ISSN 2047-3338

A Case Study: Development of Auto-configuration Server

Hammad Ali Butt[1], Junaid Arshad[2] and Ehsan-ul-Haq[3]
[1,2,3]University of Engineering & Technology, Lahore

***Abstract*—In this study, we are discussing our experiences and challenges we faced during the development of an Auto-configuration Server (ACS) for TR-069 protocol. During the development we studied many open-source products regarding ACS. We learned and analyzed them and used some of code in our project. During study we found that those products are not developed maturely and also not properly documented, so we need to go through the project by understanding the source code to find out the product architecture and process flow. This paper also discusses how we faced those challenges and solve the by using different software engineering techniques. We studied architecture of one the best product among the list and how we change it into real time product for industry. We are sharing those experiences that are helpful and provide some guidance for others who are planning to develop such kind of system.**

***Index Terms*— Auto-configuration Server, Open-Source Software, Software Engineering Tools & Techniques and TR-069**

I. INTRODUCTION

THE life cycle of large scale open source projects is very long and multiple developers are involved in development process [1]. The development of open source projects goes through several phases like code-level peer reviews and field testing [2], [3]. The process of bugs fixing and highlighting is very rapid as so many people are involved in using, developing and experimenting with these software [4]. The interest of the developers also fluctuates over time. The other important thing is the availability of the documentation of the existing system. If the documentation is not available or is not in proper form then the developer finds it very difficult to participate in development community. For this purpose of case study we take an example of a server which we developed, this server which allow service provider to manage their devices remotely.

A. Auto Configuration Server

The Automatic Configuration Server (ACS) is software that is used for managing the TR-069 enabled devices and their statistics [5]. These devices are Customer Premises Equipment (CPEs) that are provided by the service providers to their customers for getting access to the Internet [7]. The service providers need monitoring and management of their CPEs. These management and monitoring tasks include recording the statistics, updating the configurations of CPEs, upgrading firmware on CPEs.

B. TR-069

TR 069 is a technical specification is defined by Broadband Forum [5] [6]. It is also known as CPE WAN Management Protocol (CWMP). It defines that how CPEs will communicate with ACS server and how ACS manages CPEs remotely. Both ACS and CPE communicate based on bidirectional SOAP/HTTP messages.

II. CASE STUDY

One of the leading broadband services providing company of Pakistan approached our organization for the purpose of development of ACS according to their custom requirements. The customer required to manage the CPEs in an automated and efficient way. The company required an ACS for managing their 10's of thousands of CPEs, recording their stats, generating reports and updating the firmware on their devices. The initial requirement of the company was to implement the ACS for CPEs working on TR069 protocol which can manage up to 30,000 CPEs initially and they wanted to extend 100,000 CPEs.

We started our research by finding the existing open source ACS for TR069 protocol. They were very few open-source ACS that we found on Internet, all of them are just only the complete or even partial implementation of TR-069 protocol [15]. No product was able to bear actual deployment load under real circumstances. But after initial version of Open ACS, it's been improved a lot in newer version. It's not only the most matured product then all but also addresses scalability issue. Product documentation is poor as others but a lot of deployment help and material is available. Also many researchers showed their confidence to use this product for their experiment validation [24], [26].

- clj. TR-069
- Perl CWMP
- Open ACS
- Grail ACS
- jCWMP Server

- freeacs-ng
- TR-069 D-Link

Here is a comparison in Table 1 between these open source ACS servers mentioned above.

Table 1: Comparison between open-source ACS

ACS Server	Implementation	Scalable	GUI	Documentation	Language
clj. TR-069 [8]	Partial	×	✓	×	Clojure
Perl CWMP [9]	Basic	×	✓	Basic	Perl
Open ACS [10]	✓	✓	✓	Basic	Java
Grail ACS [11]	Basic	×	×	×	Groovy
jCWMP Server [12]	×	×	✓	×	Java, php
Freeacs-ng [13]	partial	Not now	Not now	×	SCGI
TR-069 D-Link [14]	✓ (but only for D-Link clients)	✓	✓	✓	Java

We started working on ACS in 2010, at that time only the initial version of open ACS was available. In initial of version Open ACS GUI was not well matured and also scalability issue was not addressed till then also many functional operations were needed to be automated. After selecting OpenACS we understood the working of TR069 protocol. The next step was to code the server software for this purpose we picked the desired code portions from the OpenACS and designed architecture for real time load.

We use java technologies to develop our product. Programing language was java and application server was glassfish server, JMS queue is used for buffering the request on server during peak time. MySQL is used as DBMS. All tools and technologies are free and open-source.

The figure below is representing the architecture of the TR069 Server as developed by us. Server and CPE communicate with each other by exchanging SOAP Messages. Server receive request from CPEs, then identify the type of the message and sends a predefine response message according to request or requirement. The patterns of request and response messages are defined in TR069 protocol.

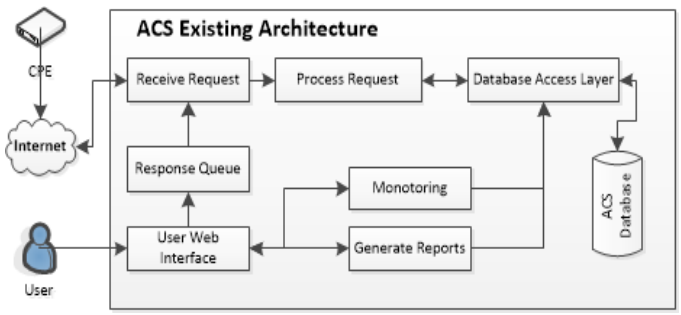

Fig. 1: Auto-configuration Server Architectures

Those response messages contain the information regarding which method is to be invoked and for some methods it contains the parameters and their values. The received messages can be a message that contains CPE statistics or response of any invoked method. CPEs are configured to send their information to a server after some defined interval of time. In our case one CPE sends its statistics after every fifteen minute. If each informs request average is equal to 5KB to 5.5KB then 30, 000 CPE can generate traffic 150 MB of real-time textual data during the interval approximately. Some other requests which are far-more greater than inform request is not considered. Currently, more than 200,000 CPEs are registered on the server. We know CPE is a machine so its need proper response of request after some interval of time otherwise CPE may behave abnormally. Some CPE refuse to send their statistics until next reboot other generate lot of traffic with similar information on the server. So server should be responsive in peak load as well in normal circumstances.

III. DEVELOPMENT EXPERIENCES AND SUGGESTIONS

In this section, we are presenting our experiences that we faced during the process of the development of the ACS. We are reporting here the suggestions for the avoiding the problems that we faced during the course of development.

A. Understand the legacy code

There is a lot of legacy code available on the Internet without proper documentation. Those codes might be helpful for many software engineers or developers but understand that code might be complex and time consuming. Developer can reverse engineer the code of product and can extract design of the code by using existing software engineering techniques and tool to understand code [18]. There are some logical and software engineering technique and methodologies to understand code

- *Static code analysis* (Like call graphs provide flow of the program) [16].
- *Dynamic code analysis* (Like profilers provide performance analysis) [17].
- *Divide and Conquer* (By inducing simplicity)
- *Error messages* (Review error messages document to understand reason of errors) [1].
- *Logging and Printing* (Generate proper log in system if missing, they also provide valuable information and provide system state during execution) [23].
- *Debugging* (Many free debugging tools provide forward/reverse debugging and real time footprint of the software)

B. Robustness

As we discuss earlier that CPEs are machine which sends data to server after some specific interval of time and during peak time we cannot make CPEs that do not send more request until ACS become free. And if we deny those request CPEs may respond abnormally, which may be worse than normal

behavior.

The server that we have developed was required to be robust enough as the company was planning to increase their devices in future. The traffic load in initial stages was also very high and the requirement was to design the architecture that can handle the rapidly increasing traffic load in the future. Designing a server for such devices it should be first priority to plan the architecture in such a way that makes the server robust. To manage such load we add a buffer between request receiving and its processing module. So, if in peak time the receiving requests are more than process requests then it provide a cushion between both modules. In off-peak time when number of receiving request are no more greater than the number of process request in system then buffer will become freed gradually. Also we made server communication with database asynchronous, which helps application server to become more responsive.

C. Scalability

OpenACS initial version was very limited; it only provided simple implementation of the TR-069 protocol generically. It was not designed for any specific product. There is no option to handle vendor specific attributes in data model if devices belong to different vendors. Also it was not designed to bear load of CPEs when they are large in number.

The CPEs from other hardware vendors were supposed to be added in the system. Initially, we have developed server for Gemtek devices and after some time Motorola, ZyXEL and sagecom were added. So the new ACS server should be scalable which can add more feature and devices and should allow different vendors CPE to be register on server easily.

D. Playing with real hardware

During the development we found that there some devices from different vendors some of them also provide ACS services to their customers. So implementation protocol to communicate with server is not as exact, as defined in the specification of TR-06 protocol. Some vendors defined their own flow of communication and some has different data model. We needed a server which not only implement TR-069 server protocol but able to handle all requests from different CPEs smoothly and transparently.

E. Database Design

Data is predefined and insertion is automated so chances of anomalies are limited also in the trade-off between performance verses quality, we focused more on performance because in this type of environment, the performance of the application is important than the database integrity.

So, if database de-normalization and results in maximizing the performance of the application then it can be done [19], [20]. The other important thing is to shift the load from application server to DBMS as much as possible by handling transaction using store-procedure and triggers. The stored procedure and trigger runs on DMBS which tends to reduce the load from application server. The application server will remain busy with CPE request management, instead of doing transactions with DBMS server.

Reporting of such system also generates load on system because data is huge so data processing will take time to generate reports. So for routine reports, developer should provide facility of precompiled report and user just has to download rather than to generate each time when he needed. For customer support user only need real time data or latest data and old data in database is always for analytical purpose. So, it is better to archive data in data warehouse which reduce the load on DBMS for routine queries.

F. Coding Standards

For large enterprise applications the coding techniques, conventions and standards should be followed. The programmer must use meaningful identifiers, add comments, profile the time consuming areas of code [22]. If developer follows the proper convention and standards then many IDEs provide facility to generate documentation automatically. Following the proper documentation standards can be useful in understanding of the system in future for any other person who may not involve in the process of development. But when he goes through the documentation, he can understand system in lesser time and be able to provide maintenance services or can do modification and improvements in system.

G. Code Re-usability

When we coded for ACS we found similar products and when found, we learned the code and picked the core code which we believed was useful for us in the development of ACS server. Before starting the development from scratch first try to do research for similar or existing systems [21]. As developers of these systems worked on them for so many years so their available source code can be very helpful. Don't reinvent the wheel. Also you can learn lots of new thing from others experiences. This code re-usability can save a lot of times and efforts.

H. Server Tuning

All technologies, used in our product provide concurrency which allow application that maximum utilization of resources. All tools which provide concurrency also provide facility to configure thread for how many concurrent requests should handle. We can increase those tread but increasing of thread is not guaranteed that it will improve performance. Because, there is always an optimum point when we achieve maximum benefit from given resource after that performance will decrease.

Threads are managed by operating system, as much number of threads is increase context switching between them will also increase. Server need to be well tuned under different condition as you can see in Figure 2 that's response time decreases as threads increase on tomacat serve [27]. But at certain point the response time again increase. In figure it is highlighted that how much threads are most optimum for system.

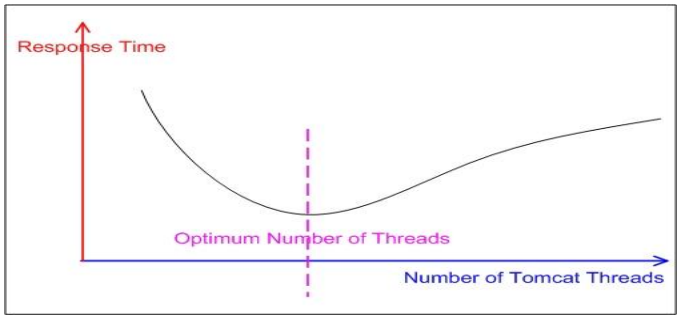

Fig. 2: Transaction response time versus number of threads

I. Deployment and Testing

Before deploying the system the stress testing should be done as the actual environment is different from the development environment. For ACS the most critical test is stress test of ACS under heavy load. So, we needed to generate extensive traffic which should be equal to actual environment. We used jmeter an open-source product and run many instance to generate traffic on server for testing purpose [25]. So, the developer should create a virtual testing environment which should be near to actual environment for monitoring the system performance. Find tools and techniques which are most suitable to do so and consider all possible factors those can be involved in a real time scenario before deployment.

IV. CONCLUSION

In this paper, we have discussed our experiences while developing large software system according to user requirement with the help of existing legacy code and have uses this project as case study for our understanding that what are the challenges that developer has to face to develop such system in which clients are machine and generate huge data and request on server and how to deal with problems during understanding the legacy source code by using software engineering techniques. Although it is difficult to rationalize that which process and technique is best to solve any problem but we believe that our suggestions will provide help to developer to develop similar kind project.

REFERENCES

[1] Butt, Khansa, A. Qadeer, and Abdul Waheed. "MIPS64 user mode emulation: A case study in open source software engineering." Emerging Technologies (ICET), 2011 7th International Conference on. IEEE, 2011.
[2] Mcconnell, S., Open-Source Methodology: Ready for Prime time?, IEEE software, pages. 6-8, July/August 1999.
[3] Debona, C., Ockman, S. & Stone, M., OpenSources: Voices from the Open Source Revolution, published in O'Reilly1999, Sebastopol, CA.
[4] Raymond, E. S, The Cathedral and the Bazaar, published in O'Reilly 1999, Sabastopol, CA
[5] http://en.wikipedia.org/wiki/TR-069
[6] http://www.broadband-forum.org/
[7] Eldering, Charles A. "Customer premises equipment for residential broadband networks." Communications Magazine, IEEE 35.6 (1997): 114-121.
[8] https://github.com/moonranger/clj.tr069
[9] https://github.com/dpavlin/perl-cwmp
[10] http://sourceforge.net/projects/openacs/
[11] https://github.com/andersnorgaard/grailsacs
[12] http://sourceforge.net/projects/jcwmpserver/
[13] http://freeacs-ng.org/
[14] http://sourceforge.net/projects/tr069dlink/
[15] http://tr069.wordpress.com/2012/09/10/open-source-tr-069-efforts/
[16] Zitser, Misha, Richard Lippmann, and Tim Leek. "Testing static analysis tools using exploitable buffer overflows from open source code." ACM SIGSOFT Software Engineering Notes. Vol. 29. No. 6. ACM, 2004.
[17] Eaddy, Marc, et al. "Cerberus: Tracing requirements to source code using information retrieval, dynamic analysis, and program analysis." Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on. IEEE, 2008.
[18] Fenske, Wolfram, Thomas Thüm, and Gunter Saake. "A taxonomy of software product line reengineering." Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems. ACM, 2014.
[19] Vajk, Tamas, et al. "Denormalizing data into schema-free databases." Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on. IEEE, 2013.
[20] Sanders, G. Lawrence, and Seungkyoon Shin. "Denormalization effects on performance of RDBMS." System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on. IEEE, 2001.
[21] Mahmood, Ahmad Kamil, and Alan Oxley. "An evolutionary study of reusability in Open Source Software." Computer & Information Science (ICCIS), 2012 International Conference on. Vol. 2. IEEE, 2012.
[22] Smit, Michael, et al. "Code convention adherence in evolving software." Software Maintenance (ICSM), 2011 27th IEEE International Conference on. IEEE, 2011.
[23] Satyanarayanan, Mahadev, et al. "Transparent logging as a technique for debugging complex distributed systems." Proceedings of the 5th workshop on ACM SIGOPS European workshop: Models and paradigms for distributed systems structuring. ACM, 1992.
[24] Rachidi, Houda, and Ahmed Karmouch. "A framework for self-configuring devices using TR-069", Multimedia Computing and Systems (ICMCS), 2011 International Conference on. IEEE, 2011.
[25] http://jmeter.apache.org/
[26] Rachidi, Houda. Design and Implementation of a Framework for Self-Configuring Devices Using TR-069. Diss. University of Ottawa, 2011.
[27] https://today.java.net/pub/a/today/2006/06/06/distribute-detach-parallelize-tomcat.html?page=2