An Approach to Build Self Managing Infrastructure Using Machine Learning Techniques

Vinay Bhat[1], J.V. Vadavi[2], G.A. Radder[3]
[1]*Student M.Tech, Dept of Computer Science, Shri Dharmasthala College of Engineering & Technology, Dhavalagiri, Dharwad-580002, Karnataka*
[2]*Associate Professor, Dept of Computer Science, Shri Dharmasthala College of Engineering & Technology, Dhavalagiri, Dharwad-580002*
[3]*Assistant Professor, Dept of Computer Science, Shri Dharmasthala College of Engineering & Technology, Dhavalagiri, Dharwad-580002*

Abstract: In this project we propose to implement an approach to building self-healing/correcting software using either backend service algorithm/software agents. It is well known that softwares suffer from different kinds of errors during runtime. Many softwares nowadays generates log files and are dependent on configuration files and users can modify these configuration files to customize the software. However, the user may unknowingly introduce errors while configuring the files in which case the software generally exits abruptly by logging error information into the log files. We propose to solve this problem by having a software agent running in background to monitor the configuration files and log files. Whenever it finds an error in the configuration file or error information in the log files, it can either replace with a backup copy of the configuration files so that the application overcomes the problem or in case of absence of a back up of the configuration file, it will be able to convey the same to the main application or administartor about the error who will then takes an appropriate measure manually. Such an approach is more user friendly and thus desirable. We propose to implement this using Python programming language.
Index Terms- Agents, Rule-based Learning, PyQt5, Self-healing, Self-protection.

I. INTRODUCTION

Self-managing infrastructure enables Self-healing or Self-correction to economize healing process, and allow applications to run continuously with high performance without or with less human intervention. Self-healing infrastructure[11] has ability to handle error on its own using some backend algorithms or service which keeps monitoring the execution environment of the application. Self-protection[11] means the system should have ability to react consistently and continuously to any situations i.e. from unusual events to external attacks. Here we will use an approach towards machine learning techniques by using either service algorithm or software agents. It extracts the high performance of the running applications, while correcting the errors for reliability and data movement. The system can accumulate knowledge about the application environment and act on it to adaptively detect system/application malfunctions and stimulate a corrective action themselves.

II. LITERATURE SURVEY

IBM proposed an autonomic computing in the year 2001[1] with the purpose to develop self-managing systems. IBM has defined following four functional properties that must be present in a system to be called as self-management systems. They are: Self-configuring, Self-healing, Self-optimization and Self-protecting. These objectives represent the system requirements, basic implementation mechanisms that can be identified by attributes such as self-monitoring, self-awareness and self-adjusting.

It is quite difficult to implement complete implementation of self-management infrastructure simultaneously at a single shot due to its complexity, but it is possible to implement all the four characteristics independently[12]. So, in this project we propose an idea to implement an approach towards building self-managing application which implements few of the characteristics of self-managing system i.e. self-healing and self-protection property. But as discussed above self-managing characteristics can be implemented independently and such independent implementations of all four characteristics in future can provide a motivation for complete integration of self-managing systems..

III. SCOPE OF WORK

The idea is to implement one of the characteristics of autonomic management of the computing infrastructure[2] to the application intent with regards to application performance, reliability and security[12]. Here we implement an algorithm/software agent[5] which runs as backend service so that it can keep monitoring our application and understand the domain knowledge of the application running. Software agents use artificial intelligence through rule-based learning. Rule-based learning simulate intelligence without having the ability to learn. It uses rules as the knowledge representation and these rules are implemented in the code in the form of if-then-else statements. Key elements of this process include the ability to understand application through its configuration files and keep monitoring. Now if the user introduces an error in the configuration files, then software agent/service running in the background as an independent worker thread can detect the error and take a decision to replace the backup copy of the configuration files (if found) as a solution to the problem of exception thrown by application. In case if the backup copy of the configuration file is not available then it should be able to convey the message to the main application or administrator about the error. User can then take an appropriate measure manually.

IV. IMPLEMENTATION

4.1 Developing Text Editor as Base Application for Implementing Self-Managing Features.

At the first, an editor application is built using python GUI module PyQt5[10]. Qt based applications are event based. In this event-based application, clicking on a widget creates signal to which respective slot functions are defined as a signal handler. All such events are queued in event loop. The main application is executed as a main thread. Execution triggered by event also runs within thread that should be synchronously executed with this main thread. Such threads share the same resources in order to communicate. But sometimes, operation within threads may take sufficient time without passing the control to the main GUI thread which will freeze the main application thread. Hence, we go for multi-threading concept where two threads communicate through events. Events triggered emits signal for which signal handler (slot functions) are defined. So to start with firstly create a thread pool and start a main thread for the editor window application[13]. From this main thread various other threads from the thread pool are invoked for various tasks and for each of these tasks signals are emitted and for each signal respective slot functions are defined and connected through connect() method.

This editor application is used as base for implementing self-protecting and self-healing features of autonomic computing. Overall the idea is to implement an approach to build self-correcting and self-protecting software using software agents in order to minimize human intervention. This editor software is dependent on its configuration files and generates log files for each and every operation that take place in the editor.

4.2 Developing Agent worker module for implementing auto-correction property

Agent worker module[3] is implemented as thread application that gets invoked through main thread editor application as an independent module, runs in background by continuously monitoring the editor application and configuration file for the changes or modifications. These agents use rule-based learning technique to simulate intelligence. Any changes in environment will be logged into the log file. In this process, user may unknowingly introduce errors in the configuration files. Such errors are detected and auto-corrected by the running agents. Agents are implemented using multi-threading technique and PyQt module.

Qt provides simple interface for running jobs in other threads say Agent worker module. For this, there are two classes say QRunnable and QThreadPool[3]. To define custom QRunnable we subclass from QRunnable base class and place the code inside this run () method. Our function is executed automatically by creating instance of worker and pass it to QThreadPool instance. Worker class handles all of our jobs. Now pass any user defined python function that executes as a separate thread. Qt also provides signals and slot framework that allow safe thread communication between GUI and running threads. Signals are emitted which is taken up by slot functions. Slot functions are linked with connect () method. For communication to occur, we define custom signals say finished signal, error signal, result signal. Connect the signal handler functions to these signals in order to receive notifications of results/error of threads.

4.3 Proposing the solution for the problem.

In this process, we propose the solution as to replace the erroneous configuration file with an original configuration file whose backup is maintained in the same directory structure. This process of finding the path to configuration file and replacing it with the backup copy is completely automated by using artificial intelligent concept of machine learning i.e. using software agents[6]. These rule-based agents simulate intelligence through if-then-else case statements and are responsible for sensing the changes in the working environment. Agents takes an appropriate decision of replacing file towards the exception thrown by the main thread of application as a solution to the

problem. In case of absence of backup copy of configuration file, an appropriate message is displayed to the user or administrator so that administrator can solve the problem manually.

The above approach is implemented using three methods that gets executed in the thread.

a. Call Back method which is responsible for getting the status information from running threads. This method loads and scans the config file of the application as JSON file data entries.
b. Progress Function method which is responsible to set the settings of the configuration file to the application.
c. Restore Back Up function method which replace the erroneous configuration file with the original back up copy in order to make the application continue to execute without any abrupt termination

V. ARCHITECTURE AND OPERATIONAL STEPS

The application consists of following components:

Key Components:

i. Main Editor Application
ii. Config Files.
iii. Agent Worker Thread/ Service Program.
iv. Log files

Main editor application supports following functionalities

- File management
- Rich-text formatting
- Table insertion and management
- Find-and-replace
- Inserting images
- And more!

5.1 Architecture Diagram

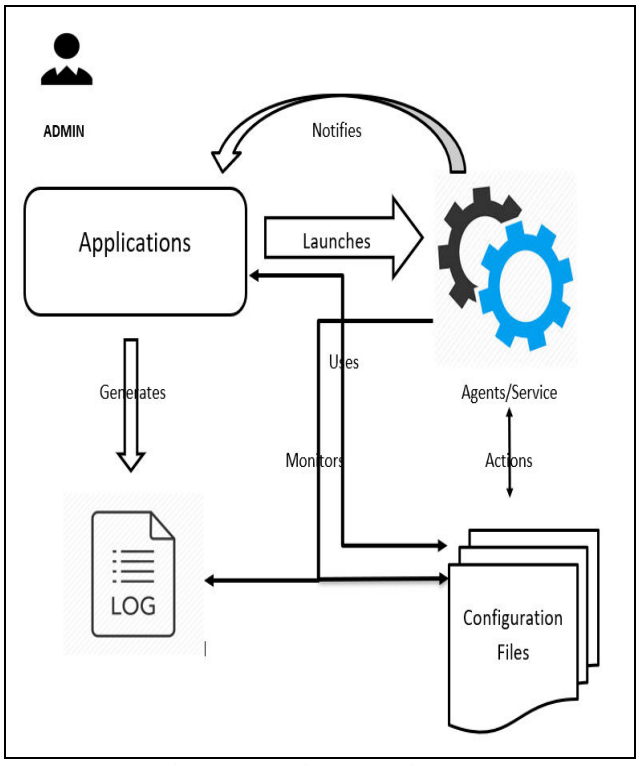


Figure I Architecture Diagram

5.2 Flow Diagram

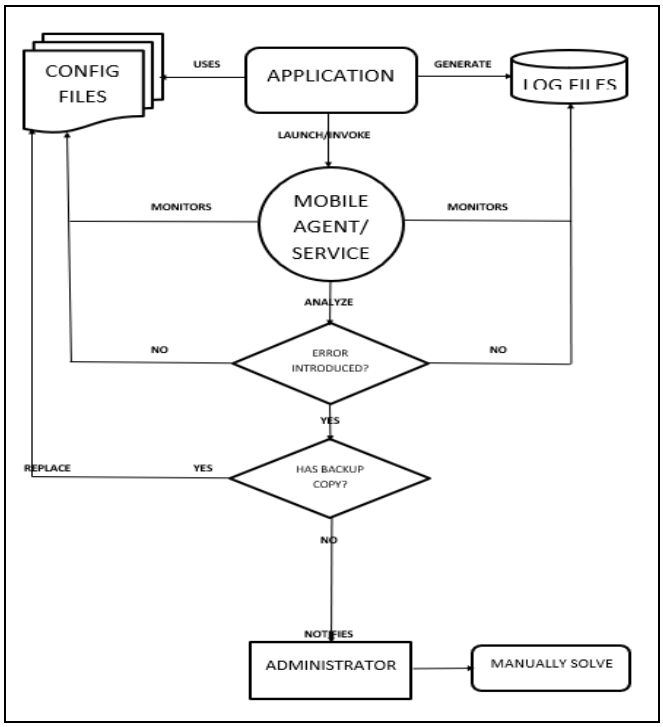


Figure II Work Flow Diagram

5.3 Operational Steps

- Application is developed and executed using configuration files.
- Running application invokes intelligent software agents[8].
- Agents run as separate thread in the back ground mode.
- Agents keeps monitoring or analysing configuration and log files.
- Error is introduced in configuration file by user.
- This process of introducing error is logged in log files.
- Configuration file now has error which is detected by agents. Agents now looks for backup copy of a configuration files to replace it as a solution to the problem.
- If found, replace it else notifies an administrator with an appropriate message so that he/she solves it manually with certain measure.

VI. SEQUENCE DIAGRAM

Sequence diagram below depicts two cases:
Case 1: Exception / error that cannot be resolved automatically by the agent
In this case user is prompted with a pop up message saying that error is not rectified, hence user should manually rectify the error.

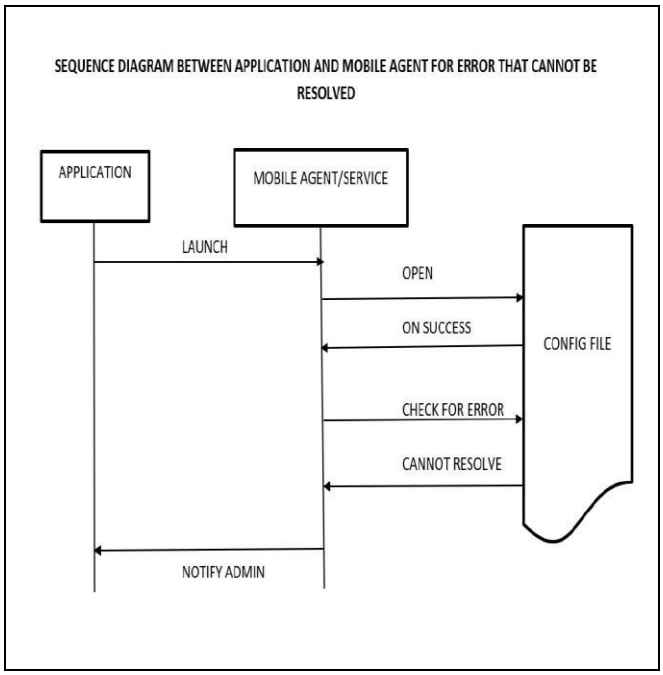


Figure III Sequence Diagram – I

Case 2: Exception /error that can be resolved on its own with no human intervention
In this case user is prompted with notification message saying that the error has been rectified. Application continues to execute without any abrupt termination. Thus, user understands that the error is solved on its own or self-healed.

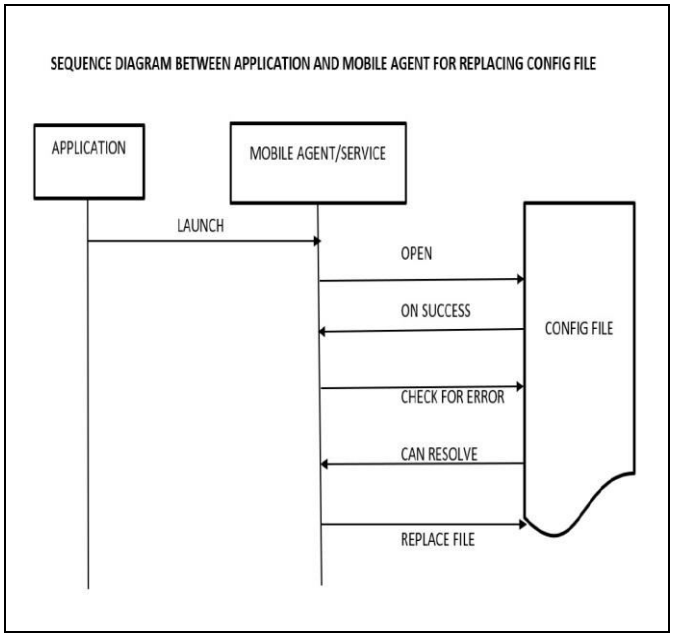


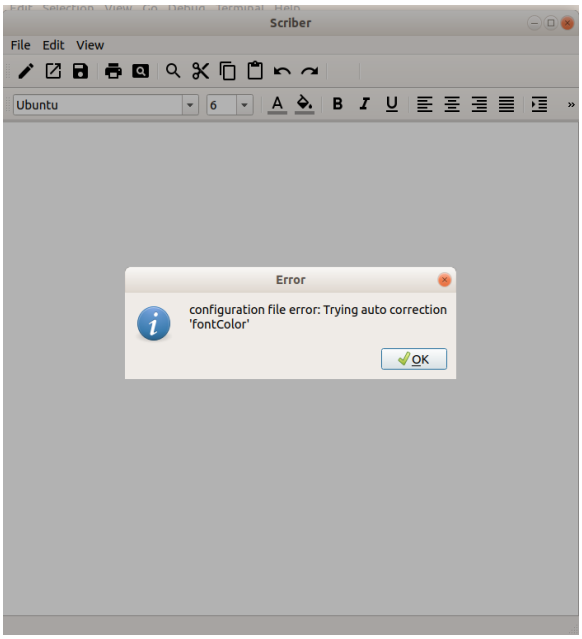
Figure IV Sequence Diagram - II

VII. OUTCOMES

By using machine learning techniques through software agents, we build an approach towards the application self-protection and self-healing with minimal human engineering, with the potential to accelerate application performance and reliability[12].

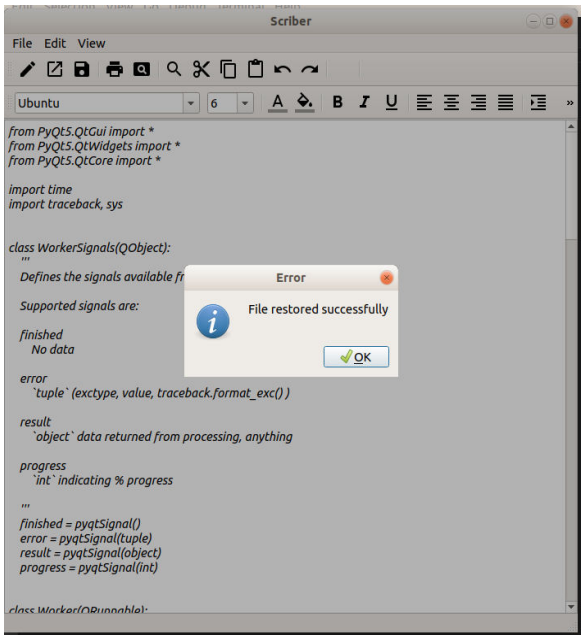
Steps towards Self-Protection/Auto Error Detection

Application has its own configuration file which consists of application settings in JSON data format.
Introduce error in the configuration file manually by the user.
Agent worker thread which is instantiated through main editor application thread keeps monitoring configuration file for the error, hence pops an exception when user introduce an error.



Steps towards Self-Correction / Self-Healing

When an exception is thrown by the application, agent thread looks for the configuration back up file so that if found the same can be replaced with the erroneous file.
Replacing configuration file takes place on its own by the agent at the back ground.
Thus, keeping main application in execution mode continuously, without an abrupt termination.
If the replacement of configuration file fails then a pop-up message is thrown to the user so that user can solve the problem manually.



VIII. DESIGN ISSUES

Various Self-managing system or autonomic computing system design approach has been proposed but only few has gain success because of its complexity[9]. Hence, here we tried an approach to build self-managing system by implementing self-protection and self-healing feature which are autonomic computing system characteristics. This paradigm of self-protection and self-healing has been implemented using software agents or service algorithm (using machine learning techniques) that runs continuously as a thread in the back ground mode by monitoring applications configuration files and logging each and every event into log files.

IX. CONCLUSION

Self-managing computing systems are implemented by enabling self-protection and self-healing property. This paradigm can be met by implementing a text editor application. This editor application runs as a base application through which an algorithm/software agent is instantiated as a separate thread that uses its artificial intelligence knowledge[14] (simulate intelligence by rule-based learning) to sense the application configuration file .Such rule-based learning has very limited ability of its underlying knowledge base. So, in our project software agents are restricted to fixed limited rule-based knowledge. Such agents detect error when it gets introduced in the configuration files (either syntax or logical errors) by modification in configuration file. agent runs in a back ground mode takes suitable action to replace the modified configuration file with the original configuration backup file such that application continues to run without an abrupt termination. This adoption model can be used as a measure to evaluate systems self-protecting / self-healing capability. Key benefits of this approach are less human intervention, better application performance and reliability.

X. FUTURE SCOPE

As of now, the project implements only self-protection and Self-correction characteristics of Autonomic Computing system[12]. In future, one can try to implement a greater number of features of self-computing or autonomic computing systems say Self-optimization, Self-configuration on a single application. The application used here is a text editor application which contains only few important functionalities, hence in future one can upgrade the application with more advanced functionalities so that one can implement all features of autonomic computing in a single application with much more specific and better results.

Interested user can think of to continue his/her work in taking this self-managing approach of a system to an enterprise level by implementing much more complex algorithms. This will definitely reduce human intervention in managing the whole/entire system at an enterprise level with more optimized results.

Currently, my project explains how one can come with an approach towards self-managing system using rule-based machine learning concepts like artificial intelligent agents[7] on a user-built editor application that runs based upon its configuration settings file.

As the technologies grow, one can increase his/her knowledge in the domain of machine learning thus making machines more capable of in understanding advanced/complex algorithms which will increase the accuracy of the machines in predicting the results.

XI. APPLICATION

1) Self-healing property of the application can be extended towards home appliances, Personal computer, next generation mobile phones. This will help users having less knowledge and will help users to solve the problems on its own.
2) Scope of Self-protecting property is used against proactive identification and protection from malicious attacks against application data, assets and other resources.
3) Unauthorized access of files related to critical applications can be avoided. Such unauthorized access might corrupt the application if the user deliberately modifies the files of the running applications.
4) Multi-thread agents can be used in distributed computing in order to optimize the utilization of resources and maintain equal work load balance across the various applications.
5) Rule based learning approach of the application can be used across various fields of domains such as insurance for insurance rating, financial services, government offices for tax calculation, telecom customer for billings, e-commerce for recommendation systems etc.

XII. ACKNOWLEDGMENTS

During my project , the staff at SDM College guiding me were very helpful and extended their valuable guidance and help whenever required for the project which I worked on.

I am thankful to Prof. G. A. Radder, Assistant professor and Prof J.V. Vadavi, Associate professor (Guide, internal) for their complete guidance and friendly support and advice during my period of internship.
I am very thankful to my HOD Prof. S. B. Kulkarni, Associate Professor for his invaluable guidance and advice during my project.
I am thankful to Prof. J. C. Karur, Associate Professor, PG Co-ordinator-MTech for her guidance and support during my project.
I am also very greatful to Prof. Basavraj Vaddatti, Assistant Professor and Prof. Anand Pashupatimath, Assistant Professor, Project Co-ordinators for their continous support during my project tenure.
Overall , the above team helped me to complete my project successfully and made my learning an enjoyable one and I am greatful to them for making it so.

BIBLIOGRAPHY

[1] Autonomic computing: IBM's perspective on the state of information technology - http://www.research.ibm.com/autonomic/manifesto/autonomic computing.pdf
[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing, "Computer, vol. 36, no. 1, pp. 41–52, 2003.
[3] Multithreading PyQt application with QThreadPool https://www.learnpyqt.com/courses/concurrent-execution-threads-processes/multithreading-pyqt-applications-qthreadpool/
[4] IBM, An architectural blueprint for autonomic computing, April 2003.
[5] Nicholas R. Jennings, On agent-based software engineering, Artificial Intelligence, v.117 n.2, p.277-296, March 2000
[6] N. Jennings and M. Wooldridge, Intelligent agents: theory and practice, The Knowledge engineering review, 1998, 10(2)
[7] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 2nd ed. Prentice Hall, 2003.
[8] S. Murugesan, "Intelligent agents on the Internet and Web", TENCON '98. 1998 IEEE Region 10 International Conference on Global Connectivity in Energy Computer Communication and Control, vol. 1, pp. 97-102 vol.1, 1998.
[9] Autonomic Computing Systems: Issues and Challenges - https://www.researchgate.net/publication/228997960_Autonomic_Computing_Systems_Issues_and_Challenges
[10] PyQt5 Reference Guide - https://www.riverbankcomputing.com/static/Docs/PyQt5/
[11] Security in an autonomic computing environment - https://ieeexplore.ieee.org/document/5386832
[12] Autonomic computing: A revolutionary paradigm for implementing self-managing systems - https://ieeexplore.ieee.org/document/6146831
[13] Building a Text Editor with PyQt - https://www.binpress.com/building-text-editor-pyqt-1/
[14] An AI Approaches compared Rule based testing vs learning - https://www.tricentis.com/artificial-intelligence-software-testing/ai-approaches-rule-based-testing-vs-learning/