American Journal of Operations Management and Information Systems
2017; 2(3): 72-75
http://www.sciencepublishinggroup.com/j/ajomis
doi: 10.11648/j.ajomis.20170203.11

Review Article

Review of Software Engineering in the Context of Embedded and Cyber Physical Systems

Omuya Odhiambo Erick[1], Julius Murumba[2]

[1]Department of Computing and Information Technology, Machakos University, Machakos, Kenya

[2]Department of Management Science, Technical University of Kenya, Nairobi, Kenya

Email address:

Omuya2005@gmail.com (O. O. Erick), j.murumba@gmail.com (J. Murumba)

To cite this article:

Omuya Odhiambo Erick, Julius Murumba. Review of Software Engineering in the Context of Embedded and Cyber Physical Systems. *American Journal of Operations Management and Information Systems.* Vol. 2, No. 3, 2017, pp. 72-75. doi: 10.11648/j.ajomis.20170203.11

Received: October 28, 2016; **Accepted**: January 4, 2017; **Published**: January 27, 2017

Abstract: Embedded systems have overwhelmingly penetrated systems globally in areas such as transportation, industrial-automation, medical-equipment, communication and energy as a result of Innovations being triggered by software embedded in these systems. These systems use approximately 98 percent of all the microprocessors produced worldwide. The objective of this study was to discuss the state of embedded systems use in software engineering, establish Opportunities Created by Embedded Systems and to investigate the Challenges of Embedded and Cyber-Physical Systems. This study utilized the literature review method to examine and analyze secondary sources of data such as conference reports, journal articles, and publication articles including google scholar. The paper aims at contributing towards knowledge and lessons that can be applied in towards building embedded and cyber physical systems in software engineering.

Keywords: Embedded Systems, Real-Time, Layout, Latencies, Concurrency, Petri-Nets, Cyber-Physical, Feedback Loops

1. Introduction

1.1. Embedded Systems Overview

In the current world, it is difficult to imagine a day-to-day life without embedded systems. We are surrounded with many embedded systems products and our lives depend on the proper working of these gadgets. Embedded systems have been greatly applied worldwide in many areas like transport, industrial automation and communication systems. A good percentage of microprocessors produced globally have actually been used in such systems [8-9]. The worldwide market for embedded systems is around 160 billion euros, with an annual growth of 9 percent. While these statistics are comparable to the world's biggest software packages, such as Microsoft Windows, embedded software is far more complex due to the real-time and interface constraints that do not affect IT, application, or desktop software. The embedded and information systems communities tend to exist in almost complete isolation from one another. This holds for conferences as well as for organization layout and products. Embedded systems most often need real time programming. Real Time operating systems and their working are generally shown by two methods: Finite state machine and Petri-Nets. But these two modeling methods have shown certain limitations as many sophisticated embedded systems are multiprocessor systems and the processes have short latencies.

1.2. Cyber-Physical Systems Overview

Cyber-Physical Systems (CPS) are integrations of computation with physical processes.

Integrated networking, information processing, sensing and actuation capabilities allow physical devices to operate in changing environments. This makes smart systems possible but also creates the need for a new 'systems science' that can lead to unprecedented capabilities. Tightly coupled cyber and physical systems that exhibit this level of integrated intelligence are sometimes referred to as cyber-physical systems. All CPS have computational processes that interact with physical components. These can be relatively simple (e.g., a heater, cutting machine) or comprise multiple components in complex assemblies (e.g., vehicles, aircraft systems, oil refineries). The computational and physical processes of such systems are tightly interconnected and

coordinated to work together effectively, often with humans in the loop [4].

Robots, intelligent buildings, implantable medical devices, cars that drive themselves or planes that automatically fly in a controlled airspace—these are all examples of CPS. Today, CPS can be found in such diverse industries as aerospace, automotive, energy, healthcare, manufacturing, infrastructure, consumer electronics, and communications. Everyday life is becoming increasingly dependent on these systems—in some cases with dramatic improvements. Software engineering aspects in relation to cyber-physical systems is similar to embedded systems. An example is given below.

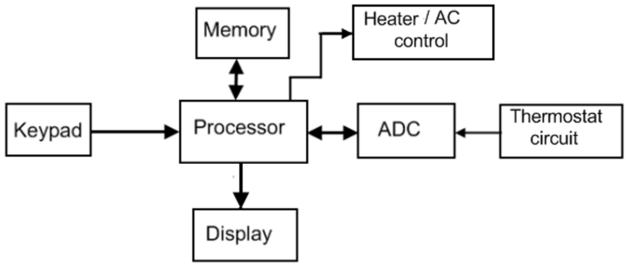

Figure 1. *Embedded system example –Digital thermostat.*

2. State of Embedded Systems in Software Engineering

2.1. Lines of Code vs Making Things Work

In embedded software the complexity is not in the lines of code, most of the times. Configuring an interrupt handler to respond to your button push and balancing the priority between a button push and a temperature sensor input might have taken a whole day to work on. At the end of the day, the embedded software developer would have written 50 lines of code. A software developer generally writes more lines of code almost always. The reason is that a software developers builds a product just with his lines of code, out of thin air [1]. An embedded guy makes a physical hardware device work with his software**.**

2.2. Algorithm & Data Processing vs System Control

A software program generally revolves around 2 aspects, ALGORITHM and DATA. Take any program, it would either be computing something (numerically or logically), which is what i refer as Algorithm or it would be working on data. It could be storing data, moving data, processing data or simply presenting or deleting data. Any software right from banking, insurance, retail, and logistics to simple PC based software like Word, PowerPoint, etc., all of them work on the aforementioned principles. An embedded software is more focused towards controlling and managing the system (or hardware). It is developed to exploit the full potential of the hardware and manage it for the benefit of the user. Though there would be data and algorithm in embedded software, it would be there only to control and manage the hardware in a better fashion.

2.3. Personal Computer vs Printed Circuit Board

Though embedded developers work on PC they are not writing software for PCs. They use the PC to build their software which eventually runs on another platform (a Printed Circuit Board with a Micro controller). Embedded software engineers develop software for these BOARDS and move the executable binary from the PC to the board using debugging tools or specific connectivity options. Software developers develop software that run on PCs or PC equivalents (like servers). Whether the software runs in a bank, shipyard, your Fedex store, Airport or Grocery store it runs on a computer. May be nowadays they run on Mobile phones and tablets too. From an embedded perspective today's tablets and mobiles don't differ much from PCs as they are all are more "General Purpose" [5-6].

2.4. Improving Software Development for Embedded Systems

To improve your own embedded development processes, and to ensure that benchmark data applies to your environment, we strongly suggest building your own history database with baselines for estimation and quality planning. To get started without much overhead, we recommend the following lean set of effective project indicators [1].

a. Effort. This is a basic monitoring parameter to ensure you stay on budget. Effort is estimated up front for the project and its activities. Afterward, these effort elements are tracked.

b. Schedule and time. Monitor results, increments, and milestones to ensure that you can keep the scheduled delivery time. Similar to effort, time is broken down into increments or phases that are tracked based on what has been delivered so far. Note that milestone completion must be aligned with defined quality criteria to avoid detecting poor quality software too late.

c. Project progress. This is the key measurement during the entire project execution. Progress has many facets and should monitor deliverables and how they contribute to achieving the project's goals. Typically, there are milestones for the big steps and earned value and increments for the day-to-day operational tracking. Earned value techniques look to the degree with which results such as implemented and tested requirements or closed-work packages relate to effort spent and elapsed time. This lets us estimate the cost and remaining time to complete the project.

d. Methodology: This is the method that is used to actually develop the embedded system. It should be monitored in terms of its progress and accuracy of delivery. One that has been repeatedly used is agile methodology which yields early product delivery, predictable schedule and relatively good quality [10].

3. Opportunities Created by Embedded Systems

Embedded systems are becoming more and more important. The exponential increase in computing power, ubiquitous connectivity and the convergence of technology have resulted in hardware/software systems being embedded in everyday products and places (for example: today, 20% of the value of each car is attributed to embedded electronics, and this will increase to 35-50% by 2020). It goes without saying that embedded systems engineers have excellent career prospects, due to the increasing integration of hardware and software in applications. You'll build your own future based on your personal interests, whether you'd like to be a researcher, a designer (developer of new products and services) or an organizer (managing an engineering department or providing consultancy services).

4. Challenges of Embedded and Cyber-Physical Systems

4.1. Challenges of Embedded Systems and Proposed Solutions

4.1.1. Creating Predictable Systems

Creating systems whose behavior can be predicted is one of the universal bottlenecks of embedded systems. System behavior includes a number of parameters like functionality, reaction and execution properties, for instance, timing and consumption of resources. Prediction of process outcome includes consideration of all possible continuations which is a difficult and expensive task to perform. In building predictable embedded systems the developers can opt to entirely use deterministic parts as the major components of the systems. The main task therefore would be to determine all causes of non-determinism and introduce ways of sorting them [11-12].

4.1.2. Robustness Through Continuity

Another major challenge pf designing embedded systems is creating systems with robust behavior even when perturbing conditions exist. Given a set of requirements, a preference metric provides a measure of how close a system comes to meeting the requirements, and how robust it is against small changes in the requirements [13-14]. These requirements can either be reactive or execution oriented and they may assume dimensions like how precise the result is, timeliness and the expected lifetime of the system. The properties of robustness can be formalized as a mathematical continuity. Continuity properties are great because they enable system performance to degrade smoothly if the environment changes either accidentally or maliciously [12].

4.1.3. Safety and Security

Risks from malfunctions of embedded software are much higher than those of application software. Security rapidly grows in relevance as embedded software communicates autonomously with other computing systems [2].

4.2. Challenges of Cyber-Physical Systems

Advancement in CPS requires a new systems science that encompasses both physical and computational aspects. Systems and computer science has provided a solid foundation for spectacular progress in engineering and information technology; a type of new systems science is now needed to address the unique scientific and technical challenges of CPS. Below are some of the challenges facing cyber-physical systems.

a. Interaction between humans and systems. Current models for human and machine behaviors are not adequate for designing CPS when humans and machines closely interact. One of the challenges is modeling and measuring situational awareness—human perception of the system and its environment and changes in parameters that are critical to decision-making. This is particularly necessary for complex, dynamic systems, such as those used in aviation, air traffic control, power plant operations, military command and control, and emergency services.

b. Dealing with uncertainty. Complex CPS need to be able to evolve and operate reliably in new and uncertain environments. An increasing number of these systems will also demonstrate emergent and unknown behaviors as they become more and more reliant on machine learning methodologies. In both cases, uncertainty in the knowledge or outcome of a process will require new ways to quantify uncertainty during the CPS design and development stages. Current methods for characterization and quantification of uncertainty are limited and inadequate. This is exacerbated by the limits of reliability and accuracy of physical components, the validity of models characterizing them, network connections, and potential design errors in software.

c. Measuring and verifying system performance. The difficulty of verifying performance, accuracy, reliability, security, and various other requirements impedes development and investment in CPS. Today's capabilities for verification and validation (V&V) of CPS are limited, time consuming, and costly, particularly when compared to development time. Two major challenges are the creation of methodologies to further the capabilities of V&V of complex systems, and the development of test beds and datasets to support a principled approach to the validation of complex CPS. If the design phase is more reliable, testing can become more informed and require less time. The evaluation challenges will become increasingly difficult at the larger scales and higher complexity expected for future CPS, which will have massive and interconnected sensor, actuator, and component networks.

5. Conclusion

This paper discussed the state of embedded systems in software, established the opportunities created by embedded and cyber physical systems. The challenges related to the development if these systems are also well articulated. The

content was obtained by examining and analyzing secondary sources of data. The research contributed to knowledge and provided lessons that can be applied in building embedded and cyber physical systems.

References

[1] Christof Abert and Copers Jones, 2009. Embedded Software: Facts, Figures and Future, IEEE, 2009.

[2] GAO, 2010. Protecting the Federal Government's Information Systems and the Nation's Cyber Critical Infrastructures. Government Accountability Office, 2010. Accessed 12/18/12. http://www.gao.gov/highrisk/risks/ safety-security/government_information_systems.php

[3] IEEE Software, special issue on software development for embedded systems, May/June 2009; www.computer.org/portal/site/software

[4] Lee, 2012. Edward A. Lee, Cyber Physical Systems: Design Challenges, International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), May 6, 2008, Orlando, FL.

[5] Newsletter and archive on embedded-software engineering and technologies: www.embedded.com

[6] Applications of Embedded Systems: http://www.amazon.com/Software-Engineering-Embedded-Systems-Applications/dp/012415 9176

[7] Differences between Embedded and Cyber-Physical Systems: https://www.linkedin.com/pulse/5-differences-between-embedded-maharajan

[8] David Greenfield, 2013, How Embedded Systems are changing automation. Automation World, 2013.

[9] P R Kolhe, M. H. Tharkar, R. M Dharskar P P. Kolhe, 2014. Impact of Embeded Systems in Modern Life, International Journal of Computer Science, Volume 2, Issue 10. 2014.

[10] James Grenning, 2011. Agile Embedded Software Development, San Jose, CA, 2011.

[11] Edwards, S. A. & Lee, E. A. 2007. The case for the precision-timed PRET machine. In *Proc. Design Automation Conference* (*DAC*), pp. 264–265. (doi: 10.1109/DAC.2007.375165)

[12] Thomas A. Henzinger. 2008. Two challenges in embedded systems design, The Royal Society publishing, 2008.

[13] Chatterjee, K., Ghosal, A., Henzinger, T. A., Iercan, D., Kirsch, C. M., Pinello, C. & Sangiovanni-Vincentelli, A. 2008 Logical reliability of interacting real-time tasks. In *Proc. Design, Automation, and Test in Europe* (*DATE*), pp. 909–914. (doi: 10.1109/DATE.2008.4484790)

[14] R. Alur et al. 2001. "Hierarchical Hybrid Modeling of Embedded Systems," Proc. 1st Int'l Workshop Embedded Software (EMSOFT 01), LNCS 2211, Springer, 2001, pp. 14–31.

[15] Embeded Systems. http://www.intel.com/education/highered/Embedded/lec-tures/.