Virtual Employee Implementation Using Temporal Case-based Reasoning

Ivan Kurilenko[1][0000-0003-1520-640X] and Igor Nikonov[2][0000-0001-8413-5049]

[1] National Research University "Moscow Power Engeneering Institute", Moscow, Krasnokazarmennaya st., 14, 111250, Russia
ivan@appmat.ru
[2] National Research University "Moscow Power Engeneering Institute", Moscow, Krasnokazarmennaya st., 14, 111250, Russia
nikonovie@gmail.com

Abstract. This paper discusses the implementation of the virtual employee that implements the function of automatic processing of issues submitted to the software maintenance department. This task is currently relevant and a variety of virtual employees are being introduced in the different areas, since their use allows to automate typical operations, reduce response time and free up human resources for more complex tasks. For the implementation of this system, modified case-based reasoning method based on temporal constraint satisfaction problem (TCSP) was proposed. This method allows to take into account temporal dependencies and model the history of the events occurring, which allows to describe the dynamics of the system behavior that led to the system failures or incorrect behavior. For comparison of numerical parameters, classic Euclidean metric was used. For estimating the measure of proximity of text parameters of cases, modification of TF-IDF measure was developed. The proposed modification allows to improve the quality of the classification, because it allows to take into account information on the contribution of the words not only in a separate case but in the entire case base. The paper describes a prototype of developed virtual employee that is used in the process of maintaining the automatic payment terminal software. At the same time, it allows not only to work on the principle of request-response, but also to integrate with various systems, interact with people and line up in business processes.

Keywords: virtual employee, case based reasoning, temporal constraint satisfaction problem.

1 Introduction

This paper discusses the implementation of the virtual employee (VE) [1-3]. VE is like an automated robot who completes a full day work. VE can "work" on positions with simple or typical duties (for example, technical support engineers, controllers, etc.). In these areas, VE can be integrated into the real business processes and perform interaction with people, imitating a real employee.

According to literature, such VE is the example of actively investigated in nowadays robotic process automation (RPA) tool [4]. Work [5] indicates that using of the RPA has become a necessity in day to day business activities. The term RPA means automation of service tasks that were previously performed by humans. The simplest technics that VE can use is transferring data from multiple input text sources (like email, notes, spreadsheets) to Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM) systems [6]. Some works concern rule-based RPA methods [7, 8]. Rule-based approach allows automation of repeatable tasks by the use of software tools, which can mimic actions performed by human users on computers to complete various business processes [9]. Another works consider using more complicated artificial intelligence technics for RPA [10-12]. In particular, neural networks and natural language processing (NLP) can be harnessed to infuse RPA solutions with supervised, unsupervised and enhanced learning capabilities [10, 11]. The work [12] describes construction of self-improving helpdesk service system using case-based reasoning techniques. This paper discusses an example of building VE for technical support department.

Many software development companies in their daily activities provide technical support for the released software products. Users are provided with support that troubleshooting most problems that they experienced. If the error message is received, the support staff needs to know the parameters and conditions under which the problem situation occurs. An in-depth analysis of the error and clarification of its cause requires an understanding of the full picture of the events that occurred. Usually modern software recording these events automatically into the log files.

In this paper, we will consider as an example of a supported product automatic payment terminals. Installing and configuring payment terminals, as well as subsequent maintenance, is a complex process that can be conducted incorrectly, especially in the case of insufficient staff qualifications. At the same time, unskilled specialists are often inclined to give out any questions about the work of the terminal (including those that they should solve on their own), as errors of the installed software. As a result, the large part of the technical support requests, for obvious reasons, have a template decision. These template decisions can be submitted in the form of decision rules. For example: “If the user reported that the system was not responding to an attempt to pay with a contactless card, then it is necessary to check the log records from the contactless card reader driver. If it contains message “Port “COM#” does not exist”, then recommend to check this option”. Since the number of installed terminals amount to several thousands, the number of calls to the support service is quite large and grows as the new terminals installs. In order to accompany the terminals, low-skilled specialists who able to correct only typical errors according to decision patterns are usually hired. In case of atypical situations, an expert is involved to eliminate the error. If the problem is related to the equipment and its setting, it is eliminated and this typical decision is added to the knowledge base of the maintenance specialists. If the problem is related to a defect in the software, then a new version is released with a fix.

To account for user requests and the accumulation of typical decisions, currently specialized software systems are often used – Service (Help) Desk, hereinafter referred to as issue tracking systems (ITS) [13]. The core of the ITS is the database, which stores user requests. Requests are usually filled in via the ITS user interface and have a number

of attributes, such as number (identifier), short description of the user, date and time of access, version of software, severity (criticality), decision priority, description of steps to reproduce the problem, the expected result and the actual result, who is responsible for deciding the request, the current status (status), the decision. Also, ITS usually provide a software interface (API, application programming interface) that allows to manipulate requests for the development of automation tools. This API allows to get a list of issues assigned to a specific executor, create new issues, reassign issues, manage issue fields, status, make comments to the issue, etc.

A typical issue life cycle is determined by its status (see Fig. 1):

- "open" - issue is registered;
- "in progress" - the person responsible for the decision is appointed;
- "resolved" - decision was developed.

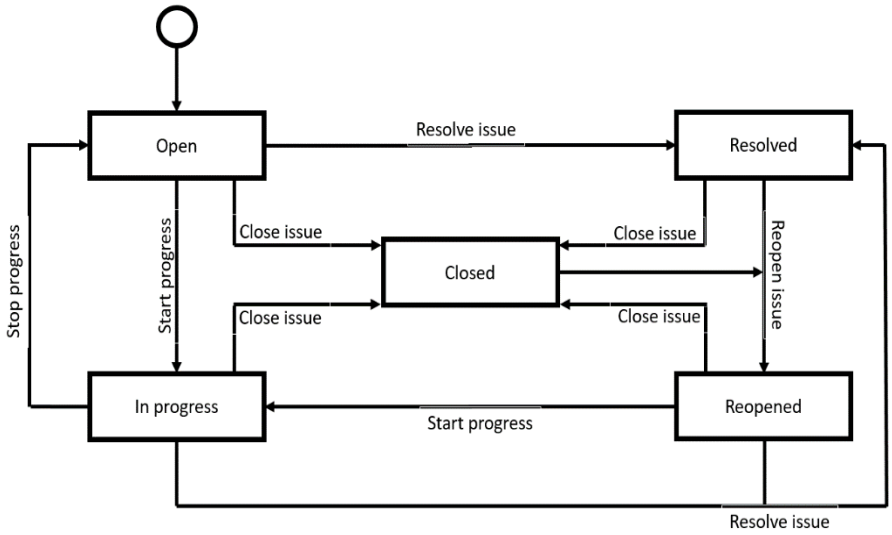

Fig. 1. Issue workflow.

Information about solved issues in the ITS can be used as a knowledge base (KB) in the technical support process.

VE considered in the work can be used as a superstructure above the ITS to reduce the amount of routine work. Through the program interface of interaction with the ITS, VE will process incoming issues to the system, conduct a search through the KB of typical decisions, change the issue status and attributes.

Methods for finding decisions based on case-based reasoning (CBR) [14] are well suited to implement such VE. Consider further the proposed design of VE and the scheme of its operation.

2 Virtual Employee Implementation

2.1 General Architecture

The architecture of VE includes the following components (see Fig. 2):

- interface with the ITS;
- case base;
- issue analyzer;
- product log analyzer;
- decision-making unit;
- learning unit.

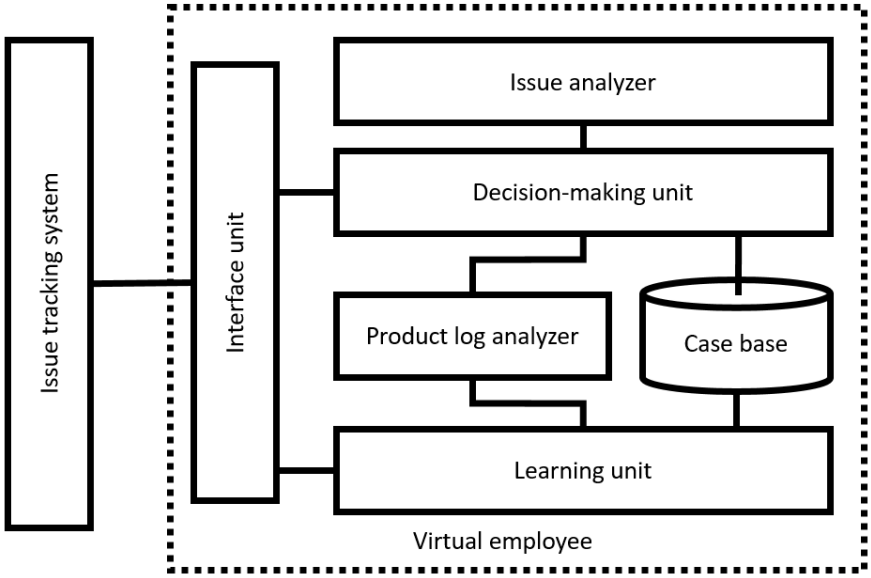

Fig. 2. A Virtual employee architecture.

The interface with the ITS provides receiving issues from the ITS and converting them into a form convenient for the implementation of the reasoning process. The issue is described as $Z = <A, H>$, where $A = (a_1, \ldots a_k)$, where a_i is the issue attribute, $H = \{h_i\}$ is the history of work on the issue (the list of events - actions of people and VE).

Searching for issues that are ready for processing carried out using the following rules:

1. If the issue did not have the attribute "responsible for the decision", then VE assigns this issue to itself and sends the issue to the analyzer.
2. Other issues are processed by VE only if it is indicated as responsible for the decision and the time specified in the attribute “time of issue change” is longer

than that specified in the attribute "decision time of the virtual employee". Such issues are send to the decision search block.

In the issue analyzer unit, checks for the correctness and completeness of the information that filling out the issue fields are making. This component is implemented based on production rules. An example of such a rule: if the "Software" attribute is set to "Access Controller Software" and the "Product log" attribute is not filled, then request the controller work log. The issue is considered suitable for transfer to the decision search unit if none of the rules has triggered.

In the decision search unit of VE the issue is matching with the existing ones in the case base. As a result, a typical decision is determined. If it have found, the corresponding recommendation is recording to the issue. Otherwise, VE reassigns the issue to the engineer.

Decision search unit uses the logs analyzer to compare software logs and will be discussed later.

The learning unit is used to form new cases in the case base from the solved issues that are available in the ITS.

2.2 Decision Search Unit

The decision search unit constructed using a modified CBR method. We will represent cases in the form of $CASE = (x_1, x_2, \ldots, x_n, T, R)$, where $x_1, \ldots, x_n$ are issue parameters; $x_1 \in X_1, x_2 \in X_2, \ldots, x_n \in X_n$, where n is the number of parameters for case describing, $X_1, \ldots, X_n$ – areas of allowable values of relevant parameters, T – function, that matches the comparison metric for the corresponding parameter, R – recommendation.

To evaluate the similarity of numeric parameters, the usual Euclidean metric is suitable. For the parameters presented in the form of text in natural language (NL), we propose the approach described below. To determine the similarity of two text parameters, we will represent each text T as an N-dimensional vector, where N is the number of all possible words in the target language. For each word w, which appears in the text of the parameter f of some case c, the value of the coefficient along the corresponding axis calculates by the following formula:

$$Weight_w(f, c) = \frac{p^w_{f,c}}{IDF^w_{f,c}} \times IDF^w_{f,rest},$$

where $p^w_{f,c}$ – is the probability that if w is present in the text of some parameter f of case c, that it belongs to case c described by this parameter and calculates by the following formula:

$$p^w_{f,c} = \frac{k^w_{f,c}}{k^w_f},$$

where $k^w_{f,c}$ – is the number of occurrences of the word w in the text of parameter f of case c, and k^w_f – is total occurrences of word w in the text of parameter f in all cases, i.e. $k^w_f = \sum_{i=1}^{n} k^w_{f,i}$, where n – is the total number of cases.

$IDF^{w}_{f,c}$ – *IDF* (inverse document frequency) value for word *w* within parameter *f* of case *c*, representing the inverse of the frequency with which the word occurs in the text [15]. $IDF^{w}_{f,rest}$ – *IDF* value for word *w* of parameter *f* for the remaining part of cases *rest*. In addition, if the word does not occur in any case in remaining part, then the value $IDF^{w}_{f,rest}$ sets to a maximum value and equals |*rest*| - the number of cases in *rest*.

Thus, the similarity metric between the text parameters of the analyzing issue *Z* and some case *C* from case base is determined as follows:

$$Sim(Z_f, C_f) = \sum_w k^w_{z,f} \times Weight_w(f, C),$$

where $k^w_{z,f}$ is the number of occurrences of the word *w* in the text of parameter *f* describing input issue *Z*. If the word is not found in the current case, the value of proximity is taken to be zero.

To speed up the calculations, the values of frequencies are pre-calculated and stored in a special database.

Important information for comparison is contained in the supported system logs. Their comparison is a difficult task. To solve it, the architecture of VE has a separate component – log analyzer.

For the considered software on the example of an automatic payment terminal system, the work process can be represented as a sequence of execution of typical operations. Each operation is described as a set of events. Abnormal situations that occur during software work can be identified by detecting non-standard or unknown events in the sequence of operations. Case can include both need for a specific event in the log — for example, errors of a particular type, or a more complex condition involving the control of previously occurred events (since the same error can have different causes). The first case does not present any difficulty for implementation, so we will not consider it in details. The second case allows for a more qualitative analysis of what is happening, since the situation will be considered "in dynamics", and the decision will be made taking into account the history of the process [16].

Formally, log is represented as a set of records *Rec* = (*t*, *s*, *k*, *m*), where $t \in D$ is the event observation time, s is the event anxiety, $s \in \{error, warning, normal, debug\}$, *k* is the type of event (for example, the start of a transaction, the end of a transaction, activation of a bank card searching, etc.), *m* is additional information (a string constant), *D* is a set of real numbers.

Log usually contains a large amount of events. Some of these events may be insignificant in the context of the case and can be filtered out. Filtering allows on the one hand to reduce the amount of the information that needs to be stored as part of a cases, and on the other hand reduces the matching time. Further, in the filtered log, boundary events are highlighted - such events that start a particular operation. Each received operation normalizes - the time of the very first event is subtracted from the time of all events included in it. Then, among the resulting set of operations, search is performed for the operation described in case. If a similar or identical operation is found (depending on the stiffness of the metric), then the similarity in this parameter is considered to be established.

In the framework described in this work, the algorithm of comparison of operations is implemented on the basis of the transition to the metric point constraint satisfaction problem.

In this paper, we consider the models, based on the presentation of information about time as constraints (dependences) between time primitives. In temporal logics using the concept of constraint satisfaction, information about time is presented as dependences between **temporal** primitives (moments, intervals or their combinations). Dependences between primitives are interpreted as constraints to real time of their appearance. Usually sets of **temporal** primitives and relations among them are presented as the Temporal Constraint Satisfaction Problem (TCSP), which is detailing of a more general Constraint Satisfaction Problem (CSP), what permits to use CSP methods to solve the TCSP. Lets see how temporal case based reasoning can be build on the base of the metric TCSP.

Metric TCSP defined as $Z = (V, D, C)$, where V - a finite set of temporal variables, corresponding to the time points; D - range of values of the temporal variables (the set of integers); C - a finite number of binary temporal constraints $C_{ij}=\{[a_1,b_1],\ldots,[a_k,b_k]\}$, where the intervals are disjoint. Constraint C_{ij} interpreted as $(a_{ij} \le V_j - V_i \le b_{ij})$.

Each constraint C_{ij} defines for the temporal variables V_i and V_j allowable distance between them. Intervals in the constraint C_{ij} are interpreted as a disjunctive [6].

We will represent the operation as $O = (V, C)$, where $V = \{V_1, V_2, \ldots, V_m\}$ is a finite set of temporal variables corresponding to time points; C is a finite number of binary temporal constraints. Each V_i is assigned a record $Rec_i = (t_i, s_i, k_i, m_i)$.

Such a presentation allows to record both the fact of the occurrence of certain events, their order, and the time of their occurrence (metric). The numbering of events in the operations depends on the time of occurrence of these events and their type. Due to this, the process of determining the similarity of operations is greatly facilitated due to the simplification of the analysis of the similarity of time constraints and the assessment of the "overlap" of operations against each other [17].

As a result, to determine the conformity of the two operations O_1 and O2, we will use the following condition:

- the set of time variables O_1 and O_2 coincides;
- for all $C_{ij}^1 \in O_1$ and $C_{ij}^2 \in O_2$ the condition is satisfied: $|lo(C_{ij}^1)-lo(C_{if}^2)|<\varepsilon \wedge |hi(C_{ij}^1)-hi(C^*_{ij}{}^2)|<\varepsilon$, where ε is the threshold value, $lo(C_{ij}) = a_k$ is a function that yields the lower bound of the constraint, $hi(C_{ij}) = b_k$ is a function that yields the upper bound of the constraint.

As a result, the reasoning unit extracts from the case base all cases that exceed a certain threshold similarity value. At the same time, if several alternative decisions have found, VE can operate in two custom modes: either choose a case with a maximum measure of similarity, or transmit a decision on choosing a more suitable case to an expert.

3 Conclusions and Future Work

The paper considers an example of using methods of CBR for constructing virtual employee for a software maintenance department. The prototype of VE was developed in C# in the Microsoft Visual Studio 2017 development environment using the Point Time library and is cross-platform. VE prototype is used in the process of maintaining the automatic payment terminal software. The case base of that prototype currently contains 80 precedents. The prototype of VE made it possible to evaluate the practical benefits of the proposed approach. Currently, work is underway to improve the fullness of the prototype BP. The implementation of VE allows getting an increase in the speed of the initial processing of the issues for maintenance, as well as the economic effect due to the reduction of the burden on specialists.

References

1. van Diggelen, J., Muller, T., van den Bosh, K. Intelligent virtual agents // Proc. 10th International Conf., IVA 2010, Philadelphia, PA, USA. p 28-34.
2. Kurilenko I.E. Case-based reasoning application to maintenance department virtual employee implementation // XVI Int. AI conference RCAI 2018, vol. 2, pp. 238 – 244. RBC.
3. Jungsun (Sunny) Kim, Anthony Gatling, (2018) "The impact of using a virtual employee engagement platform (VEEP) on employee engagement and intention to stay", International Journal of Contemporary Hospitality Management, Vol. 30 Issue: 1, pp.242-259.
4. Sutherland, C. (2013). Framing a Constitution for Robotistan. Hfs Research, ottobre.
5. Madakam, Somayya, Holmukhe, Rajesh M., & Jaiswal, Durgesh Kumar. The Future Digital Work Force: Robotic Process Automation (RPA). JISTEM - Journal of Information Systems and Technology Management, 16, e201916001. Epub January 10, 2019.
6. Lacity, M., Willcocks, L. P., & Craig, A. (2015). Robotic process automation at Telefonica O2. 15. http://eprints.lse.ac.uk/64516/1/OUWRPS_15_02_published.pdf
7. Clint Boulton (2018). What is RPA? A revolution in business process automation, https://www.cio.com/article/3236451/business-process-management/what-is-rpa-robotic-process-automation-explained.html
8. J. Geyer-Klingeberg, J. Nakladal, F. Baldauf, and F. Veit, "Process mining and robotic process automation: A perfect match," in Proc. of the Dissertation Award, Demonstration, and Industrial Track at BPM 2018 co-located with 16th International Conference on Business Process Management (BPM 2018), Sydney, Australia, September 9-14, 2018., ser. CEUR Workshop Proceedings, vol. 2196. CEUR-WS.org, 2018, pp. 124–131.
9. Rajesh, K.V.N. Ramesh and Hanumantha Rao (2018). Robotic Process Automation: A Death knell to dead-end jobs. CSI Communications-Knowledge Digest for IT Community, Volume No.42, Issue No.3, 10-14.
10. Del Fiol G, Michelson M, Iorio A, Cotoi C, Haynes RB. A Deep Learning Method to Automatically Identify Reports of Scientifically Rigorous Clinical Research from the Biomedical Literature: Comparative Analytic Study. J Med Internet Res 2018;20(6):e10281
11. Jurafsky, D., Martin, J.H.: Speech and Language Processing. Printice Hall (2000)
12. Kai H. Chang, Pradeep Raman, W. Homer Carlisle, and James H. Cross. 1996. A self-improving helpdesk service system using case-based reasoning techniques. Comput. Ind. 30, 2 (September 1996), pp. 113-125.
13. ITIL Foundation Handbook (3rd ed.). The Stationery Office, Norwich.

14. Kolodner, Janet. An introduction to case-based reasoning. Artificial Intelligence Review. 6. 1992. 3-34. 10.1007/BF00155578.
15. J. Ramos, Using TF-IDF to determine word relevance in document queries", In First International Conference on Machine Learning, New Brunswick: NJ, USA, 2003.
16. Eremeev A.P., Kurilenko I.E., Varshavskiy P.R. Temporal Case-Based Reasoning System for Automatic Parking Complex // International Journal of Computer, Electrical, Automation, Control and Information Engineering, Vol 2., 2015, № 5, p. 1274-1280.
17. Eremeev A.P., Kurilenko I.E Methods for modeling temporal dependencies in intelligent systems using temporal case-based reasoning // Information Models and Analyses Vol. 2, 2013, № 4, p. 324-335.