

# Conceptual Model Detection in Business Web Applications Within the RPA Context

Dora Moraru, Adrian Sterca<sup>a</sup>, Virginia Niculescu<sup>b</sup>, Maria-Camelia Chisăliță-Crețu<sup>c</sup>  
and Cristina-Claudia Osman<sup>d</sup>

*Babeş-Bolyai University, Cluj-Napoca, Romania*

**Keywords:** Web Automation, Robotic Process Automation, Data Model Extraction, Entity Detection in Web Applications.

**Abstract:** Robotic Process Automation (RPA) platforms target the automation of repetitive tasks belonging to business processes, performed by human users. Web automation is a part of RPA that deals with the automation of web applications. In our previous work, we introduced a web automation tool in the form of a browser plugin that is able to discover simple processes on a target business web application and is able to later execute these simple processes automatically without human intervention. Our web automation tool relies on a description of the data model used by the target web application. This data model must be manually specified by the human user in advance. In our current work, we propose a new method for discovering this data model automatically from the target web application itself. We also performed some experiments in the paper that show that our method is viable.

## 1 INTRODUCTION

Companies operating in different business areas often share the same economic and human objectives while performing specific business processes. Economic objectives involve reducing costs, increasing productivity, improving efficiency, and service quality. Human objectives indicate practices that allow employees to feel valued and supported in the working area, which frequently means reducing the amount of repetitive and manual work.

Usually, business processes require interaction between various business applications (e.g., Customer Relationship Management (CRM) systems, Enterprise Resource Planning (ERP) software).

Robotic Process Automation (RPA) is the application of specific methodologies and technologies that aim to automate business process-related repetitive tasks achieved by human users (Hofmann et al., 2020).


Existing RPA tools provide capabilities for developing software robots (or bots) that mimic the actions performed by humans to execute the steps of a particular business process.


Automation includes a wide range of actions (Hartley and Sawaya, 2019), such as data entering, simple calculations, data reading and extraction from ERP systems, PDF files, or images, form fill out, e-mail reply, attachments opening or saving, particular application features execution, operations over files and folder, data scraping from web pages, etc. More, multiple technologies such as optical character recognition (OCR), natural language processing (NLP), and various artificial intelligence (AI) methods are used for data extraction (Hegde et al., 2018).


Software robots automate rule-based processes using high-volume and well-defined structured data processed in a deterministic context (Willcocks and Lacity, 2016).


Current RPA platforms (e.g., UiPath<sup>1</sup>, Automation Anywhere<sup>2</sup>, Blue Prism<sup>3</sup>, Microsoft Power Automate<sup>4</sup>, etc.) help increase work efficiency and accuracy by automatizing business processes and executing them more robustly by avoiding possible human errors.

RPA refers to those tools that operate on the user interface (UI) aiming to perform automation tasks using an "outside-in" approach (Van der Aalst et al.,

<sup>a</sup>  <https://orcid.org/0000-0002-5911-0269>

<sup>b</sup>  <https://orcid.org/0000-0002-9981-0139>

<sup>c</sup>  <https://orcid.org/0000-0002-1414-0202>

<sup>d</sup>  <https://orcid.org/0000-0002-9706-2915>

<sup>1</sup><https://www.uipath.com/>

<sup>2</sup><https://www.automationanywhere.com/>

<sup>3</sup><https://www.blueprism.com/>

<sup>4</sup><https://powerautomate.microsoft.com/en-us/>

2018). RPA is not an invasive technology, based on existing systems, without being constrained to develop or replace costly platforms (Willcocks and Lacity, 2016).

RPA projects usually follow an automation process consisting of task identification, process redefinition from AS-IS to TO-BE, actual bot development, testing, deployment, and monitoring (Huang and Vasarhelyi, 2019), (Gex and Minor, 2019).

RPA developers identify UI components of a software application like buttons, text input controls, drop-down lists, and tables, and then customize activities by writing code snippets in a programming language to act on these UI controls (e.g. clicking the selected button, writing data in the text input, select all data from a table or a drop-down list, etc.). The code referencing the selected UI controls forms the automated business process which can be executed many times later with different input parameters.

This paper emphasizes a complementary approach, where we automatically navigate over the web application and aim to identify a conceptual model employed throughout the executed business processes.

The original contributions presented in the paper can be summarized as follows. In previous work (Sterca et al., 2023) we developed a web automation tool in the form of a browser plugin that is able to discover simple processes in a target business web application and it can later execute these simple processes automatically without human intervention; an example of such a simple process which we call a conceptual operation is the operation of adding an account in a CRM business application. In our previous work, we relied on a description of the data model used by the target web application (i.e. the concepts and entities used by the web application together with their properties). This data model must be created manually by the human user either from the official documentation of the target web application or from the UI of the target web application itself. This manual labour is tedious and takes time. In our current work, we propose a new method for the automatic discovery of this data model of the target business web application, keeping the human user intervention to the lowest degree possible.

The rest of the paper is structured as follows: Section 2 provides related work about business processes in the context of RPA. Section 3 offers details about our web automation tool developed in previous work. Our algorithm for the automatic discovery of the data model of the target web application is described in detail in Section 4, while Section 5 describes some evaluation tests that we performed in order to show

that our method of automatic detection of the data model of a business web application is viable. The paper concludes with the final remarks and future work ideas.

## 2 RELATED WORK

The software development cycle (SDC) starts with a high-level design that gets translated into source code. Reverse engineering flips this process around, taking the source code or the software product and identifying the original design intent. Reverse engineering allows the extraction of ER models (Hainaut, 1991) or UML models (Iftekhhar et al., 2019), (Abb and Rehse, 2022) from schema or schema-less databases.

RPA is a cutting-edge technology that allows companies to automate repetitive office tasks. Utilizing software robots programmed to execute specific instructions, RPA imitates human actions by interacting with various web and desktop applications (Van der Aalst et al., 2018). On the other hand, Process Mining (PM) represents a set of techniques that extracts models based on human behaviour recorded as event logs (Van Der Aalst, 2016). The intersection of these fields is described as Robotic Process Mining (RPM) (Leno et al., 2021). RPM instruments aim to create RPA scripts from UI logs. Nevertheless, there are some differences between event logs (input for PM) and UI logs (input for RPM) (Leno et al., 2021). Firstly, the level of abstraction is different: UI logs are generated at GUI level, while event logs are generated at the business process task level. Therefore, a business process task may consist of several UI actions. Event logs ask for a case ID, while a unique identifier for each case is not required at UI log level. The data produced/consumed within an event is not mandatory to be stored at the event log level, while UI logs need this data. The fourth difference mentioned by (Leno et al., 2021) refers to the missing events that in some situations are ignored if they are not recorded by the information systems used by the involved participants.

A method for analyzing UI logs with the aim of identifying sequences of actions is depicted by (Bosco et al., 2019). The focus is on discovering routines that can be automated to improve the user experience or streamline processes. The limitations of their research refer to routines involving loops (when the exit condition is not identifiable) and noise (the presence of an event that does not belong to the current routine). There are also approaches that discover candidate routines from unsegmented UI logs in the presence of noise (Leno et al., 2020). Application logs are also analysed with the aim of extracting the most

frequent user tasks. Usually frequent pattern mining techniques and sequential pattern mining techniques (Mannila et al., 1997), (Agrawal and Srikant, 1995) are used, but (Dev and Liu, 2017) propose a rank pattern based on *membership based cohesion*.

A reference data model for process related UI logs is proposed by (Abb and Rehse, 2022). This model is described as an extension of the standard used for event logs, XES (Günther and Verbeek, 2013). Our plugin browser maps the UI operations to conceptual operations in a database, the main idea being focusing on the connection of UI actions to database operations.

Web scraping techniques are used to automatically extract information from websites. These techniques have diverse applications, including gathering contact information, monitoring and comparing price changes, collecting product reviews, compiling real estate listings, tracking weather data, detecting website changes, and integrating web data (Zhao, 2017). Three phases are needed for web scraping projects: site analysis, data analysis and design, and production (ten Bosch et al., 2018). Our approach is different than web scraping as the plugin browser identifies the conceptual operations and it can execute these operations. Furthermore, the data model is discovered based on the identification of the conceptual operations.

In a systematic mapping study, Sepulveda et. all (Sepulveda et al., 2017) summarizes the methods and techniques to obtain the data model from business process models. In the RPA context, our method extracts the data model by examining the Graphical User Interface (GUI) components and the Document Object Model (DOM) structure. None of the approaches presented earlier focus on extracting conceptual data models based on the GUI/DOM structure.

### 3 WAPLugin FOR WEB AUTOMATION

WAPLugin is a web automation tool implemented as a browser plugin that automatically translates human user operations on the UI of a target web application onto conceptual operations in the underlying database. We consider conceptual operations as being CRUD (i.e. Create, Retrieve, Update, Delete) operations on concepts; the term *concept* refers to a data type stored in a database table, while an *entity* refers to a row/record of a database table. A *concept* always describes a set of *entities*; e.g. an ‘Account concept’ is stored in an ‘Account’ table in the database, while each entry/record from this table represents an ‘Ac-

count entity’.

This web automation tool is developed as a browser plugin, and so it bears the advantages of this kind of applications. The usage of this tool involves two stages:

- one stage through which the tool discovers the specific conceptual operations
- and another stage in which the tool is able the automatically executes these operations.

In the first stage, the human user guides the tool by navigating in the target business web application (i.e. clicking through the various buttons, links and other clickable elements of UI of the target web application), so that the plugin discovers what we call primary blocks for process automation - which is another name for a conceptual operation. The user has to follow on the UI all the steps that are needed to each conceptual operation; all these operations assisted by the user form the stage that we may call *the learning stage*. Based on this guidance, the plugin discovers and extract these primary blocks, and then on a further stage, it can automatically execute such primary blocks on the target web application or it can form complex processes with the discovered primary blocks and it can execute those too. The human user specifies to the browser plugin which primary block (i.e. conceptual operation) to execute and he/she can also provide parameters for the respective conceptual operation; for example, if the user wants to execute an ‘Add New Account’ automatically, he/she must specify which conceptual operation should be executed to the browser plugin (in this case its an ‘INSERT Account’ conceptual operation) and then the user must provide the parameters for this conceptual operation (e.g. the name of this new account, its address, description and other properties described in the data model of the business web application).

For further details, please see (Sterca et al., 2023).

The WAPLugin automation tools requires up front a description of the data model of the target business web application. This data model is just a list of all the concepts used by the target web application. Each concept is defined by a list of properties and is equivalent to a database table in the underlying database used by the target web application. This data model is an input parameter for all functionalities offered by our browser plugin (together with the root URL of the target web application and the access credentials to the application). In our previous work, we specify this data model of the target web application manually in the form of a long JSON string. The purpose of our current work is to determine this data model (or an approximation of it that will be further refined by a human expert) automatically by browsing through

the web pages generated by the target business application. This automatic discovery of the data model of the target business web application would reduce the amount of labour required from the human user who otherwise needs to consult the technical documentation of the application or he/she needs to browse through the webpages of the UI of the application in order to manually describe the data model of the application in JSON format.

#### 4 DETECTING THE DATA MODEL OF THE TARGET WEB APPLICATION

The basic idea of our method for detecting the underlying data model of a business web application is that as long as the business web application exposes the *Update* functionality for its entities (and most, if not all, business applications offer this kind of functionality to its human users; otherwise, all the entities would be read-only in the application), a browser plugin can detect these entities by discovering html input fields (like `<input type=text >`, `<select >`, `<textarea >`) in the web pages generated by the application and can associate these input fields to textual labels from the UI of the application. These textual labels will form the properties/fields of entities in the data model. As long as the web application offers update functionality, it has to use html input fields in order to get values from the human user (the web application can, of course, implement its own custom input fields, but this involves complex javascript code which is inefficient as long as html tags described by the HTML standard language are available and were especially created for this purpose - to get the input from the human user).

The algorithm used for automatically browsing through the web pages generated by the business web application, algorithm which is implemented in our browser plugin, works by running a depth-first traversal on the entire web application. The various web pages of the application are traversed through clickable elements. This algorithm is described in code listing 1.

In the first lines of the algorithm, lines 1-3, we perform initializations of various structures used in the algorithm: the stack and the current *clickableElements* present on the web page. The algorithm assumes that the starting HTML document of the web application is already loaded in the browser (this means the access credentials were already input by the human user before running the algorithm). The

Algorithm 1: The algorithm that navigates automatically through a business web application.

**The AutomaticBrowsing algorithm is:**

```

1: Stack = [] ; newPath = [];
2: clickableElements = GetClickableElements()
3: Stack.push(newPath,clickableElements)
4: while (Stack is not empty) do
5:   clickableElementsSet = Stack.top()[1]
6:   if clickableElementsSet == NULL then
7:     continue
8:   end if
9:   ClickElem =
10:  GetFirstUnvisitedClickableElement(clickableElementsSet)
11:  ClickElem.state = visited
12:  if ClickElem is not in the current DOM then
13:    Navigate(Stack.top()[0])
14:  end if
15:  remaining = GetUnseenClickableElements()
16:  Stack.push(newPath,remaining)
17:  entity = DetectEntity()
18:  if entity! = NULL then
19:    DataModel.add(entity)
20:    Stack.pop()
21:    GoBackOneStep()
22:  end if
23: end while

```

algorithm retrieves the *clickableElements* from this HTML document in line 2. An element of the stack is formed by the *clickableElementsSet* found on the web page and a list (*newPath*) formed by the XPATHs of the clicks that led to the current webpage (in line 3, *newPath* is an empty list).

Following, lines 4-23 define the main cycle of the iterative depth-first traversal algorithm. The *Stack* stores all the *clickableElements* that need to be traversed/visited. An element is removed from the stack when all its *clickable elements* are visited by our tool or when an entity for the data model is found. On lines 5-14 we take the first element from the top of the stack, get its first unvisited (i.e. unclicked) *clickable element* and check if it is present in the current DOM; if not, the plugin navigates to the path of *ClickElem*, using the first list from the element retrieved from the stack (*Stack.top()[1]*). Each element of the stack is a pair element containing at index 0 the *path* - sequence of clickable elements that got us to this webpage and at index 1 the *webpage* - defined by a set of clickable elements. Reaching the web page that contains our current *clickable element*, the browser plugin clicks it, the DOM is expected to change, and in the following steps (lines 15-16) we get the new *clickable elements*, that have not been seen until now by the plugin, and store them into the stack in order to be visited later.

Next, on line 17, we try to detect if an entity is present on the current web page (*DetectEntity()*). Lines 18-22 are for the case when an entity from the

data model is present in the loaded HTML document. In this scenario, we store this entity in the data model, remove the last element from the stack because we are no longer interested in the current web page (since we have found the entity), and go back a step (there's a higher probability that the next unvisited clickable element is situated on the previous page than others).

The algorithm for detecting an entity for the data model is based on the premise that a label associated with an input is situated on its North-West side. Therefore, we firstly detect the inputs (<input>, <select>, <textarea>) present on the HTML web page and then try to find for each input the closest textual label that is positioned in the North-West quarter of a circle with the centre in the middle of the input. If for more than one input we have found a pair in the list of labels, it means that the plugin has found an entity for the data model of the page, and the associated textual labels are the properties/fields of the entity of the data model.

## 5 VALIDATION

We conducted an experiment in which we tried to determine the data model for a business web application. The application we used to test our browser plugin is Microsoft Dynamics 2016 CRM. This business application has more than 150 distinct types of web pages generated (i.e. more than 150 functionalities). We let the plugin discover entities for 30 minutes, time in which it found 21 different entities out of 21 possibilities (apparent 100% precision; also the plugin did not detect any false entity - an entity that does not exist in the application). This precision of 100% is apparent because although the plugin detected all the available entities in the time frame of the experiment (30 minutes), occasionally some of the properties/labels of the detected entities were not properly detected.

We realize that it would have been better to let the plugin run for a significantly longer period of time, so that it discovers all entities in the data model of the Microsoft Dynamics 2016 CRM application, but the web pages generated by this application are similar and we consider that running the data model detection algorithm of our plugin for about 30 minutes is statistically relevant for the whole Microsoft Dynamics 2016 CRM application. After running the browser plugin for 30 minutes, we report the entities it has found. Some of these entities were found multiple times because of their numerous occurrences in the web pages of the application. Being a complex application, it has a variety of buttons and features, such as generating reports or downloading files. Therefore,

the plugin had to traverse these steps with the aim of discovering an entity.

Running the data model discovery algorithm for a longer time period would, of course, allow the plugin to discover more entities from the data model of the target business web application.

In screenshot Fig. 1, we can see how the plugin detected the clickable elements and bordered them in red. We compute the XPATH of these HTML elements and push them into a stack, in order to be later clicked one by one in a depth-first search manner.

In Fig. 2 and Fig. 3, the plugin firstly detected the input fields present on this web page and then associated the closest textual label to each input. Therefore, the labels which are associated with an input, represent the properties of an entity, or more exactly, the column's names of an entity in a database. In rare cases, where the wrong labels are closer to an input, the associations are incorrect.

The list of entities that make up the model is saved by the browser plugin in JSON format due to the ubiquitous support for this format in web applications. Each entity is described by its name and a list of attributes/fields (which are equivalent to column names in a database table). For example, a small snippet of the entities from the Microsoft Dynamics 2016 CRM application identified by our browser plugin tool is the following :

```
[
  "Invoice" :
  ["Invoice ID", "Name", "Currency",
   "Price list", "Prices locked",
   "Opportunity", "Order", "Customer",
   "Description", "Date delivered",
   "Due date", "Shipping method",
   "Payment terms", "Bill to address",
   "Ship to address"],
  "Order" :
  ["Order ID", "Name", "Currency",
   "Price list", "Prices locked",
   "Opportunity", "Quote", "Potential
   Customer", "Description", "Requested
   delivery", "Date fulfilled",
   "Shipping method", "Payment terms",
   "Freight Terms", "Bill to address",
   "Ship to address"]
]
```

We should also note that the detected model of the application is just an approximation of the data model of the database used by the target web application. The actual data model of the database used by the web application is hidden from the human user (i.e. the human user does not have direct access to the underlying database of the web application). The detected data model is the one imprinted in the UI of the web application (i.e. it is comprised from the labels

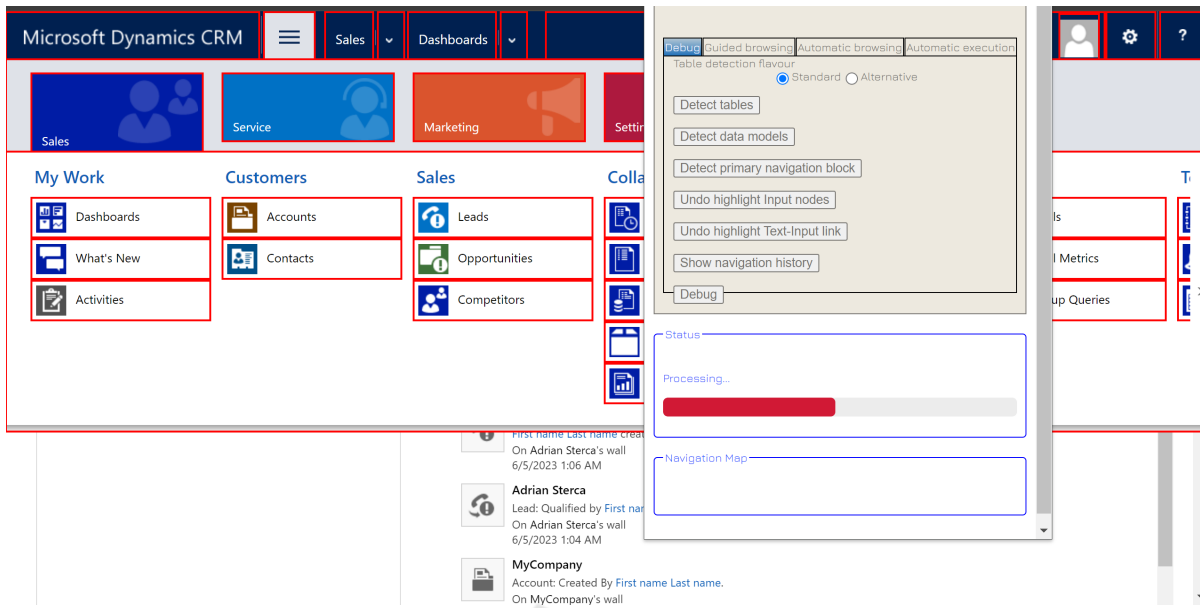


Figure 1: Detection and highlight of clickable elements from a web page.

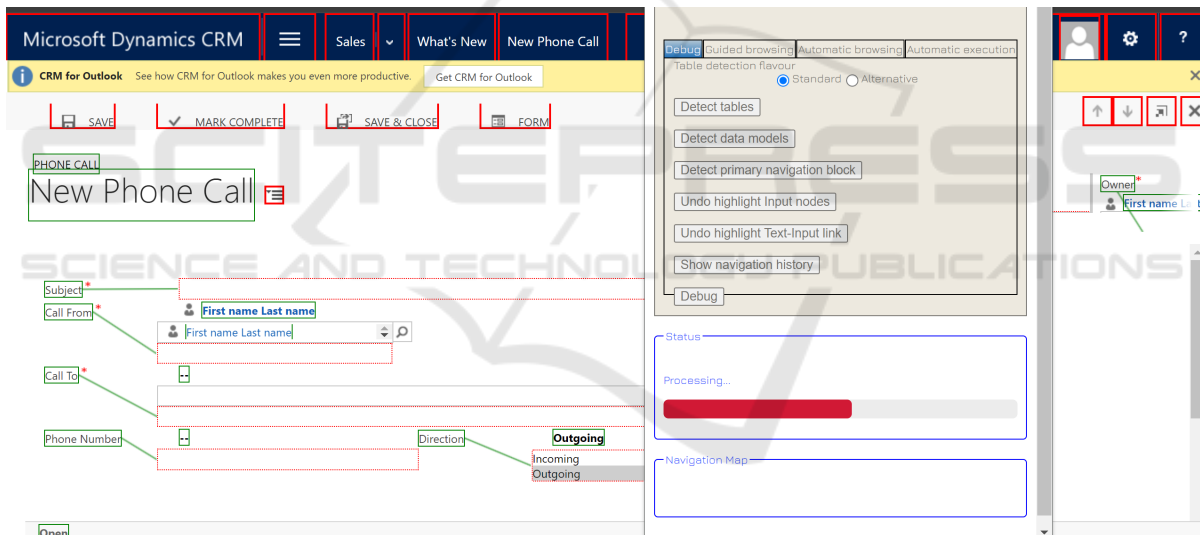


Figure 2: Detecting an entity in Ms Dynamics 2016 CRM - example 1.

used in the UI of the web application). But this approximate data model is the one we need if we want to automate the usage of the target web application as explained in Section 3.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a method for automatic discovery of the data model used by a target business web application. Our method is implemented

in a browser plugin and is based on automatic navigation through the web pages generated by the target business web application. It identifies specific DOM elements (i.e. html input tags) and associates these to properties of concepts in the data model to be discovered. We note that this data model is just an approximation of the actual data model of the underlying database used by the target web application (this database is hidden from the human user). But this approximate data model is all we need in order to implement a web automation tool in the form of a browser plugin (i.e. a tool that is able to discover business processes - with or without the help of the human user -

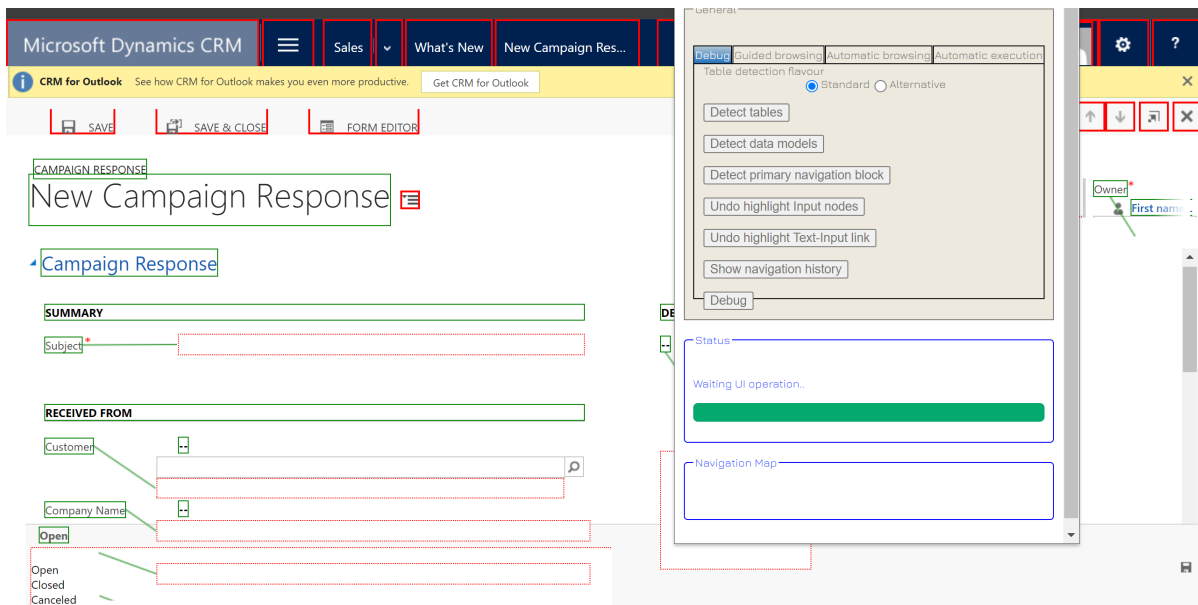


Figure 3: Detecting an entity Ms Dynamics 2016 CRM - example 2.

in a target business web application and is later able to execute these processes automatically.

We performed some evaluation tests on a business web application, namely Microsoft Dynamics 2016 CRM in order to show that our method is viable. Our method indeed discovered entities from the underlying data model used by the target web application.

Of course, our work has limitations. In future work, we intend to perform evaluation tests on other business web applications, not just on Microsoft Dynamics 2016 CRM.

Also, our current work is not able to detect references/links between concepts in the data model (i.e. the equivalent of foreign keys in database tables). And this would be a good starting point for future work.

## REFERENCES

- Abb, L. and Rehse, J.-R. (2022). A reference data model for process-related user interaction logs. In *International Conference on Business Process Management*, pages 57–74. Springer.
- Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*, pages 3–14. IEEE.
- Bosco, A., Augusto, A., Dumas, M., La Rosa, M., and Fortino, G. (2019). Discovering automatable routines from user interaction logs. In *Business Process Management Forum: BPM Forum 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pages 144–162. Springer.
- Dev, H. and Liu, Z. (2017). Identifying frequent user tasks from application logs. In *Proceedings of the 22nd international conference on intelligent user interfaces*, pages 263–273.
- Gex, C. and Minor, M. (2019). Make your robotic process automation (rpa) implementation successful. *Armed Forces Comptroller*, 64(1):18–22.
- Günther, C. W. and Verbeek, E. (2013). Xes standard definition. 2014. *BPMcenter.org*.
- Hainaut, J.-L. (1991). Database reverse engineering: Models, techniques, and strategies. In *ER*, pages 729–741.
- Hartley, J. L. and Sawaya, W. J. (2019). Tortoise, not the hare: Digital transformation of supply chain business processes. *Business Horizons*, 62(6):707–715.
- Hegde, S., Gopalakrishnan, S., and Wade, M. (2018). Robotics in securities operations. *Journal of Securities Operations & Custody*, 10(1):29–37.
- Hofmann, P., Samp, C., and Urbach, N. (2020). Robotic process automation. *Electronic Markets*, 30(1):99–106.
- Huang, F. and Vasarhelyi, M. A. (2019). Applying robotic process automation (rpa) in auditing: A framework. *International Journal of Accounting Information Systems*, 35(C).
- Iftekhar, N., Warsi, M. R., Zafar, S., Khan, S., and Biswas, S. S. (2019). Reverse engineering of relational database schema to uml model. In *2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–6. IEEE.
- Leno, V., Augusto, A., Dumas, M., La Rosa, M., Maggi, F. M., and Polyvyanyy, A. (2020). Identifying candidate routines for robotic process automation from unsegmented ui logs. In *2020 2nd International Conference on Process Mining (ICPM)*, pages 153–160. IEEE.

- Leno, V., Polyvyanyy, A., Dumas, M., Rosa, M. L., and Maggi, F. M. (2021). Robotic process mining: Vision and challenges. *Business & Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK*, 63(3):301–314.
- Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data mining and knowledge discovery*, 1:259–289.
- Sepulveda, C., Cravero, A., and Cares, C. (2017). From business process to data model: A systematic mapping study. *IEEE Latin America Transactions*, 15(4):729–736.
- Sterca, A., Niculescu, V., Chisăliță-Crețu, M.-C., and Osman, C.-C. (2023). Primary building blocks for web automation. In Zhang, F., Wang, H., Barhamgi, M., Chen, L., and Zhou, R., editors, *Web Information Systems Engineering – WISE 2023*, pages 376–386, Singapore. Springer Nature Singapore.
- ten Bosch, O., Windmeijer, D., van Delden, A., and van den Heuvel, G. (2018). Web scraping meets survey design: combining forces. In *Big Data Meets Survey Science Conference, Barcelona, Spain*.
- Van Der Aalst, W. (2016). *Data science in action*. Springer.
- Van der Aalst, W. M. P., Bichler, M., and Heinzl, A. (2018). Robotic process automation. *Business and Information Systems Engineering*, 60:269–272.
- Willcocks, L. P. and Lacity, M. (2016). *Service automation robots and the future of work*. SB Publishing.
- Zhao, B. (2017). Web scraping. *Encyclopedia of big data*, 1.

