DOI: http://dx.doi.org/10.26483/ijarcs.v8i9.5179

ISSN No. 0976-5697

Volume 8, No. 9, November-December 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A CRITICAL ANALYSIS OF PERVASIVE COMPUTING IN EMBEDDED SYSTEMS USING INTERNET OF THINGS

Ankita Sharma
Department of Computer science
JIMS Rohini Sec 5
Delhi, India

Manav Bali
Software Developer
DION Global Solutions Ltd
Delhi, India

Abstract: As we know Traditional computing is not able to fulfil the growing needs for information requirement anywhere anytime. So Pervasive computing now a day marks the next generation of computing in which technology follows the current users without any particular contribution from their end. Pervasive computing is completely opposite to the desktop paradigm, which makes use of a single user and a single device for a particular purpose, on the other hand one who uses pervasive computing gets into use many computational devices and systems together and may not be aware that they are doing so, (follows the concept of availability and invisibility).

Pervasive computing follows three basic properties which are interaction, coordination, cooperation of embedded computing devices. These sensors have embedded software's that provide crucial information about location of the people and devices available for interaction. This paper focuses on the basic concepts of pervasive computing and pervasive computing in embedded systems using internet of things.

Keywords: Pervasive computing, embedded systems, software, LAN, WAN, IoT

I. INTRODUCTION

Pervasive computing has come a long way back in 1970's when the computers started coming closer to its users. Information technology has come a long way where desktop computing is replaced by mobile computing devices that have the capacity to interweave itself into everyday life. In 1991, Mark Weiser, stated that "the most profound technologies are those that disappear". He said, "they weave themselves into fabric of everyday life until they are in distinguishable from it" [1]. Computing has also seen a growth from the era of mainframe computing which was all about one computer being used by many users to the era of desktop computing era in which the usage became so much that is gets frustrating at times. This is where pervasive computing come into picture when distributed computing is followed by mobile computing

Architecture of Pervasive Computing

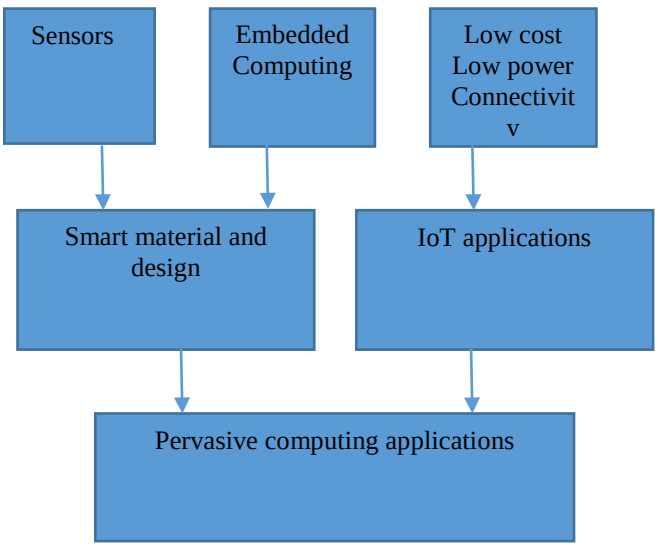

Fig 1: Architecture of Pervasive computing [5]

II. PERVASIVE AND EMBEDDED SOFTWARE

The effect of Moore's law has made the computing system to become cheaper and can be made use in bulk. The Embedded technology is the process of introducing computing power to various appliances. These devices have small sensors that' are intended to perform certain specific jobs and processors giving the computing power are designed in an application oriented way. These computing devices are generally known as embedded systems, because they are embedded into larger technologies, instead of being only sold as a computer. Unlike desktop computers, they basically are for a single user and are customized for the same [2]. They rarely have a keyboard, hard disk, or display instead they make use of internet for communication.

III. PERVASIVE COMPUTING AND INTERNET OF THINGS

The internet of things (IoT) has evolved from pervasive computing. Some say there is really no difference or very little difference between pervasive computing and IoT. Like pervasive computing IoT connected devices communicate and provide notifications about the usage. The internet of things is on the way to provide vision and turning common objects into connected devices. Pervasive Computing basically works with the idea of having an environment that is both perceptual and invisible and proves beneficial for the users given anytime and anywhere. The concept of Pervasive computing evolved since 1970 when electronic machines (computers) came closer to its users [4]. The involvement of information technology into our lives has already taken mobile computing technology into the next level of computing where information is made available to the users anywhere and anytime. His technology was coined as the software that was found "EVERYWARE".

IV. EMBEDDED SOFTWARE

Like any desktop system, embedded system needs software. These software's are highly difficult to make because of the growing demands in pervasive computing. Instead, software for embedded microprocessors is approaching the complexity of many desktop computing applications, both in the number of lines of code, and the functions being performed. For example, we're asking software to control complex machinery, monitor multiple processes, establish and maintain both voice and data communications on mobile connections, and perform precise computations for guidance and navigation [3].

Software has a unique role to play in the design and use of embedded technologies. The functions of even the simplest systems have multiplied over the last decade, as users' needs and processing power have increased. New and far more complex generations of electronic devices require tens or even hundreds of thousands of lines of software to deliver on their potential. Yet writing this software is much more difficult than writing desktop application software. Embedded device programmers have significant challenges not faced by the average desktop programmer. Because embedded devices lack a full range of interaction with the programmer, software can't be developed on the target device itself. It usually can't even be developed on the same microprocessor and operating system used on the device.

V. THE IMPORTACE OF RELIABLE SOFTWARE IN PERVASIVE COMPUTING

The failure of the software in terms of reliability and performance has proved to be quite serious on the embedded systems. A desktop computer might undergo a software error that may crash the application or the computer. But on the other hand, the consequences on the embedded systems is much more serious, especially when the devices make use of the internet for communication. Also because of the internet, the methods for creating embedded software are no longer good enough. The demand for new, good quality software is becoming high. This leads the industry to hire employees who are less experienced and talented for a simple reason to get the software written. The Experienced and talented programmers have to work in a smarter way to become very productive.

To a great limit these issues are been faced in the development of the software for a number of years. There are number of software technologies like class libraries, user interface builders, and code generators that help to improve the delivery of the complex applications for desktop computers and servers. There's still too much software to write, for too many devices in the era of pervasive computing.

VI. OVERCOMING DEVELOPMENT OBSTACLES

One of the most well-established technology that provides reliable applications is O*BJECT ORIENTED SOFTWARE*. This established technology promises more reliable applications as object-oriented software. The Object-oriented software provides better methods for modularization of the program data that is to be abstracted and also the software components that are to be used. All these features of the object-oriented software make it easy to write and maintain a program because it breaks down the difficult and complex code into small and understandable pieces. The software programmers now have to remember lesser number of concepts and relationships to write better software's.

Also, there is another method to improve the reliability that is known as computer aided software engineering or CASE. CASE is a generic term for software whose goal is to improve the processes behind writing software programs. The CASE software packages have different parts of the software creation process which includes analysis and designing [8].

There have been several limitations with CASE applications: -

- The process for the development of a software is really a complex process and most of the CASE tools have only been able to take hold of the front end of the process Also as we apply the CASE tools we make that portion of the project more reliable and more productive for the developers, but it does not translates into a significant improvement for a complete project because it fails to address the back end that is the implementation and testing part.

- Also, the CASE tools are often used as a standardized format for storing and exchanging of data. This obviously means that if we make an attempt to combine the commercial tools into complete development tools we would require spending an enormous amount of time and energy in data translation and integration. Also, the worst part could be that the tools could simply exchange information and we had to manually re-enter the data.
- The process of software development is so complex that many CASE tools have to pose artificial restrictions on the process or the technical requirements to create a working automation framework. The restrictions that are put on the architecture used by the developers, often puts a limitation to the designing and implementation. Also because of these restrictions many developers do not prefer working with these constraints.

VII. AN ALTERNATE APPROACH

Next comes an alternative to the CASE that is UML (Unified Modelling Language). The UML combines a number of different diagramming techniques to provide a consistent definition of the software system. There are a number of diagrams such as State charts, Sequence diagrams and object modelling diagrams that help the developers to model the different aspects such as architecture of the model or either the flow of data [6].

The most important and unique characteristic of UML is that applying it correctly results in an unambiguous model of a software system. This obviously makes use of the software libraries to automatically produce the source code from the set of UML diagrams. Also, a number of libraries are used to create the diagrams that are used for the implementation of the design and this designing becomes the actual software implementation.

Also, the concept is very powerful in building a software because incomplete designs just represent a huge amount of software errors. Also, the ability to check the correctness and completion of the design and then implementing the software directly from the designing part has the ability to make the software very much reliable. This complete process helps in removing the time-consuming stage of having the hand-coding and designing which makes it possible to deliver the final code very quickly [7].

During the development process, the engineers can also generate the code directly from the diagrams. The engineers make the changes to the code and reflect them back into the model. The model, represented by the UML diagrams, and the codes stay consistent throughout the life of the software**.**

VIII. CONCLUSION

The promise of ubiquitous computing is to make the life smart thus creating a sensor network capable of collecting, processing, and sending data and finally communicating as a means to adapt the data activity in the essence of network and its surroundings and which further leads to enhance human experience and quality of life, assisted by computers. The idealistic visions painted by the ubiquitous computing movement stand in stark contrast to what we see when we boot up our computers each day. There is an immediate barrier because you have to know how to use a computer to use a computer. For example, if you sit in front of computer without knowing how to use a mouse we will not be able to do anything. It's unlikely. The computer won't help you, either, since you have to know how to use the computer to ask it for help on how to use it! When computers do offer assistance, it still tends to fall short of the mark.

IX. REFERENCES

[1] Bakre, A, Badrinath, B.R., Handoff and System Support for Indirect TCP/IP. In Proceedings of the Second Usenix Symposium on Mobile & Location-Independent Computing pp 11-24,April 1995

[2] Kistler, J.J, Satyanarayanan, M., Disconnected Operation in the Coda File System. ACM Transactions on Computer Systems 10(1), pp 3-25, February 1992,

[3] Brewer, E.A, Katz, R.H., Chawathe, Y., Gribble, S.D, Hodes, T, Nguyen, G, Stemm, M, Henderson, T., Amir, E, Balakrishnan, H., Fox, A., Padmanabhan, V.N., Seshan, S., A Network Architecture for Heterogeneous Mobile Computing. IEEE Personal Communications 5(5), pp 8-24, October 1998.

[4] Want, R., Hopper, A., Falcao, V., Gibbons, J, The Active Badge Location System. ACM Transactions on Information Systems 10(1), pp 91-102 January 1992.

[5] Debashis Saha and Amitava Mukherjee, Pervasive computing: A Paradigm for 21st century, Published by the IEEE Computer Society 36(3), pp- 25-31March 2003, 0018-9162.

[6] Lynch, N.A., Morgan Kaufmann, Distributed Algorithms. 1993

[7] Mummert, L.B., Ebling, M.R., Satyanarayanan, M. Exploiting Weak Connectivity for Mobile File Access. In Proceedings of the 15th ACM Symposium on Operating Systems Principles, 29(5) pp-143-155, December 1995.

[8] Weiser, M., The Computer for the Twenty- First Century [J].Scientific American, 1991(3)pp- 94-100.