Thinking Lifecycle as an implementation of machine understanding in software maintenance automation domain

Alexander Toschev[1] and Max Talanov[1]

Kazan Federal University, Russia.

Abstract. The main goal of our work is to test the feasibility study of automation of incident processing in Infrastructure as Service domain to optimize the operational costs of management services that are delivered remotely. This paper also describes a framework that authors have developed to deliver an integrated incident, problem solution and resolution approach as an event-driven Automated Incident Solving System, for Remote Infrastructure Management (RIM) Model. Current approaches are mainly automated scripts, but this is a specific approach for one specific problem. Those systems can't think. Our approach is a system that exploits a thinking model thus can think and can learn. In other words system is capable of recombining its knowledge to solve new problems. Based on Minsky [11] thinking model we have created a machine understanding prototype which is capable of learning and understanding primitive incident description texts.

Keywords: Artificial intelligence, Machine Understanding, Remote Infrastructure Management, NLP, Reasoning, Automation, Knowledge Base, Intelligent Agents

1 Introduction

Our inspiration for this work is MIT Metafor [9, 8, 10], a tool that generates Python classified according to sentences in natural language (shallow English). Our assumption is: human understanding is completely based on thinking. The goal of the project is to study and propose system architectural design that is totally based on a human concept thinking process that is highly optimized for a RIM, Cloud-based offering. We used Marvin Minsky mind model taken from "The emotion machine" book [11] to create a thinking/understanding system, that is capable of learning and as a consequence, solving a range of problems or incidents. We explored different approaches for machine understanding described; for example "Artificial Intelligence The Modern Approach" [14] and found that Minsky's approach that we consider as the most promising [13]. We also performed an analysis of a dump with thousands of incidents taken from Incident Management Systems and found a lot of repeatable and trivial incidents like: "Please install Firefox", "I can't open PDF files" [3]. Also team composition

on project in IT maintenance domain has a lot of low-level specialists [2].
However, in our work we do not limit ourselves to one solution and reuse several frameworks for reasoning, for example, OpenCog [5], NARS [12].
At the current level our framework has version 1.0. We start evolution for automation from Decision tree in version 0.1 and found that this solution is not flexible. (see Menta 0.1 http://tu-project.com/demo/). After that we use Genetic algorithms to found solution, but they are very slow (see Menta 0.2 http://tu-project.com/demo/). In version 1.0 we used Minsky approach and found it suitable for flexibility and speed.

2 Materials and Methods

For the automated handling of events (incident, problem tickets via the first line of Service Desk support) or in Incident Management Systems, the core principle that has been adopted is to classify the highest priority event into an appropriate "class" and take a simple corrective action in response when applicable (e.g., restart a failed process or an interface), followed by increasingly more obtrusive actions (reboot, reimage, mark-as-failed) if simple actions do not fix the fault.
Additional aspects of automation include the creation and resolution (when possible) of problem tickets describing the incident, and influencing the placement engine to prevent provisioning for example of VMs on faulty or overloaded hypervisors.
We developed a framework for incident event-based automation solution search using Sensitive Emotion Machine to understand infrastructure IT domain elements, enhanced with state persistence to maintain a machine self-holistic view of the structure of subject matter, maintenance of event-action history for improved system self-decision making, and fault tolerance for dependability, features typically unavailable in off-the-shelf management systems. The IPM framework is described below.
Artificial Intelligence Approach as a vital core of the presented framework to infrastructure elements servers (hardware and hypervisor software providing virtualization services), network components, and shared storage components as an IT infrastructure Knowledge Base . The current system prototype handles approximately 10 different events. Each instance of incoming incident is used to treat and understand by system a clients infrastructure element from "birth" (deployment) to "death" (removal), based on events received from the Service Desk ticket aggregation and correlation system. An event that represents incidents (faults) in a given IT element is acted upon based on the type of event, the current state of the IT element in the clients infrastructure environment, the history of past events and actions taken on that clients system, and the working policies (instruction) associated with the resulting state transition encoded in the IPM system.
At the early stage we performed research and analysis of incident dump and typical tasks of IT Support team from the large IT Outsourcing Company. We

have analyzed an option to automate several actions of Infrastructure as service domain. Typical domains are:

- Windows Intel (Wintel) domain (Windows operating System)
- Microsoft Active Directory Management
- UNIX management
- Storage management
- We created and implemented architecture for Marvin Minsky model and finally implemented application of it.

We created and implemented architecture for Marvin Minsky model and finally implemented application of it. As a NLP parser we used modified NLP RelEx [6] from OpenCog [5] project. To increase results quality we used public API for grammar correction: After The Deadline [1] , Google. We created learning mechanism based on Minsky [11] approach to be able to train system.

3 Related work

There is no documented evidence of such complex systems that can be collected officially. Even though some vendors of IT support software work in this direction, there is still no data supporting this.
We used results of many researchers in artificial intelligence. For natural language processing we use OpenCog Relex [6] . As a reasoning mechanism OpenCog PLN is used [4].
In 2010 we start our work to solve Software Development automation problem [15], but we found that amount of complex problems in Software Development are much higher than in IT infrastructure maintenance domain. That was the reason to change target domain. However, we use genetics algorithms to solve problem, now the whole approach has been revisited. Previously we have tried different approaches. See section 1.
Due to application context of the work it can be compared to IBM Analytics [7], where the same problem trying to be solved. Also, wolfram alpha contains such knowledge system [16].

4 Solution

We base our solution on the Minsky approach [11] and use $Critic- > Selector- > WayToThink$ triplet, six level thinking model, but implemented only five:

1. Learned
2. Deliberative
3. Reflective
4. Self-Reflective
5. Self-Conscious

Our system supports two modes: learning and operating. To better understand our solution lets make 3 input examples:

1. User Request: ”Please install Firefox”
2. Train request: ”Firefox is a browser”
3. Complex Request: ”Office 2007 installed but Office 2010 required”

According to the Minsky approach, level 1 is at the bottom and level 5 indicates a top thinking level.
Critic represents some kind of trigger. In everyday life individuals face problems and tasks, such as ”Open the door”, ”Eat with a spoon”. Critic is a reaction to this problem. In ITSM domain, for example, when an auto generated incident comes to the queue the Auto Generated Incident critic will be activated. After activating, Critic becomes a Selector. From another point of view activated Critic is a Selector.
Selector is capable of retrieving Resources (Critic or Way to think) from memory.

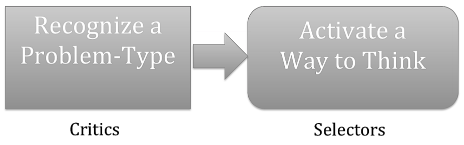

Fig. 1. Critic Selector Way to Think collaboration

Selector can activate another Critic or a Way To Think (see Figure 1).
The Way To Think according to Minsky is a way how a human thinks. For example, ”If I know the problem, I use my experience in analogy situation to solve it”. For example we have several types of way-to-think:

- Simulation
- Correlation
- Reformulation
- Thinking by analogy

Practical example 1. If an incident is automatically generated, the system should process it using instruction book A.
Practical example 2. If a system recognizes the problem, use analogy to solve it.
Way To Think in current implementation is a worker that modifies short term memory.

4.1 Learned level

On the first level the system uses previously obtained knowledge. The knowledge is obtained via training including communication with human expert (See for more info 4.7). On this level there are several types of components used:

1. Preprocess manager

2. Direct Instruction Analyzer
3. Problem With Desired State Analyzer
4. Problem Without Desired State Analyzer

Preprocess manager. This manager activates several Ways To Think to perform initial incident processing. The goal of this critic is to prepare incident description. There are several Way To Thinks:

– Auto Correction of spelling
– Synonymic search
– Annotation finding existing concepts in Knowledge Base

Direct Instruction Analyzer. This Critic is activated when direct instruction is detected in inbound request. For example, "Please install MS Office 2012" is a direct instruction for the system.
Problem With Desired State Analyzer. Critic is activated when a problem with desired state detected by the system. For example, I have Internet Explorer 8 installed, but finance software required is Internet Explorer 7. In other words, Problem With Desired State is a class of problems descriptions that contains current state and desired state descriptions.

4.2 Deliberative level

On the deliberative level systems search for solution of the problem. The solution is stored in the KB as a link to recognized concepts. For example: "Install Firefox" recognized. "Install Firefox" has link to "InstallFirefoxSolution". However, there are could be several solution linked, especially in case of using induction or deduction during reasoning. System will try to apply different solutions, after successful application of a solution (The user estimates if solution successful or not) the score for this solution will be increased and in new request that match the same concepts of this solution will be used as priority one.

4.3 Reflective level

On Reflective level system sets goals and monitors for SLA. On the reflective level the system sets processing goals, performs time control, solution completeness manager. Processing goals. Processing goals required to increase performance of incident processing. They have linked critics, way to think Main goal is a "Help User". There are other goals, that derive from it, for example:

– Resolve incident
– Understand incident type
– Model Direct Instruction

Time control. The time control watches across the thinking levels for time of incident processing. (SLA in terms of ITSM) Solution completeness manager. This manager searches for how-to and solution for current incident.

4.4 Self-Reflective level

On this level the system controls results of lower levels such as initial context or start time control. All communication of a user is also regulated on this level by Do Not Understand Manager and communication with end-user. Human-user communication is activated when system faces unresolvable problem.

4.5 Self-Conscious

On the Self-conscious level the system traces its Emotional State. For example: reacting for long incident processing, system changes emotional state to allocate resources for processing.

4.6 Conceptual Process Stages

There are several conceptual stages in system in cooperation with Minskie's levels. The all concept stages in system processed on 2 levels.

1. NLP Processing (Learned level)
2. KB Annotating (Learned level)
3. Simulation (Learned level)
4. Reformulation (Learned level)
5. Solution Search (Deliberative)
6. Background processes
   (a) SLA control, local resource control (Reflective level)
   (b) User communication (Self-Reflective level)
   (c) Emotional state (Global Resource control) (Self-Conscious)

Background processes is not directly involved in processing stages, but they are very important. Emotional state is very important as a resource indicator. For example, in the "rush" emotional state there are insufficient resources. 6.a for local resource control for single request in comparison with 6.c where are global resource indicators. (System Dashboard).

4.7 Knowledge base

Systems knowledges based on graph of concept. System has 2 base concepts: Action and Object. All other concepts can be trained through the training mode or communication with human expert. For example, in training mode we system can study base concepts.
H(Human Expert): Software is an object.
S(System): Ok.
H: Internet explorer is a browser.
S: I can't understand browser.
H: Browser is a software.
S: Ok.
H: Internet explorer is a browser.
S: Ok.
Such graph representation introduce possibility of applying different logic style:

– Deduction
– Induction
– Abduction
– e.t.c.

4.8 Memory

As a memory conception we used Short-term memory and Long-term memory. Short-term memory is a context for current request. Long-term memory is a global system memory. During the request systems acquires all necessary resources from Long-term memory and put them into Short-term memory context. During the request system can ask help from the human expert. Moreover, using the logic system can produce new knowledge. For example, system knows solution for "MSI software installation", system also knows that "MS Office is a MSI package". As a result system can create link and produce new knowledge "MS Office installation". New knowledge will be stored in Short Term memory (request context). After success request execution Short Term memory will be merged to Long Term memory and will be erased.

4.9 Sample

When first incident comes to the queue, Reflective level sets new goals and system starts processing from 1st level. It processes by NLP and Knowledge Base annotator on the Learned level (stages 1 and 2 See 4.6), which searches learned concept in local Knowledge Base and corrects NLP results. Knowledge Base annotator searches using the generalization and specialization mechanism. In other words, if the system knows browser concept and knows that "Firefox" is a "browser" it will find all concepts linked to browser.
After this system creates simulation for a complex problem (stage 3 and 4). In our example this is Case 3. Simulation creates problem description and finds that office 2010 required.
On the Deliberative level system searches for solution (stage 5).
In the second case Firefox is a browser system processes Stage 1 and 2. Then the system searches for known concepts and creates links between unknown and known concepts between Firefox and browser in our case.
In 3rd case "Office 2007 installed but Office 2010 required". System proceeds through Stage 1 to 5. Detailed example can be shown as dialog:
H(Human user): Please install firefox.
S(System): *Stage 1. Process request through NLP* (Italic means hidden for user actions).
S: *Stage 2. System finds concepts in database. For "please" - "form of politeness", for "install"-"install action", for "firefox"-"FirefoxConcept".*
S: *Stage 3, Stage 4 skipped because problem is direct instruction.*
S: *Stage 5. Find solution "InstallFirefoxSolution".*
S: Done. Does solution suitable for you?
H: Yes.

5 Conclusions

We prove an approach of automation in case of simple incidents. As a result, we have created architecture for Minsky model and created a proof of concept for the automated incident solving. We analyzed incidents in IT Outsourcing domain and found 3 types of typical incidents [2]:

1. Direct instruction (Please do something)
2. Problem with desired state (I have A but I need B)
3. Problem without desired state (I dont have B)

We processed 3 types of control incidents 2. However, the ability to learn can easily extend these types. Additional demo can be found at http://tu-project.com/demo/. The dump for the research contains 1895 incidents collected from IT maintenance company dump for 3 month. We studied dump and found 43 category. All of categories are objects from base class of problem (see 4.1). The categories have been assigned by technical specialists from IT maintenance team. Top most:

- Invalid login
- Connectivity problem
- Can't find software
- Reinstall software
- Wrong application has been installed
- Insufficient disk space
- Shared disk connection required
- Invalid computer time
- Setup Wifi
- Add Alias for host
- No access to printer
- Insufficient rights
- Need additional info
- Clean profile after migration

We select 45 incidents over the all categories and process them using the prototype. The results presented on 2.

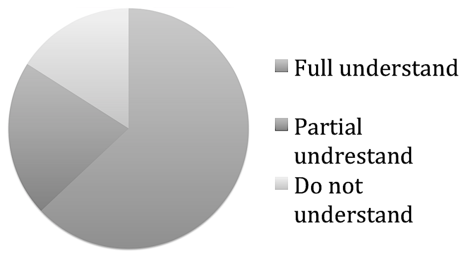

Fig. 2. Incident understanding diagram

Processing result	Quantity	Summary
Full understand incidents	63%	All concepts have been understood
Partial understand incidents	21%	Some concepts do not understand correctly, but solution can be found
Do not understand	16%	Most of the concepts haven't been understood, solution can't be found

Table 1. Incident processing results numbers

5.1 Deployment Experience

The Automation Incident Handling component is currently deployed in VM infrastructure and can be presented for operational build in a six-eight months period. The scalability and persistence features of commercial version will meet the current and according road map design goals during steady state operations and as it based on business requirements to system under Remote Infrastructure Management business model.
However, implementation based on Java platform and can be easily run on different platform like:

- Windows
- Linux
- Mac OS X

Prototype implementation supports multi-threading as part of architecture and can be easily deployed on multiprocessor systems.

References

[1] After the deadline. web (04 2012), http://www.afterthedeadline.com/
[2] Alexander, T.: Thinking model and machine understanding in automated user request processing. CEUR Workshop Proceedings 1297, 224–226 (2014)
[3] Alexander Toschev, M.T.: Analysis results for it outsourcing. web (04 2013), http://tu-project.com/for-business/
[4] Goertzel, B.: Probabilistic Logic Networks. A Comprehensive Framework for Uncertain Inference. Springer (2008)
[5] Goetzel, B.: Opencog. web (04 2012), http://opencog.org/
[6] Goetzel, B.: Opencog relex. web (04 2012), http://wiki.opencog.org/w/RelEx
[7] IBM: Ibm watson. web (00 2014), http://www.ibm.com/analytics/watson-analytics/
[8] Lieberman, H., Hugo Liu, in H. Lieberman, F.P., Wulf, V.: Feasibility studies for programming in natural language,. End-User Development (2005)
[9] Liu, H., Lieberman, H.: Metafor: Visualizing stories as code. In: IUI 05 - 2005 International Conference on Intelligent User Interfaces (2005)
[10] Liu, H., Lieberman, H.: Programmatic semantics for natural language interfaces,. In: ACM Conference on Computers & Human Interaction (CHI-2005) (2005)
[11] Minsky, M.: The Emotion Machine. Simon & Schuster (2006)
[12] Pei, W.: Rigid Flexibility: The Logic of Intelligence. Springer Netherlands (2006)

[13] Sloman, A., Chrisley, R.: Virtual machines and consciousness. Journal of Consciousness Studies (2004)

[14] Stuart Russell, P.N.: Artificial Intelligence. A Modern approach. Pearson (2010)

[15] Talanov Maxim, Krekhov Andrey, M.A.: Automating programming via concept mining, probabilistic reasoning over semantic knowledge base of se domain. In: 2010 6th Central and Eastern European Software Engineering Conference, CEE-SECR 2010 (2010)

[16] Wolfram: Wolfram alpha. web (00 2014), `http://www.wolframalpha.com/`