Language Processing in Action: Understanding, analyzing, and generating text with Python. 1st Edition. Publisher: Manning Publications, 2019. 544 p. ISBN-10: 9781617294631.
6. Szymon R. Practical Bot Development: Designing and Building Bots with Node.js and Microsoft Bot Framework. Publisher: Apress, 2018. 671 p. ISBN: 9781484235409.
7. Kai W. Designing Autonomous AI Agents with LangGraph (Explore the latest advancement in AI from the fundamentals of Crew AI, langGraph and langchain). Publisher: Independently published, 2024. 171 p. ISBN-13: 979-8301134241.
8. Kai W. LangGraph.js Handbook: A Practical Guide to Building LLM-Powered Applications in JavaScript (Explore the latest advancement in AI from the fundamentals of Crew AI, LangGraph and LangChain). Publisher: Independently published, 2024. 229 p. ISBN-13: 979-8302223234.
9. Boiko Yu.V., Kryvyi S.L., Pohorilyi S.D. ta in. Metody ta novitni pidkhody do proektuvannia, upravlinnia i zastosuvannia vysokoproduktyvnykh IT-infrastruktur. Kiyv: VPTs «Kyivskyi universytet». 2016. 447 s.
10. Kryvyi S.L., Pohorilyi S.D., Slynko M.S. Formalizovanyi metod proektuvannia zastosuvan v tekhnolohii GPGPU. *Problemy prohramuvannia*. 2018. № 2-3. S. 12-20.
11. Kryvyi S. L. Skinchenni avtomaty: teoriia, alhorytmy, skladnist: pidruchnyk dlia studentiv vyshchykh navchalnykh zakladiv / pid. zah. red. Palahina O.V. Kyiv-Chernivtsi: Bukrek, 2020. 428 s. ISBN 978-617-7770-45-8.

Yevhenii Vovk **Євгеній Вовк**	Assistant at the Department of Computer Science and Software Engineering, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» https://orcid.org/0000-0001-8667-6893 e-mail: vovkzenia@gmail.com	Асистент кафедри інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»
Juliya Polupan **Юлія Полупан**	Candidate of Technical Sciences, Associated Professor at the Department of Computer Science and Software Engineering, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» https://orcid.org/0009-0000-0243-824X e-mail: juliy_polupan@i.ua	Кандидат технічних наук, доцент, доцент кафедри інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»