INFORMATION
SYSTEMS IN
MANAGEMENT

Information Systems in Management (2015) Vol. 4 (2) 110−122

EXPERT SYSTEMS IN MEDICAL RESCUE

KRZYSZTOF LICHY, ADRIAN ZIEMECKI

Institute of Information Technology, Lodz University of Technology

In the paper the conception of creating expert system in medical rescue is presented. As the result created application is introduced and discussed. This kind of expert system helps medical dispatcher in taking proper action with interaction of caller. Problem of creating such software is actual and there is no existing software to be used for comparison. Application is built in Java with usage of CLIPS system. Efficiency of working application is discussed. The basis architecture of experts system is introduced..

Keywords: Expert Systems, Artificial Intelligence, Medical Rescue, CLIPS, Java

1. Introduction

The aim of this paper is to create a prototype of an expert system which aids medical rescue dispatchers and checks whether such a system is practical in use. The expert system helps in deciding upon a course of action, based on the information gathered from the caller. The document, titled “The project of the initiative to aid the medical rescue system with notable emphasis on medical dispatchers” (pol. “Projekt inicjatywy wspierającej system ratownictwa medycznego ze szczególnym uwzględnieniem dyspozytorów medycznych”), contains the algorithms that serve as the basis for the rules used in the system.

1.1. Purpose of the paper

The main reason while deciding on the topic was the desire to check if expert system is good solution for this kind of problem. AI itself is broad and encompasses a huge variety of subfields. These are both general (learning and perception) and specialized (playing chess, proving mathematical theorems). As such, practical applications range from autonomous systems, like industrial robots, to systems aiding humans in the decision-making processes in business or medicine.

The spark to focus on medical rescue came from learning about the initiative project. The initiative started in March 2013 under the auspices of the Grand Orchestra of Christmas Charity (pol. Wielka Orkiestra Świątecznej Pomocy), with help from the Association of Medical Dispatchers in Poland and the Medycyna Praktyczna publisher [11]. A completed document was handed to the Minister of Health in May 2013. As of the 10th of January 2014, the initiative effectively received green light from the Ministry of Health to begin implementing their proposals.

As a result of these two reasons, the decision was made to see if an AI system to aid medical dispatchers could be created. Since the problem appeared complex and very specific, an expert system was selected as the tool to handle it. Due to the novelty of the selected project, there was no existing application that could be used for comparison at the time.

1.2. Goals of the paper

The main focus of the paper is to utilize the capabilities of expert systems and check their usability for medical rescue dispatchers. The goals are the following:

- Create an expert system in accordance with the information provided by the initiative document.
- Provide a clear and easy to use interface for the user that will work seamlessly with the expert system.
- Check the viability of the created system.

1.3. Scope of the paper

The created prototype is essentially a single application, but it can be divided into three components: a graphical user interface, the expert system itself and a data handler. All of them are necessary for the system to function properly and fulfil its purpose.

The expert system is the main focus of the paper. It is responsible for deciding on the threat level that is present in the current situation and, as a result, what advice is given to the user. The decision is based on the information provided by the user, which he gathers from the caller, and the rules laid out beforehand. The rules

are created with the help of the algorithms contained within the initiative document.

The user interface represents the part of the application the user can directly interact with. As such, it is imperative the interface is clear and intuitive. That way the user will not waste time during input and can easily read the decision and suggestions presented by the system as output.

The data handler is tasked with preparing and manipulating all of the stored data beforehand. It is also responsible for seamless interaction between the expert system and the user interface. This includes correctly transmitting information from the user to the expert system and sending back the resulting decision.

2. Expert systems

The central component of the created solution is the expert system. According to Edward Feigenbaum, "An Expert System (ES) is a computer program that reasons using knowledge to solve complex problems. (...) Traditionally, computers solve complex problems by arithmetic calculation (not reasoning using logic); and the knowledge needed to solve the problem is known only by the human programmer and used to cast the solution method in terms of algebraic formulas". Expert systems are therefore a part of the thinking rationally approach to AI, presented in Figure 1 [3].

The intent of the expert system is to achieve a level of competence in solving problems of a specific domain of work that would rival the performance of a human expert in that field. The need to pursue such systems came from the conclusion that most difficult problems originate from complex physical or social environments and do not have simple algorithmic solutions. Moreover, the people who could be relied on to solve such problems and give accurate expertise were too few to meet the needs. This sparked attempts to digitalize the specialized knowledge and emulate the problem-solving process of an expert. That being said, expert systems are mostly used as interactive aid to humans, rather than given any autonomy in making the decisions on their own [2], [3],[5]].

2.1. Software architecture

Expert systems are a part of the knowledge-based class of computer programs. They use a knowledge base and reasoning procedures to solve problems. The knowledge and methods are modelled on the experts in that particular field. The basic concept of an expert system is presented in Figure 1.

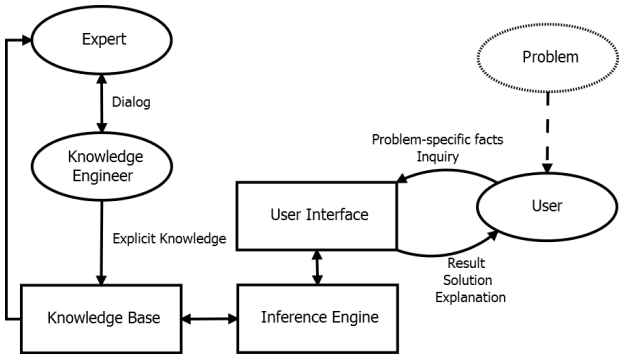

Figure 1. Basic concept of an expert system

The two crucial parts of an expert system are the knowledge base and the inference (reasoning) engine. The knowledge base contains known sets of information about the specific domain. That information is acquired with the help of an expert of the domain in question. The expert works with a knowledge engineer – a person who knows how to code information into the system. The expert's expertise and experience is turned into explicit knowledge by the engineer, with the expert providing comments and feedback on the accuracy and behaviour of the system. The inference engine is responsible for providing results, based on information from the user and the knowledge base. The user communicates with the expert system with the help of a user interface. Here, the user can provide information about the problem and receive a solution, along with the reasoning behind it, in return.

The knowledge in the system can be divided into the following 3 types, based on the source the knowledge originates from:

- Expert knowledge, which includes both factual and heuristic knowledge. Factual knowledge consists of knowledge that is widely shared in a given domain and commonly agreed upon by experts. Heuristic knowledge is the non-rigorous, practical knowledge of the expert.
- User-specific knowledge, which describes the information provided by the user about the problem and the situation surrounding it.
- Knowledge acquired as a result of the processes of the inference engine. This comprises both the final solution, as well as all the intermediary information that was deduced on the way to the solution.

The most popular way of developing expert systems is the rule-based system. Here, the heuristics for the inference engine are represented in the form of production rules, or simply rules. A rule consists of an IF and THEN part. The IF portion lists a set of conditions in some logical combination that needs to be fulfilled for the rule to be applicable. The THEN portion is a set of actions to be executed when the conditions for the rule are met. The process of matching known facts against

the rule condition patterns is called pattern matching. During execution time, the inference engine selects an applicable rule and then the actions of that rule are executed (which may affect the list of applicable rules by adding or removing facts). If there is more than one applicable rule, then the inference engine picks the one with the highest priority. After all the actions of the chosen rule are carried out, another applicable rule is selected. The engine continues this process until there are no valid rules left.

The method of finding viable rules and firing their actions is known as forward chaining. This means that the inference engine starts with a set of known conditions and moves towards some conclusion. Another possible method is known as backward chaining. In this case, the inference engine begins with a known conclusion and seeks out conditions under which the particular line of reasoning will be true [4].

2.2. The initiative to aid the medical rescue system

The Grand Orchestra of Christmas Charity got involved in the initiative to aid the medical rescue system in march 2013. The charity states on its official website that the decision was a consequence of a series of tragic events, which resulted in the death of a number of people in need of medical help, among them children. The designated goal of the project was to change the way the dispatchers gather medical information and to provide them with the tools that would aid them in that task. This would be done by turning the medical inquiry into a set of algorithms to be followed, based on the response from the callers. Ideally, implementing such a solution would aid the dispatchers in their everyday work and increase the safety of those in need by minimizing potential errors. The project was divided into three stages:

- The first stage was concerned with changing the existing regulations, so that it was necessary for the medical dispatchers nationwide to follow the prepared set of algorithms during the medical inquiry.
- The second stage was to prepare the correct algorithms of the medical inquiry and create a handbook and software containing the necessary procedures, questions and recommendations for the dispatchers to follow in their work. It was imperative that this part was prepared with utmost care and professionalism.
- The final stage is concerned with implementing the previous stages into real life, in the form of a training program for the dispatchers. This is to allow the created rules of conduct to be applied as quickly as possible.

The initiative gathered members from both the charity and the Ministry of Health, along with aid from the Association of Medical Dispatchers in Poland and the Medycyna Praktyczna publisher. The former was invaluable in signaling the needs of the dispatchers themselves during each stage of the project. The latter provided

its expertise both in preparing the proper procedures for the dispatchers, as well as the training techniques and necessary tools (among them computer software containing the procedures) [11].

3. Technologies used

Although there were a number of tools to choose from (such as JESS, Drools or Haley Rules), CLIPS was chosen for a number of reasons. First of all, it is maintained as public domain software, which means that the tool can be used without any serious restrictions free of charge. Secondly, over the years of its development, CLIPS has been fully documented. This includes a reference manual and a user's guide, along with an architecture manual, although the latter is slightly out of date. Thirdly, it is highly portable thanks to being written in C. CLIPS has been installed on a variety of computers, ranging from personal computers to CRAY supercomputers. Efforts are even made to adapt CLIPS to mobile systems. Additionally, it is easily integrated with other languages and extended by users if needed. The large number of custom versions of CLIPS, such as integration versions for ADA, Java, C++ or Perl, are proof of its flexibility. Lastly, its popularity throughout the government, industry and academia was also taken into consideration. All NASA sites, branches of the US military and numerous US federal bureaus, government contractors and universities are among the users of this expert system delivery tool [14].

3.1. Java Technology

Selecting CLIPS as the software tool to develop the expert system portion of the project allowed for relative freedom in terms of the programming language and framework to be used for the graphical user interface. Since CLIPS was inherently easy to integrate and was also available to the public community as public software for many years, any chosen technology would most likely have an interface or integration with CLIPS readily available. As such, the decision was mostly a matter of choosing the most subjectively preferred technology, as opposed to being restrained by the expert system itself. For the purposes of this project, the Java programming language was selected to create the GUI [6].

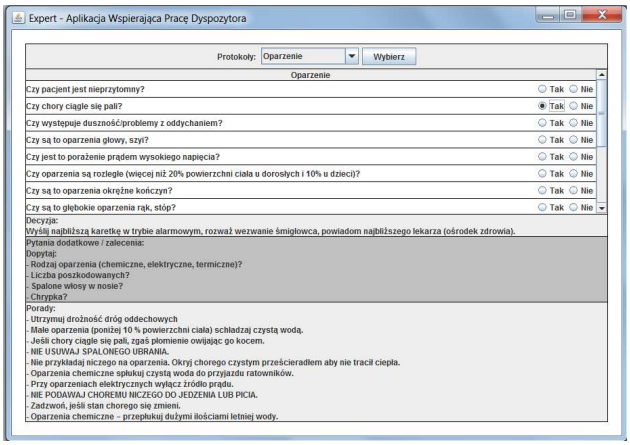

Figure 2. Example of GUI

3.2. Reasons for choosing

Beyond personal preference, there are 3 main reasons for choosing Java as the solution to building a graphical user interface over other programming languages. First of all, the selected solution's inherent portability to other platforms and hardware was a major factor. Created software will work on any JVM in exactly the same way. Further recompilations to meet the needs of a specific platform configuration are not required. Secondly, it is easy to build a GUI in Java. The Java API already has the necessary libraries for creating GUI's – the swing and awt packages. Lastly, prior experience with the CLIPSJNI was a factor in favour of selecting Java.

4. Implementation

As stated in the beginning, the goal of the system was to aid the medical rescue dispatchers in their work, based on the protocols inside the initiative project document. The first order of business was to translate the procedures and protocols of the document into a model interpretable by the system. Then, the necessary functionalities were laid out. Lastly, the application was built, one component at a time.

4.1. Procedures of the medical inquiry

During each medical inquiry, the rescue dispatcher is expected to follow a simple algorithm that will, ideally, allow to acquire all the needed information and

take the best course of action. The algorithm presented in Figure 3 can be divided into two main parts: the general inquiry and the situation-specific protocol.

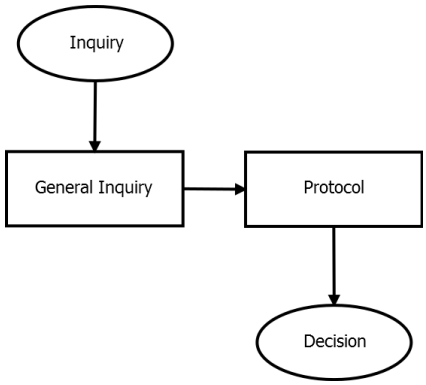

Figure 3. Medical inquiry algorithm

The call starts with the general inquiry. Here, the dispatcher asks for basic information that will allow him to better understand the situation. This includes questions where the incident has happened, how many victims there are and what type of incident it is. This part is done purely by the dispatcher. When the basics have been established, the dispatcher can then select the most appropriate protocol for the described situation and put the expert system into action. With its help, the necessary (if any) action is taken, such as sending an ambulance, notifying the nearest health centre, instructing the caller on how to perform CPR, and so on.

Structure of the chosen protocols

Over 30 protocols for different medical situations have been made for the initiative document. 5 out of those have been chosen to test the validity of the system. The protocols themselves were further divided into 3 subgroups (main medical ailments, accident/injury, urgent condition), but that was not necessary for the test system. As such, the division was omitted, though its later implementation would not be a problem.

The structure of the selected protocols, as described in the initiative document, is presented in the tables below. In general, each protocol has a number of 'threat levels', usually 3. These go from highest (immediate medical attention required) to lowest (no or minimal medical attention required) and decide upon the course of action that should be taken. The patient is categorized to a threat level with simple yes/no questions, where even one positive answer to a question labels the patient to that level. Each level can also have additional questions or other suggestions that the rescue dispatcher might ask the caller to receive more precise information. Finally, each protocol might have some other general tips or instructions that the dispatcher

can give to the caller, if applicable. The following is the shortcut one of protocols taken from the initiative document. (Protocol: Hypothermia)

Table 1. The hypothermia protocol

<table>
<tr><td colspan="3">Protocol: Hypothermia</td></tr>
<tr><td>Questions</td><td>Decision</td><td>Additional questions/suggestions</td></tr>
<tr><td>1. Is the patient unconscious?
2. Is the patient not breathing or the breathing is not correct?
3. Is the patient entangled?
4. Is the patient unable to move on his/her own?
5. …</td><td>YES
↓
Send nearest ambulance in emergency mode</td><td>Ask:
• How long the patient could have stayed in low temperature?
• Is the patient being healed from other reasons?
• …</td></tr>
<tr><td></td><td>NO</td><td></td></tr>
<tr><td>1. Does the patient have any shivers?
2. Is the patient battered, elderly or a child?
3. …</td><td>YES
↓
Send an ambulance (without signals)</td><td>Ask:
• What, if any, medication does the patient take?</td></tr>
<tr><td></td><td>NO</td><td></td></tr>
<tr><td>1. Does the patient have the feeling of cold?
2. …</td><td>YES
↓
Switch to primary care</td><td>• Contact a doctor
• Suggest measuring the temperature in case the patient feels cold (fever)</td></tr>
<tr><td colspan="3">Tips:
• Do not leave the patient without care. If possible, move the patient to a warm place or room.
• If unconscious, then switch to the unconscious protocol.
• If the patient has shivers, then suggest doing simple physical exercises.
• …</td></tr>
</table>

4.2. The data structure of the protocols in the system

Due to the relatively simple and uniform structure of the protocols, it was decided that using a database would pointlessly overcomplicate the system. Hard-coding the data into the project was also rejected, as this would greatly reduce flexibility. Instead, the data is stored inside a text file that is formatted using the eXtensible Markup Language (XML) rules. The reasons for this solution are as follows:

- The XML is an accepted markup language standard that is easy to use. The first version of the XML standard became a World Wide Web Consortium Recommendation on 10th February 1998 and, with modifications, has been in use since [1].
- Loading the data from an external file provides an easy way to extend the number of protocols without the need to recompile the whole system. Adding the rest of the protocols into the system only requires writing them into the file (with respect to the established format) and the application will take care of the rest.

- The standard Java libraries already have a built-in XML Parser, which is more than enough for the task.

The general structure of the .xml file is the following:

```
<Protocols>
    <Protocol Name="Name of Protocol">
        <Tips>
            <Text_Line>Tips text line 1</Text_Line>
            <Text_Line>Tips text line 2</Text_Line>
            <Text_Line>Tips text line 3</Text_Line>
        </Tips>
        <Threat Level="0">
            <Response>Response text</Response>
            <Suggestions>
                <Text_Line>Suggestions text line</Text_Line>
            </Suggestions>
            <Question>Question 1</Question>
            <Question>Question 2</Question>
            <Question>Question 3</Question>
        </Threat>
        [...]
    </Protocol>
    [...]
</Protocols>
```

Figure 4. General structure of the .xml file

All the protocols are stored inside the <Protocols> element, which is considered an obligatory root element of the whole file. A single protocol is embedded within a <Protocol> element. The name of a protocol is stored within the Name attribute. The tips/additional information for each protocol is stored inside the <Tips> element, divided into text lines in case there is a need to uniquely separate the text (for example, a list). As described in 4.1.1, each protocol has a number of threat levels, each consisting of a number of questions. The threat is embedded inside the <Threat> element, with the Level attribute defining the gravity of the threat. Note that the scale of the level goes from highest to lowest, meaning that 0 is the highest threat level. Each threat has a number of questions, embedded inside the <Question> element, as well as their own response and suggestions for the dispatcher (<Response> element and <Suggestions> element respectively). The suggestions are also divided into separate text lines, for the same reason as the tips.

4.3. Activity

The activity diagram depicts the workflow of the application – the sequence of steps needed to be done to accomplish each of the use cases. Due to the low amount of interaction available to the user and their dependency, it was possible to create one integrated diagram of the workflow. Figure 5 presents the full activity diagram of the created system.

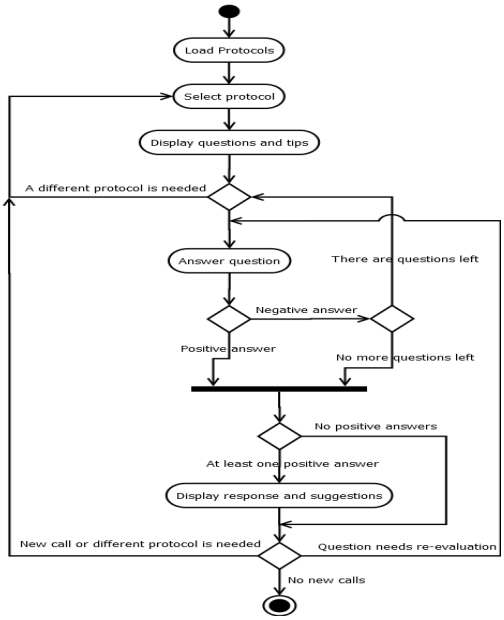

Figure 5. Activity diagram of the application

5. Results and conclusions

The goal of the project was twofold: to create an expert system with a easy to use user interface to aid the medical rescue dispatchers and to check the usefulness of the resulting creation. It means that not only the system needs to provide the expected response for each and every protocol, but also do it efficiently enough to warrant using an expert system.

Thankfully, the low number of possible use cases for the user meant that each and every possible interaction could be tested for errors. In this field, the system does indeed fulfil its designated role: each protocol was loaded correctly, each question was in its place on the list and every answer yielded the expected response and suggestions from the system. There was no form of interaction with the user interface that resulted in unexpected errors. However, the .xml file, from which the protocols are loaded, is very sensitive to mistakes. Incorrect data in that file did lead to unwanted behaviour and errors that made the whole application inoperable. Nevertheless, it can be concluded that the system has been created and works as intended.

However, when it comes to the usefulness of the expert system, the results are not very optimistic. The reason for this comes from the linear structure of the protocols. Expert systems work best in complex environments, with multiple set of

rules that need to be followed and large quantities of data to interpret. Here, the decision path is short and simple and could be easily handled with one loop.

As a result, it is authors' opinion that the expert system in this case is not a good solution. While entirely possible to use, it is simply not efficient when compared to the simplicity of the problem and the needed overhead of the expert system.

Should the application be continued, the best course of action would be to forgo using the expert system and instead implement a method to handle the decision-making process. Other than that, the system could use better protection against errors inside the .xml file the data is gathered from. Moreover, additional space to store information from the general inquiry could be made, as well as a system for saving whole sets of data (general inquiry, protocol(s) selected, as well as answers and the response) into separate files. Due to the way the protocols are stored, adding the rest would prove little trouble.

REFERENCES

[1] *Extensible Markup Language (XML) 1.0*. Available online: www.w3.org/TR/1998/REC-xml-19980210 retrieved 06-10-2014

[2] Feigenbaum, E. (1991) *Expert Systems: Principles and Practice*, Knowledge Systems Laboratory, Stanford University, Report No. KSL 91-79

[3] Feigenbaum, E.; Barr A. (1981) *The handbook of Artificial Intelligence Volume I*, William Kaufmann Inc., ISBN 0-86576-005-5

[4] Fischer, S. R. (1999) *A History of aguage*, Reaktion Books, ISBN 1-86189-080-X

[5] Giarratano J., Riley G. (1998). *Expert Systems Principles and Programming*. Third Edition. Course Technology, ISBN 978-0534950538

[6] Horstmann, C.S., Cornell G. (2003) *Core Java 2 Podstawy*. Wydawnictwo Helion, ISBN 83-7197-984-3 (in Polish)

[7] Kisielnicki J., Sroka H. (2005) *Systemy informacyjne biznesu*. Third Edition. Wydawnictwo PLACET, ISBN 83-85428-94-1

[8] McCarthy J. (2007) *What is Artificial Intelligence?*. Stanford University. Available online: http://www-formal.stanford.edu/jmc/whatisai , retrieved 11-10-2014

[9] Nilsson N.J. (1983) *Artificial Intelligence Prepares for 2001*, AI Magazine, Presidential Address to the Association for the Advancement of Artificial Intelligence

[10] Nilsson, N.J. (2009) *The Quest for Artificial Intelligence*. Cambridge University Press, ISBN 9780521122931

[11] *Projekt Inicjatywy Wspierającej System Ratownictwa Medycznego Ze Szczególnym Uwzględnieniem Dyspozytorów Medycznych*. WOŚP; Medycyna Praktyczna, 2013. Available online: http://skrm.pl/wp-content/uploads/2012/03/Projekt_procedur-WO%C5%9AP.pdf retrieved 15-06-2014 (in Polish)

[12] Russell S.; Norvig P. (2009) *Artificial Intelligence: A Modern Approach*. Third Edition. Prentice Hall, ISBN 0-13-103805-2

[13] Searle J.R. (1980) *Minds, Brains, and Programs*, Cambridge University Press, Available online: http://cogprints.org/7150/1/10.1.1.83.5248.pdf retrieved 20-12-2013

[14] *What is CLIPS?* Available online: http://clipsrules.sourceforge.net/ WhatIsCLIPS.html retrieved 10-10-2014