DESIGN AND DEVELOPMENT OF A WEB-BASED GRAPH EDITOR AND SIMULATOR APPLICATION

PÉTER MILEFF
University of Miskolc, Hungary
Institute of Information Technology
peter.mileff@uni-miskolc.hu

Abstract. In today's world, data, the optimization of business processes, and artificial intelligence are becoming increasingly important. With the help of continuously evolving modern IT tools, numerous new opportunities have opened up for companies in recent years. However, the availability of data is not always straightforward. IT systems must be prepared to ensure that each component can provide the appropriate output, from which important, logically higher-level interconnected information can later be extracted through some form of transformation. This publication deals with the graph-based description of business processes and the real simulation of the process descriptors obtained in this way. A graph design application operating in a web environment has been implemented, and we will present its structure and most important requirements. With the help of the designed software, data sets corresponding to process descriptions can be generated, which can then serve as input for various process mining and other tools.

Keywords: process mining, RPA, graph simulator

1. Introduction

Modern companies today often use complex IT systems to support their administrative processes. Over time, the growth of the company results in increased complexity of these processes. Transparency and comprehensibility decrease, and unnoticed bottlenecks can often develop. Managing such complex systems raises more and more issues, both from a security perspective and in terms of process traceability. Recent developments in User Activity Monitoring (UAM) and Robotic Process Automation (RPA) solutions aim to address these problems [20]. One important goal articulated by our research project is to be able to predict certain events based on activity logs extracted from some information system, using an effectively configurable procedure and a model employing artificial neural networks. A key question in the research and solution is the appropriate dataset, as it enables the design of the subsequent processes and learning algorithms, and ensures proper training.

During our research work, it can unfortunately be stated that obtaining high-quality data is not easy. Companies generally cannot publish data because their processes contain sensitive information. However, in order to train a neural network for prediction in a sample system and to conduct experiments/measurements with it, some form of dataset is absolutely necessary. Moreover, the dataset must contain a

substantial amount and quality of samples to ensure that the network can be trained effectively. Therefore, within the project, the question of data generation naturally arose. Although generated data generally never reaches the quality of a real dataset, it is nonetheless the best way to develop and test various algorithmic design alternatives and different neural network models. Since the volume of such a dataset needs to be large with numerous transactions, manual creation is not feasible. Data generation is definitely required.

Continuing along the same lines, one of our important goals within the project related to this topic was to design and implement a system capable of generating datasets that are close to reality.

The key requirements for creating the sample system are:

- **Online Operation and Efficient Infrastructure:** The system should be able to operate online, meaning users should be able to describe events and submit them to the processing engine via a browser. It is advisable to use a forward-looking infrastructure that can be easily upgraded in the future.

- **Event Description:** The system should support the schema description of event flows. Users should be able to define and create the process from which data will be generated, using an online solution.

- **Visualization and Validation:** Since we are dealing with complex event graphs, the software should include some visualization options to make processes more comprehensible and event descriptions more effective. Additionally, validation solutions are needed to check the correctness of the graph and ensure the current processing engine can interpret and execute it.

- **Processing Engine:** The central component of the software is responsible for receiving, interpreting, and simulating the event descriptions submitted online. All other listed requirements support the work of this module. While manual parameterization of the module is possible, it would significantly reduce usability, making it very cumbersome to develop predictive algorithms.

- **Data Export:** To evaluate the results of calculations and simulations, it is necessary to export the data in appropriate formats. Supporting multiple formats (e.g., CSV, XES) is advisable, as well as providing the final result back to the client side for the user to download onto their device.

Since it was not possible to fully develop the entire system within the scope of the project, the greatest emphasis was placed on the processing engine and the proper export of data.

2. Robotic Process Automation

Based on an initial review of the literature, Robotic Process Automation (RPA) is characterized as the use of specific technologies and methodologies, driven by software and algorithms, to automate repetitive tasks traditionally performed by humans [1, 2, 3, 4]. Primarily guided by simple rules and business logic, RPA interacts with various information systems through existing graphical user interfaces [5]. Its primary function is to automate repetitive, rule-based activities using non-invasive software robots, commonly referred to as "bots" [6, 7, 8]. Recently, the definition of RPA has broadened to encompass its integration with

artificial intelligence (AI), cognitive computing, process mining, and data analytics. The advent of advanced digital technologies has shifted the role of RPA from merely handling repetitive and error-prone tasks in business processes to undertaking more complex, knowledge-intensive, and value-adding activities [9, 10, 11].

In assessing the current state of the RPA market, Forrester [12] identified 12 RPA vendors that provide enterprise-level solutions capable of supporting the demands of "shared services" or organization-wide RPA utilities. Although some vendors offer industry-specific solutions, Schmitz et al. [13] argue that the general concept of RPA is not industry-specific. Additionally, partnerships between RPA vendors and leading AI providers have facilitated the expansion of traditional RPA functionalities with new and emerging technologies, such as process discovery-based self-learning, robot training, AI-driven screen recognition, natural language generation, and automated process documentation [9].

A Deloitte survey [14] of 400 companies revealed that the majority have embarked on their RPA journey, with nearly a quarter planning to do so within the next two years. The survey also found that payback periods average around a year, with companies meeting or exceeding their expectations for cost reduction, accuracy, timeliness, flexibility, and compliance [14]. Forrester [12] projects that by 2021, over 4 million robots will be automating repetitive tasks, with the focus increasingly shifting toward AI integration and enhancements in RPA analytics. Similarly, Everest Group [15] notes that while most buyers are satisfied with RPA solutions, there is a growing demand for improved analytics and cognitive capabilities.

Despite the significant benefits of RPA, only 5% of companies in the Deloitte study [14] have deployed more than 50 robots in their operations. The success of RPA projects hinges on organizational capability and a clear understanding of business objectives for RPA implementation. Key challenges identified include a lack of understanding of RPA's potential applications, insufficient management support, and employee concerns about job security [16]. To address these challenges, a change management strategy, a shift in organizational culture, and a new mindset are recommended to bridge the gap between viewing RPA as merely an IT tool and recognizing its broader business impact [17, 19, 16]. Additionally, participants in the Everest Group study [18] emphasized the importance of strong customer support, comprehensive training and educational resources, RPA maintenance services, and a robust vendor ecosystem for complementary technologies as critical drivers of RPA adoption. Moreover, the introduction of new technologies raises questions about robot management, central control, and governance [12].

3. Software Architecture

Establishing the appropriate research architecture/infrastructure is crucial for the success of the project in terms of research efficiency. A general goal is to strive for a research software architecture that efficiently meets the requirements of the project and research.

A general expectation regarding the software set as a goal is that it should operate in a web environment. Typically, the foundation for achieving this is a client-server architecture. From the project's perspective, this solution meets expectations. Based on this, the schematic description of the system modules is as follows:
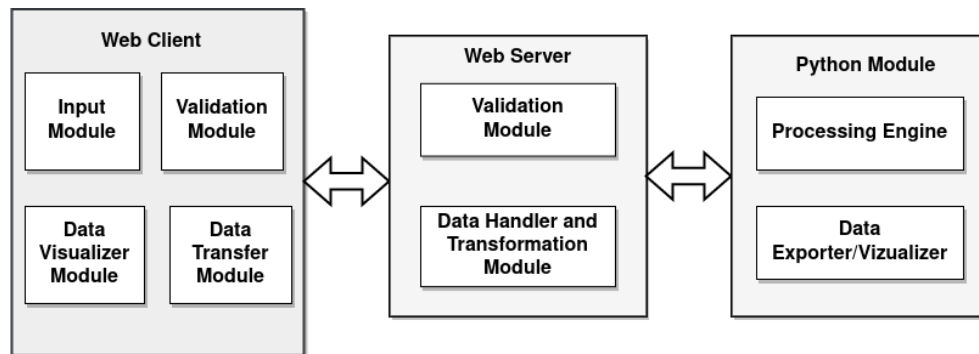
Figure 1. Sample generation system architecture

The diagram clearly shows that the architecture consists of three main parts: client-side, server side, and the processing engine.

<u>The software operates as follows:</u> The user opens the website (client-side) and starts describing a new event graph. This can be done iteratively. Graph creation is broken down into elementary steps; in the background, each modification made on the client-side sends necessary data to the server-side (e.g., creating a new node), where the in-memory (and textual descriptor) model of the graph is built. This model will continuously evolve during editing. Finally, once the user is finished, they can send the graph to the processing engine for the purpose of event set generation. The processing engine interprets the constructed graph model and generates an event set according to the parameters, which is saved in XES and CSV files.
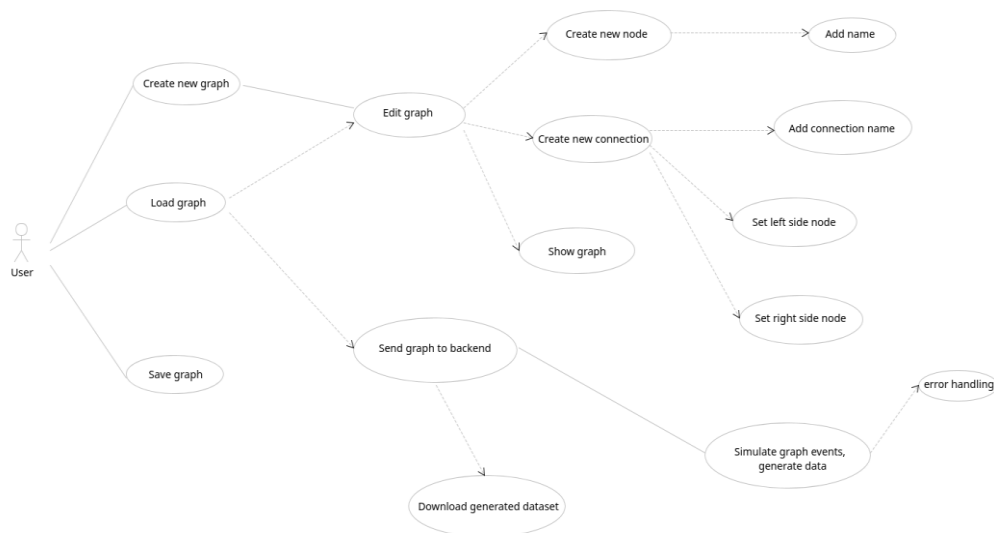
Figure 2. Use-Case Diagram of the graph editor and simulator system

The resulting files can be directly used as input datasets for further investigations. It can serve as input for various process mining and other tools in order to analyze processes. In [21] and [22] authors use event logs and analyze process discovery algorithms of pm4py within numerous benchmarks.

3.1. Client side

On the client side, interactive event input is displayed, which can be visualized and sent to the server for processing. From a technology perspective, it is recommended to use a modern JavaScript or Dart-based framework. Typically, these include *Angular*, *ReactJS*, *Vue.js*, and *Flutter*. With their help, the necessary modules for the client-side can be efficiently developed.

Four important modules can be highlighted:

Input Module: The input module allows the user to describe the elemental events of a specific process using a graphical user interface. It is important that the process elements can be entered step-by-step, as parameterization and other data input for certain events are expected. Two special event types are modeled: so-called AND and OR type branches.

It is essential that the data sent to the server-side engine is also prepared in a well-structured textual format. Essentially, the input module generates this "file" using various input boxes and other elements. The advantage of the resulting textual descriptor is that it can be well stored as a file and reused later. Multiple such formats will be presented in the future.

Basic requirements for creating an event descriptor graph:

- Creating a new graph
- Editing existing graphs
- Iteratively adding and removing elements
- Deleting the entire graph
- Managing node attributes
- Basic error handling

Validation module: Describing the events of a process will result in a graph. Since our goal includes modeling more complex processes, human validation alone is no longer sufficient for larger graphs. This module aids in this process by identifying basic issues before reaching the server, ensuring that event generation can proceed without encountering fundamental problems or generating events based on an invalid graph structure.

Basic validations:
- Syntax correctness verification
- Existence of initial and final states in the graph
- Detection of cyclic paths
- Validation of provided parameters

Data visualization module: In any software of similar nature, visualization naturally emerges as a requirement. Human visual perception plays a significant role in comprehending and understanding complex processes when we can visually inspect the generated event set. In the central repository of recommended JavaScript-based technologies, there are packages available (e.g., *d3*) that are capable of displaying these graphs. One potential issue arises when modeling events that can be executed in parallel. These are known as AND branches. It needs to be examined whether the available packages are capable of creating custom graphs.

Data handler module: Finally, the data handler module's task is to transmit and receive data towards the server. It maintains communication with the server and, if

necessary, transforms the created graph into a format suitable for the Python processing engine. Recommended communication format includes REST API, JSON-based data transfer.

3.1.1. The Angular framework

AngularJS is an open-source JavaScript framework developed by Google for creating dynamic web applications. It greatly simplifies frontend development of web applications. With AngularJS, the toolbox of HTML expands, and the components of applications are more clearly separated. Thanks to Angular's data binding and dependency injection, a lot of unnecessary boilerplate code can be eliminated.

The main objectives of the framework are:

- Ideal for defining the interface with declarative description (HTML), while imperative programming is excellent for expressing business logic.
- Separate DOM (Document Object Model) manipulation from application logic.
- Testing the program is as critical as writing it.
- Complete separation of client and server-side applications.

One of the biggest advantages of the Angular framework is that due to its relatively structured nature, coding occurs within defined frameworks, thereby reducing the chances of errors. Its main components are components and modules. Components are the elements visible to end-users and are often reusable (for example, headers and footers are typically the same on every page). Modules group related components and declare which components can be used by other modules. Like with any comprehensive framework, Angular also has ready-made extension modules available. These make it easy to expand the basic framework toolkit.

3.1.1.1 Two-way data binding

Most template systems only support one-way data binding, combining template and model components into a view. Once integrated, changes made in the model do not automatically reflect in the view. Even more problematic, changes in the view are not propagated back to the model. To avoid inconsistency, developers need to ensure continuous synchronization. Angular, however, adopts a different approach. Firstly, the template compiles in the browser (raw HTML augmented with Angular directives). Unlike the former case, the compilation result is not static. Consequently, any change in the view immediately updates the model, and changes in the model propagate back to the view. This ensures that the state is solely stored in the model (single-source-of-truth), greatly easing the developer's life. Essentially, the view can be seen as a projection of the model. This approach isolates the controller, thereby promoting testability.

Two-Way Data Binding
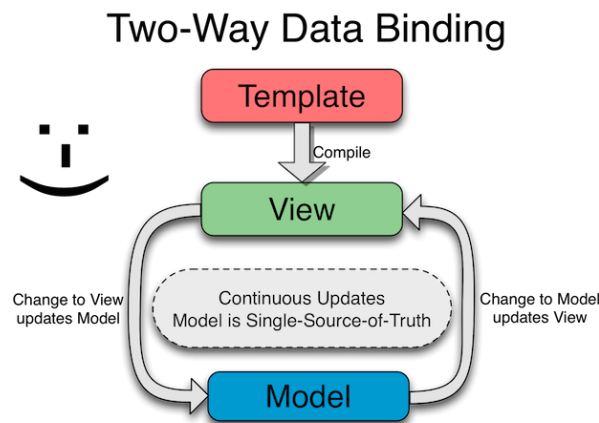
Figure 3. Two-way data binding support architecture [23]

3.1.2. The Angular framework

Input data and the graphical representation of the specified process model are crucial. It's advisable to make visualization continuously available and visible. The reason is that process models are typically developed iteratively, so it's beneficial to display intermediate, partially complete models. This allows creators to receive immediate feedback, enhancing the clarity of the evolving model. Text-based models are not always easy to comprehend, hence visual representation aids understanding.

Numerous libraries are available online for this purpose, with the recommended solution being D3.js [24]. D3 is a JavaScript library for manipulating documents based on data. Its name, Data-Driven Documents, reflects this. D3.js is a dynamic, interactive framework for online data visualization widely used on various websites. It's primarily designed for creating different types of interactive charts and stands out for its ability to create even the most complex graphics relatively easily. Additionally, it supports extensions that enable the visualization of DOT formats, which is advantageous for the current objectives.

3.2. Server Side

The server-side's role is to act as an intermediary layer between the client and the processing engine. The actual model of the graph is constructed here. It receives graph-building instructions from the client-side: creating nodes, deleting nodes, establishing connections, etc. Each function has its own endpoint through which the graph model can be constructed.

Arguments in favor of building the graph on the server-side include the following: Since both the server-side module and the processing module are implemented in Python, passing data between them, i.e., transmitting data from one to the other, can be easily achieved. On the server-side, the model can be stored directly in Python data structures, facilitating its direct transfer to the processing engine.

Server-side implementation is typically advisable as a lightweight environment because it performs less complex tasks. It offers endpoints to both sides and transforms data as well as forwards it in the required direction.

Two main areas should be distinguished:

- **Validation module:** naturally, it verifies the quality of the received data.

The task differs between data received from the client side and data received from the processing engine.

- **Data transmission and transformation module:** it transforms the transmitted data into the required format as needed. If both sides adequately prepare the data, its tasks are minimal.

It is important to note that if the goal is to generate a large amount of data, considering the possibility of asynchronous operation is advisable. Since the Python processing engine is likely to work for a longer period, the client side may not be able to wait for so long due to synchronous HTTP requests. The correct implementation is definitely the asynchronous solution, although this path is somewhat more complex. In this case, on the client side, there needs to be an interface where already created/generated files can be viewed. However, on the server side, management of the files and their status must be implemented. Records must be kept of the generated log files and the original process description graph.

In the initial implementation, the goal is definitely to implement synchronous HTTP requests. Since there is a third module, the processor, it is advisable to implement the server-side implementation in the same environment as the processing engine. In this case, that would be Python.

3.2.1. *Python Processing Engine*

The most popular environment for implementing artificial intelligence computations currently is Python. For this project, Python was chosen as the primary programming language and environment. This decision is easily justified, as Python is an interpreted, general-purpose scripting language with a development history spanning several decades. It is widely used in artificial intelligence, neural networks, and other mathematical computations due to its flexibility and efficiency. Python enables easy and rapid prototyping based on its wide-ranging modules. Virtually every type of problem can be addressed using Python's extensive package ecosystem. It offers rich resources specifically for neural networks as well.

However, using Python also comes with limitations. Since the implementation language of the processing engine is fixed, it is practical to implement the server-side environment in Python as well. However, this is not conventional; other languages might be more suitable for this task. Opting for a different implementation language introduces additional complexity into the current process. In such cases, the server-side environment needs to somehow invoke the Python processing engine, properly parameterized with correct data.

General expectations towards the processing engine:

- **Input validation:** Although validation has been present in previous steps, unfortunately, we cannot overlook it here either. Incorporating some minimal validation is expected and advisable.

- **Parsing, building object model:** Interpretation and parsing of the graph sent in textual format. Based on the textual content, the engine will construct an in-memory structure containing actual objects. Essentially, it maps the content of the textual graph to an object model.

- **Event generation:** Essentially, this is the functionality we need. Based on the constructed object model, the processing engine's task is

to generate data sets considering the specified parameters and attributes, potentially in sizes ranging from several threads to even thousands.

- **Data export:** The generated data initially exists in memory. However, these need to be somehow returned to the server-side first and then to the client-side. Physical storage of the data is necessary for this purpose. During export, the expected functionality is to support at least two formats: CSV and XES.

The processing engine is the core of the software. If it malfunctions, it renders the other components meaningless. Therefore, careful implementation and robust behavior are crucial considerations. It is well-known that Python implementations are not among the fastest in terms of runtime performance; however, speed is not the goal of this project. Likely, computational performance is not a critical requirement, and what matters more is the flexible environment provided by Python with its numerous built-in functionalities.

4. Describing events

One of the key elements of the proposed software is the description and visualization of processes. Using an interactive method, the user can define a process graph. It is advisable to approach the problem by dividing it into two phases. In the first phase, the user can separately create the necessary nodes, associating them with the appropriate data and properties. Then, using descriptive language, the user establishes the connections between the nodes. Finally, the completed graph is visualized on the screen and, if finalized, can be sent to the server side for processing. Below, we present several implementation options.

4.1. GraphML (Graph Markup Language)

GraphML is a comprehensive and user-friendly file format for describing graphs. It consists of a core language that defines the structural properties of the graph, and a flexible extension mechanism for adding application-specific data. Its main features include support for:

- Directed, undirected, and mixed graphs
- Hypergraphs
- Hierarchical graphs
- Graphical representations
- References to external data
- Application-specific attribute data
- Lightweight processors

Unlike other graph formats, GraphML does not use custom syntax. Instead, it is based on XML, making it ideal as a "common denominator" for various services that create, archive, or process graphs.

As an example, let's consider a simple graph:
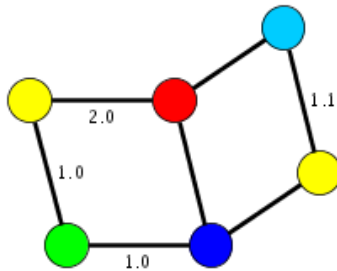
Figure 4. Sample graph

The GraphML description of this is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
          http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <key id="d0" for="node" attr.name="color" attr.type="string">
    <default>yellow</default>
  </key>
  <key id="d1" for="edge" attr.name="weight" attr.type="double"/>
  <graph id="G" edgedefault="undirected">
    <node id="n0">
      <data key="d0">green</data>
    </node>
    <node id="n1"/>
    <node id="n2">
      <data key="d0">blue</data>
    </node>
    <node id="n3">
      <data key="d0">red</data>
    </node>
    <node id="n4"/>
    <node id="n5">
      <data key="d0">turquoise</data>
    </node>
    <edge id="e0" source="n0" target="n2">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e1" source="n0" target="n1">
      <data key="d1">1.0</data>
    </edge>
    <edge id="e2" source="n1" target="n3">
      <data key="d1">2.0</data>
    </edge>
    <edge id="e3" source="n3" target="n2"/>
    <edge id="e4" source="n2" target="n4"/>
    <edge id="e5" source="n3" target="n5"/>
    <edge id="e6" source="n5" target="n4">
      <data key="d1">1.1</data>
    </edge>
  </graph>
</graphml>
```

4.2. JSON Graph format

This JSON Graph Format focuses on conveniently capturing fundamental graph structures. It allows the use of metadata objects within the graph, nodes, and edges, which can be used for any other graph-related data that needs to be managed in graph data files (e.g., graph layout, style, algorithm results, etc.). In recent years, several efforts have been made to create JSON Graph specifications, which are made available on GitHub. The JSON schema is accessible here, providing essential tools to standardize it.

JSON Graph utilizes the JSON schema to specify and validate properly formatted JSON files. A JSON Graph file is not considered valid until it passes validation against the JSON schema specification. The main specification has been designed to be as concise as possible for broad application usage. Sub-specifications can enforce the main specification and accommodate part-specific requirements for JSON Graph data files.

The following example demonstrates a sample graph based on JSON:

```
{
  "graph": {
    "id": "1",
    "type": "weighted network",
    "directed": false,
    "label": "None",
    "nodes": { "a": {}, "b": {}, "c": {}, "d": {}, "e": {}, "x": {} },
    "hyperedges": [
    { "nodes": ["a", "b", "x"], "metadata": { "weight": 17 } },
    { "nodes": ["a", "b"], "metadata": { "weight": 123 } },
    { "nodes": ["c", "d"], "metadata": { "weight": 45 } },
    { "nodes": ["d", "e"], "metadata": { "weight": 46 } }
    ]
  }
}
```

4.3. DOT description language

DOT is a textual graph description language. It provides a simple description of graphs that is readable by both humans and computers. Files in DOT language typically have extensions .gv (or .dot). Since the .dot extension is also used by Microsoft Office, it is preferable to choose the .gv extension. DOT files can be processed by numerous programs. Some of them—such as dot, neato, twopi, circo, fdp, and sfdp—display the parsed DOT file graphically. Others—such as gvpr, gc, accyclic, ccomps, sccmap, and tred—perform computations on the parsed DOT file. Yet others—such as GVedit, KGraphEditor, lefty, dotty, and grappa—provide interactive user interfaces. Most of these listed programs are part of the Graphviz software package or use it in the background.

The DOT format is human-friendly, employing a very "sympathetic" notation system. The format uses a simple text file that is easily editable, even by hand. It supports essential schemas such as directed graphs, subgraphs, clusters, attributes, and more [25].

As an example, let's consider the following graph:
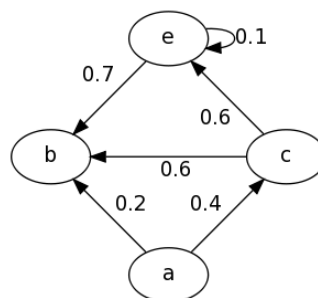
Figure 5. Sample graph

The formulation of the graph in DOT language is as follows:

```
digraph {
    a -> b[label="0.2",weight="0.2"];
    a -> c[label="0.4",weight="0.4"];
    c -> b[label="0.6",weight="0.6"];
    c -> e[label="0.6",weight="0.6"];
    e -> e[label="0.1",weight="0.1"];
    e -> b[label="0.7",weight="0.7"];
}
```

The DOT format is quite popular in practice. Several software tools support their display, and programming languages often provide support through various packages.

For the project, it initially seems like an excellent choice due to its ease of understanding and processing. The only potential headache may arise from parallel AND nodes. One possible solution could be the following concept.

The following diagrams illustrate two different types of graph operations:
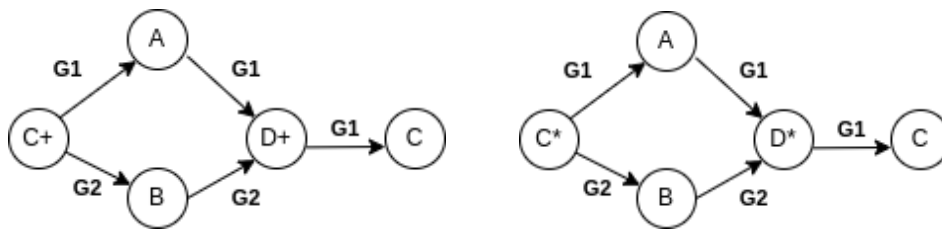
Figure 6. Extending DOT graph with custom symbols (control nodes)

While in the first case we see a sequential, so-called "or" type network, in the second case we have a parallel "and" type. To distinguish between these two types, we introduced special nodes into the notation system. In the diagram, these are represented by the notations C+, D+, C*, and D*, which we refer to as control nodes. The rule we've established for graph representation is as follows: any node marked with a "+" sign indicates an "or" branching or "or" synchronization node. Similarly, if a node is marked with a "*" sign, it signifies an "and" node and "and" synchronization node. With these two elements, complex events as described above can be modeled as graphs.

One of the software's functions is to allow users to define graphs. Another important feature is studying the construction of neural networks and developing various alternatives.

For describing the graph, a pseudo-language similar to DOT can be used:

Table 1. Customized graph description

Sample graph with "OR" relationships:	Sample graph with "AND", parallel relationships:
| ```
graph {

C+ -> A  (G1)
C+ -> B  (G2)
A -> D+  (G1)
B -> D+  (G2)
``` | ```
graph {

C* -> A  (G1)
C* -> B  (G2)
A -> D*  (G1)
B -> D*  (G2)
``` |

```D+ -> C (G1)``` ```}```	```D* -> C (G1)``` ```}```

The description clearly defines the direction of transitions from each node and also specifies which agent performed the step. G1 and G2 indicate who executes the respective transition. So, for example, G1 is Andrew, and G2 is Peter.

4.4. Store graphs

Although the user performs editing on the client side, the graph is assembled on the server side in the background. It is advisable to provide a storage mechanism that is both usable and efficient for this purpose. Since the current software is a prototype, it is not advisable to use a database management system at this stage. Instead, some NoSQL solutions can be effective. After examining various available NoSQL software, the arguments favor the Redis package.
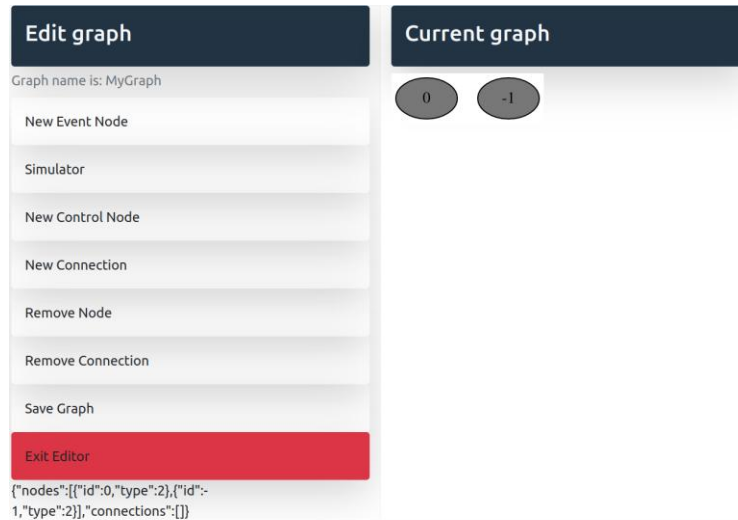
Redis is an open-source (BSD licensed) server-side application, in-memory data structure store, used as a database, cache, message broker, and streaming engine. To achieve optimal performance, Redis operates with in-memory datasets in key-value pairs. Depending on the use case, Redis can preserve data by periodically writing the dataset to disk or appending each command to a disk-based log. Persistence can be disabled for feature-rich, networked, in-memory caching needs. Redis supports asynchronous replication with fast, non-blocking synchronization and automatic reconnection, along with partial resynchronization in case of network partitions.

Redis is written in ANSI C and operates without external dependencies on most POSIX systems, including Linux, *BSD, and Mac OS X. Linux and OS X are the primary operating systems where Redis is most developed, tested, and recommended for installation.

5. The graph simulator application

During the research, the prototype version of the application was implemented. The whole user-related functionalities are achieved in the browser, only the simulation process is performed at the backend. Both the backend and the frontend of the application are available in a Docker container. The communication between the two takes place via a docker network, in the form of HTTP requests (JSON) and responses. Docker allows us to easily add new elements to the system, be it a database, a cache database, or other microservices. Another advantage of using Docker is that on a machine with Docker installed, the system can be started by issuing a docker-compose up command line.

To test the application, we created test graph sets of varying complexity, in which both OR and AND type process descriptions, as well as recursion and nested graphs, appear.
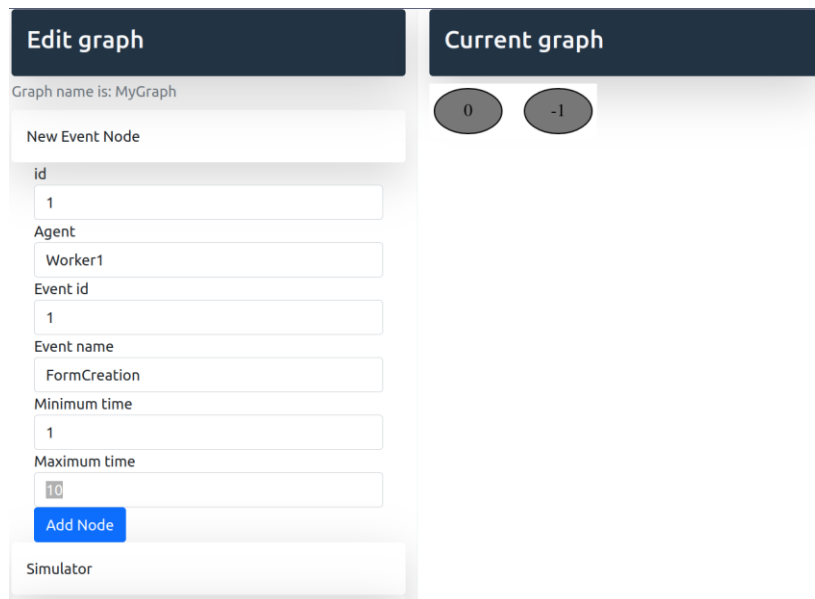
Figure 7. Main functions of the application

New Event Node

Adding a new event node is performed in a simple form. All fields of the form must be filled in according to the labels. Identifiers must be non-negative integers. The uniqueness of the node identifier is checked, and an error message is received if the identifier is not unique. Agent and event names are subject to the same restrictions as graph names, so they can contain English ABC letters and numbers.

The duration of the event is determined randomly during the simulation, and we can set lower and upper limits for this random duration by setting the Minimum time and Maximum time fields. These fields can only contain positive or zero values.
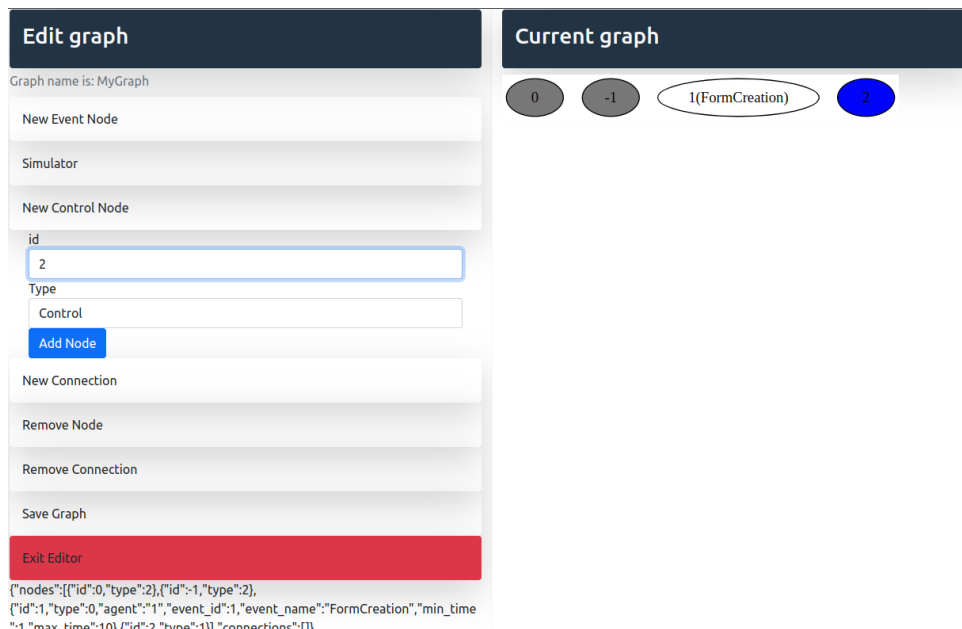
The screenshot below shows a completed form.

Figure 8. Adding a new event node

New Control Node

Adding a new control node is available under the New Control Node menu item. The control nodes will not appear in the simulation, instead they will modify the operation of the simulation. They must have a unique identifier (id) in the same way as the event nodes, but apart from that we only need to specify its type. The type can be of two types: XOR and Control.

The XOR node is actually the same as the event node, except that it does not appear in the simulation. So are the starting and ending points. The Control type is the actual control node that enables parallel events running in the simulation. Each event node to which an edge leads from a Control node can be executed in parallel and run on parallel branches until the edges meet at another Control node.

The screenshot below shows the completed control node form and the node added with the parameters in the figure. The Control nodes are depicted with a blue background, while the XOR control nodes are shown in gray, as can be seen at the start and end points.
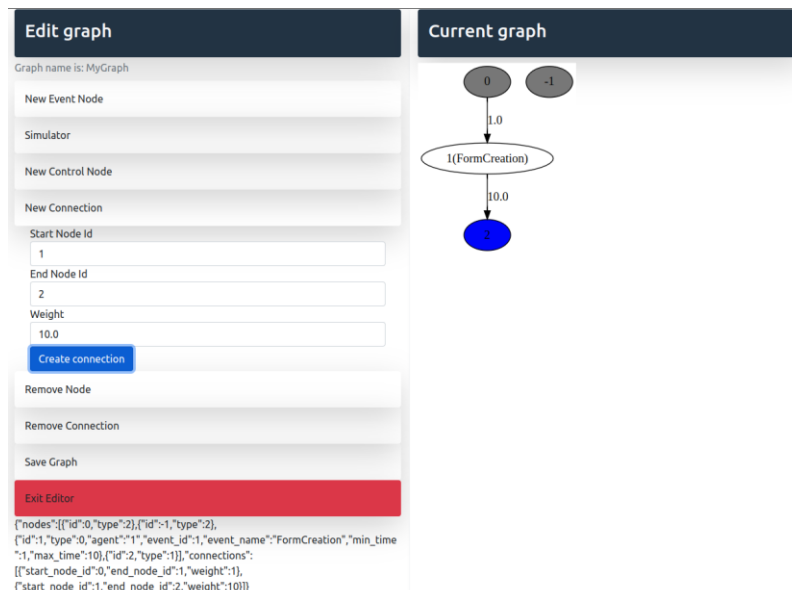
Figure 9. Creating a new control node

New Connection

To add a new edge we can specify from which node to which node we want to send the edge. In addition, the weight of the edge can also be entered here. Specifying the weight will be important in the simulation, because only one of the edges starting from a node (if the node is not a Control node) will be valid. So, from an event or XOR node in a simulation, we proceed in only one direction, and if several edges lead from the edge, the simulator will randomly choose from among the edges. The weights indicate the probability that the given edge will prevail compared to the others. If one edge has a weight of 10 and another has a weight of 1, then the edge with 10 is 10 times more likely The application currently performs few validations only on edges, and practically only checks the syntax of the entered data during construction: the weight must be a positive floating-point number. Here, the user must be careful to enter a logically correct graph.
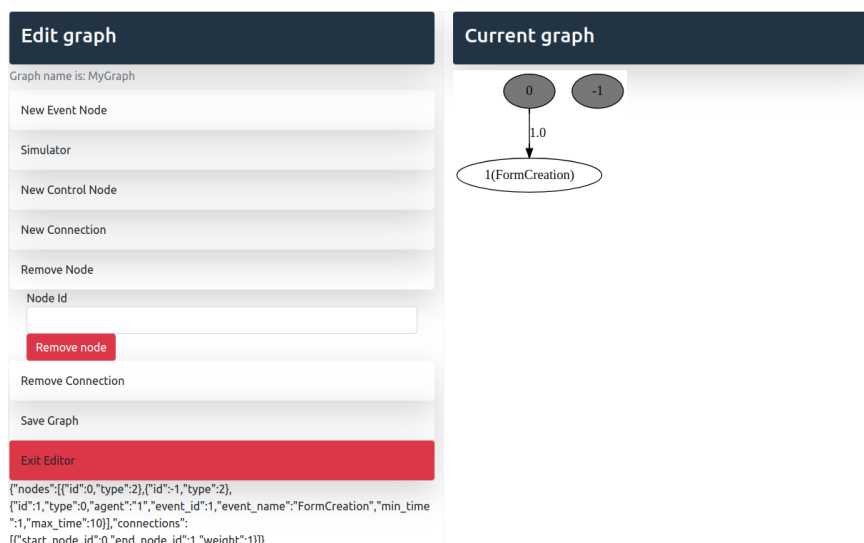
The screenshot below shows the open shape and some added edges on the graph.

Figure 10. Creating new connection

Remove Node

Deleting a node may be necessary if we mess up the editing of the graph and want to delete something back, or if we want to edit an existing graph in some way. In the delete node form, there is only a drop-down list, in which the IDs of all existing nodes (except the start and end nodes) will be listed. To perform the deletion, select the ID of the node to be deleted from the list, then press the Remove Node button. Note that when we delete a node, all of its edges are also deleted (that is, whether it starts from it or ends in it). For example, the screenshot below is of the graph built so far, after deleting the Control node; during deletion, the edge pointing to it was also automatically deleted.

Figure 11. Remove node

Remove Connection

To support connection remove function, the application has a form which contains two drop-down lists. Both drop-down lists contain all existing nodes. The starting
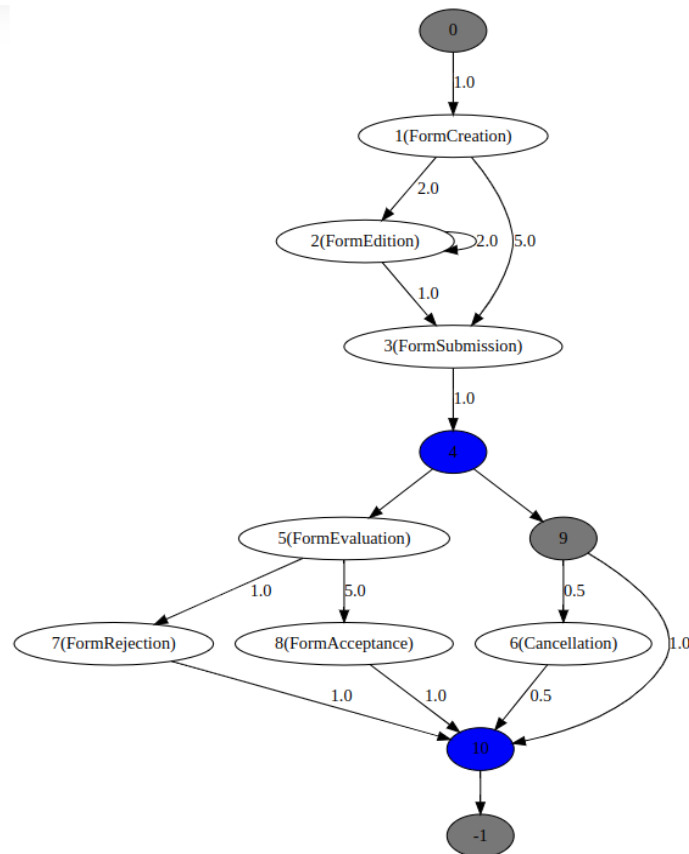
point of the edge to be deleted must be entered in the first, and the end point in the second.

Save Graph

Saving the graph is currently implemented as downloading the graph's description in the browser. The graph descriptor is a JSON String that contains the data entered in the forms. The name of the downloaded file will be the name of the graph with the .json extension. The downloaded graph does not have to be correct. One of the purposes of the download is to save the graph on the client machine as a file and open it again later for editing.

Simulator

One of the most important functions of the application is summation. It is possible to simulate the graph in the Simulator menu item. It is important to attempt to simulate the graph only when the graph is already valid.

Figure 12. Sample graph with simulation parameters

During the simulation, the description of the graph was transferred to the backend page, where it is sent to the Python processing engine. Based on the received graph description, the Python engine builds its own object structure, on which it performs a simulation according to the parameters (e.g. number of runs). The result of the process is a text file containing the generated events.

6. Conclusion

Process mining is a crucial area today, where analyzing complex event data can be used for many purposes. The conclusions drawn from the data help the company to operate more efficiently. The research focused on a very interesting area: the creation and simulation of event description graphs, i.e., the generation of sample data. During our work, we designed software that operates in a web environment, enabling the creation of the necessary graph using any browser. The software is capable of generating a dataset based on the graph. In its current form, the completed software can perform the basic tasks and is functional. However, there are numerous potential development opportunities in various areas, the implementation of which could create an even more efficient testing environment.

References

[1] Gejke, C.: A new season in the risk landscape: Connecting the advancement in technology with changes in customer behaviour to enhance the way risk is measured and managed. Journal of Risk Management in Financial Institutions. 11(2), 148–155, 2018

[2] Institute for Robotic Process Automation: Introduction to Robotic Process Automation. https://irpaai.com/wp-content/uploads/2015/05/Robotic-Process-Automation-June2015.pdf, 2015

[3] Mendling, J., Decker, G., Hull, R., Reijers, H. A., Weber, I.: How do Machine Learning, Robotic Process Automation, and Blockchains Affect the Human Factor in Business Process Management? Communications of the Association for Information Systems. 43, 2018

[4] Ratia, M., Myllärniemi, J., Helander, N.: Robotic Process Automation - Creating Value by Digitalizing Work in the Private Healthcare In: ACM International Conference Proceeding Series: International Academic Mindtrek Conference, 2018, https://doi.org/10.1145/3275116.3275129

[5] Greyer-Klingeberg, J., Nakladal, J., Baldauf, F.: Process Mining and Robotic Process Automation: A Perfect Match. In: 16th International Conference on Business Process Management, Sydney, Australia, 2018

[6] Lacity, M., Willcocks, L.P.: Robotic process automation at telefónica O2. MIS Q. Executive. 15, 21–35, 2016

[7] Leno, V., Dumas, M., Maggi, F.M., La Rosa, M.: Multi-Perspective Process Model Discovery for Robotic Process Automation, CEUR Workshop Proceedings, 2114, pp. 37-45, 2018

[8] Rajesh, K.V.N., Ramesh, K.V.N.: Robotic Process Automation: A Death knell to dead-end jobs?, CSI Communications-Knowledge Digest for IT Community. 42(3), 10-14, 2018

[9] Anagnoste, S.: Robotic Automation Process – The operating system for the digital enterprise. In: Proceedings of the International Conference on Business Excellence, vol. 12, no. 1, pp. 54-69. De Gruyter Poland, 2018, https://doi.org/10.2478/picbe-2018-0007

[10]   Boell, S.K., Cecez-Kecmanovic, D.: On being `systematic' in literature reviews in IS. J. Inf. Technol. 30(2), 161-173, 2015, https://doi.org/10.1057/jit.2014.26

[11]   Tsaih, R., Hsu, C. C.: Artificial intelligence in smart tourism: A conceptual framework. In: Proceedings of The 18th International Conference on Electronic Business, ICEB, Guilin, China, December 2-6, pp. 124-133, 2018

[12]   Forrester: The Forrester Wave™: Robotic Process Automation, Q1 2017 - The 12 Providers That Matter Most and How They Stack Up, Forrester Research, Inc., https://www.forrester.com/report/The+Forrester+Wave+Robotic+Process+Automation+Q1+2017/-/ERES131182, 2017

[13]   Schmitz, M., Dietze, C, Czarnecki, C.: Enabling Digital Transformation Through Robotic Process Automation at Deutsche Telekom, In: N. Urbach, M. Röglinger (eds.) Digitalization Cases - How Organizations Rethink Their Business for the Digital Age, 13-53. Springer, 2019, https://doi.org/10.1007/978-3-319-95273-4_2

[14]   Deloitte: The robots are ready. Are you? Untapped advantage in your digital workforce, https://www2.deloitte.com/content/dam/Deloitte/tr/Documents/technology/deloitte-robotsare-ready.pdf, 2017

[15]   Everest Group: Robotic Process Automation Annual Report 2018-Creating Business Value in a Digital-First Word, https://www2.everestgrp.com/reports/EGR-2018-38-R-2691, 2018

[16]   Suri, V. K., Elia, M., van Hillegersberg, J.: Software bots-The next frontier for shared services and functional excellence. In: International Workshop on Global Sourcing of Information Technology and Business Processes, pp. 81-94. Springer, Cham, 2017, https://doi.org/10.1007/978-3-319-70305-3_5

[17]   Forrester: Building a Center of Expertise to Support Robotic Automation: Preparing for the Life Cycle of Business Change, http://neoops.com/wp-content/uploads/2014/03/ForresterRA-COE.pdf, 2014

[18]   Everest Group: Defining Enterprise RPA, https://www.uipath.com/company/rpa-analyst-reports/defining-enterprise-rpa-everest-research-report, 2018

[19]   Lacity, M., Willcocks, L.P.: Robotic Process Automation: The Next Transformation Lever for Shared Services. In: The Outsourcing Unit Working Research Paper Series, Paper 16/01, http://www.umsl.edu/~lacitym/OUWP1601.pdf, 2016

[20]   Mileff P.: Role of user activity monitoring in RPA processes, Production Systems and Information Engineering 11(1): 27-42, 2023, https://doi.org/10.32968/psaie.2023.1.3

[21]   Erika Baksáné Varga, Attila Baksa: Discovering Process Models Containing XOR Branches. Production Systems and Information Engineering - ERPA Project, Vol. 12 No. 2, 2024.

[22]   Erika Baksáné Varga, Attila Baksa: Evaluating Process Discovery From Loop Structures. Production Systems and Information Engineering - ERPA Project, Vol. 12 No. 2, 2024.

[23]   Angular Web development Framework: https://angular.dev/, 2024

[24]   D3JS Javascript library: https://d3js.org/, 2024

[25]   DOT language specification: https://graphviz.org/doc/info/lang.html, 2024