Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 68 (2014) 22108-22112

ISSN: 2229-712X

Challenges evolved in building a complex software system

M.Shanmugaraj
Department of Computer Science & Information Systems, University of Limerick,Limerick, Ireland.

ARTICLE INFO

Article history:
Received: 13September 2013;
Received in revised form:
22 February 2014;
Accepted: 1 March 2014;

Keywords
Challenges,
Complex Systems,
Software Engineering.

ABSTRACT
New advancement in the field of Software Engineering increases more issues and problems. After reviewing quite a few research papers & articles in the field of Software Engineering, we obtained various challenges that rise in building a complex software system. This paper describes several challenges evolved in building a complex software system.

Introduction

Software is everywhere. It makes us wakeup, makes a call, connectivity with friends, fly in space and lot more. A typical cellphone comprise 2 million software codes; by 2010 it is expected around 10 times as much as now [36]. A typical software system consists of computer programs, software, configuration files, documentation, associated programs and etc... Software can empower or even accelerate human, social, economic and technological changes [4]. There are many different types of meaning associated with software depending on the context. The areas where the software systems used are database management systems, telecommunication networks, military command and control systems, air traffic control systems, operating systems, computer reservation systems, online shopping, expert systems and etc…

Software System

A software system is a system of intercommunicating components based on software forming part of a computer system. Software system consists of few number of separate programs, configuration files, which are used to set up these programs, system documentation, which describes the structure of the system, and user documentation, which explains how to use the system [33]. Generally software is considered as normal programs/instructions/commands which are stored in the device of a computer. These programs allow the user to perform certain specific task to be performed. These programs are often written by software engineers. Software can be categories into system software, application software and embedded software. System software is the basic software needed for a computer to operate.

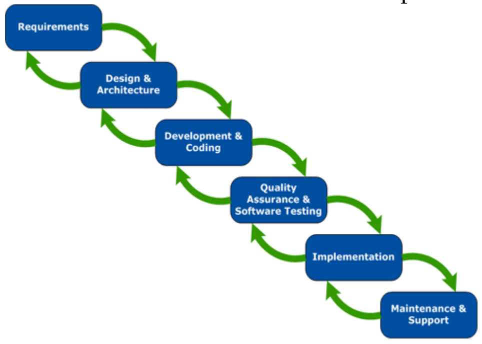

Figure 1: Software Development Process [35]

Understanding the problem

Generally it is difficult to completely understand how software works. Because of software failure, we lose billions of money every year. Even a small error can cause wastage of billions. For example, in 2004, U.S. government spent around $60 billion on software; a 5 percent failure rate means $3 billion was probably wasted. There are many factors which contribute to the failure of the project namely:

- unrealistic or unarticulated project goals,
- inaccurate estimates of needed resources,
- badly defined system requirements,
- poor reporting of the project’s status,
- unmanaged risks,
- poor communication among customers, developers, and users,
- use of immature technology,
- inability to handle the projects complexity,
- sloppy development practices,
- poor project management,
- stakeholder politics,
- Commercial pressures [36].

To make the software system autonomic the key properties of a system should be self-configuring, self-healing, self-protecting and self-optimizing with attribute properties such as self-aware, environment-aware, self-monitoring and self-adjusting [6]. This is how NASA’s mission expects the next generation software systems should be built on. Fig 2 explains the various NASA’s mission towards autonomic system nature. This is very crucial in terms of operation cost & resource constraints.

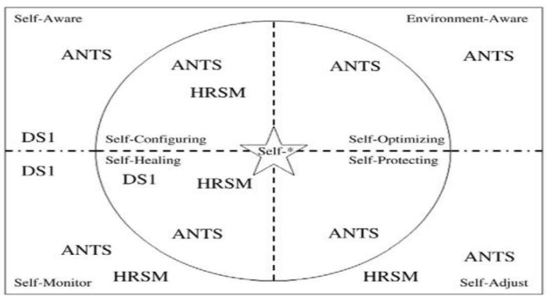

Figure 2: Evolution of Self properties in NASA's mission [6]

Tele:
E-mail addresses: innoraj@gmail.com

Most challenges for software system arise from automated or autonomous systems [12]. There are various operations and functionalities to be performed by software systems in space. Those operations and functionalities should be integrated into the system without any constraints. This is considered to be one of the greatest challenges for people involved in the mission. Software integration is another part involved in software evolution. When two companies merge to produce software, the company has to consider several factors such as architecture, platform, technology used, reuse of the existing software, upgrading system to the new requirement specified. The process flow in such case could start with evaluation, design, decision, analysis, user involvement, early meetings, separating stakeholders, active upper management, architecture centric process and managing different people [13].

Challenges evolved

With software engineering, several challenges evolve in a number of different ways throughout the development of the software. Certain challenges could be addressed [28], while others are really complex and questionable. Challenges could exist in development methods & models, software vs. hardware, development of the environment, unpredicted behavior of the system, lack of proper communication, rules & regulations, software maintenance, software engineering education. These are a list of few which obtained from research articles & papers. They are challenging in various domains such as power systems [15], software intensive systems [16, 22], data intensive system [24], SPRUCE [20], embedded & networked aerospace software systems [18], mobile software system [21], chemical industry, automotive system [19], Control System Engineering (CSE) [32], Bioinformatics [31], MPSoC [30], self-adaptive software [26], space exploration [29], safety critical systems [25], aviation industry and so on.

Development Methods & Models

Generally software development involves various activities such as requirement gathering, analyzing the requirements, planning, implementation, testing, documenting, deployment, maintenance and etc… There are several development models such as

- waterfall model,
- spiral model,
- iterative and incremental development,
- agile development,
- rapid application development
- code & fix.

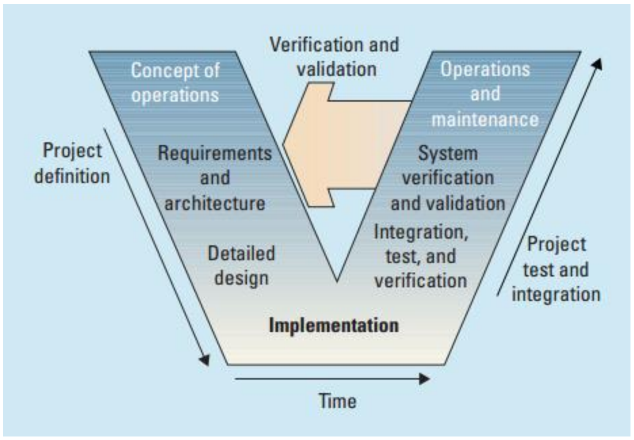
Verification and validation
Concept of operations
Operations and maintenance
Project definition
Requirements and architecture
System verification and validation
Integration, test, and verification
Detailed design
Project test and integration
Implementation
Time

Figure 3: Common NASA Software Development Process [3]

Each model follows different approach of developing software. It is based on the team to select the appropriate model for the project. Software system developed for aircraft is so complex which contains a lot of software components and associated programs. It is very important that the software developed for aircraft should be perfect enough without any error & fault. The software development is concerned with the maintaining and extending existing software systems [8]. Software engineers find it easier to build from the existing software system rather than starting from the new one. For example [8] describes how various different features for the TV could be added based on the user requirement using engineering approach.

Few lines of codes could be added to solve various problems. Figure 3. Describes the software development process follow in NASA. This common process applies to all the software development carried out in the entire mission for space exploration. Software development process generally starts from requirement phase until verification & validation phase.

Software Vs Hardware

Without hardware, software is nothing. It is very important that we need to know the working of hardware to ensure the correct operation of the software. Hardware is normally a physical device connected to computer, but it contributes in many ways such as infrastructure, data centers, storage, servers, networking equipment's, cables & so on. Initial software engineers have to consider various hardware limitations such as memory & disk space. Today software engineers are provided with highly sophisticated hardware. They find it really difficult to understand and development software with the increasing functionality of the hardware. So a large gap exists between software possibility and hardware capability.

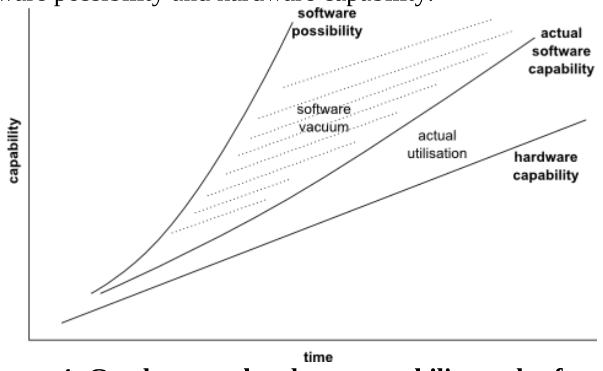

Figure 4: Gap between hardware capability and software possibility [37]

This gap should be reduced so that the software development could be matched with hardware. The fast evolution of hardware gives the programmers not only the performance improvement but also slower reaction of development tools and new hardware functionalities [7]. These new features make programmers & developers difficult.

Developing Environment

Software development is generally considered to be a complex process. In today's world software is developed round the clock people located in different parts of the world. There exists lot of complexities such as geographical distance, temporal distance, cultural difference, language & communication barriers and etc… Even though with the great communication technologies such as internet, certain complexities are unavoidable. The software engineers are expected to work in a global environment. They have to use email, international communication via skype, video conferencing and other social media. In developing a huge software, some part of the software are outsourced to remote organizations.

Software engineers are expected to collaborate with different type of people from different culture & team. There rise various issues such as global distance, temporal distance and language constraints. It is important for engineers, practitioners, developers and educators to understand the issues with developing environment of the software. Requirements vary with system to system. One of the important problems faced during the development of e-scape software system [11] was different requirement and priorities of the user. This made to think whether separate the user into different system or to integrate into hierarchy of users [11].

CLARAty [5] is robotic software used in the NASA mission. It is difficult to build reusable software due to various factors such as platform, information rate, architecture of the system, device & system configuration and different application program. The major challenge faced in CLARAty reusable robotic software where control heterogeneous robots, integrate and interoperate new capabilities, adjusts access levels and implement a generic framework [5]. Since there is no specific standard for robotic platforms, normal framework could be sufficient to develop robotic system. They face several issues such as physical variability, interface, sensor configurations, robotic algorithms, hardware/software framework, architectural mismatch, different forms of similar information, integrating different technologies. In 1994, White House provided executive order to agencies directing them to increase the use of commercially available software systems. Changing from traditional development to Commercial Off The Shelf (COTS) development imposed several challenges to manufacturers and integrators [14]. The engineering process should be tailored to regulations and rules defined by the Department of Defense (DOD).

Unpredicted Behavior of the System

Software system built for critical real time systems are particularly difficult and challenging. They depend on several factors such as engineering & management, appropriate tools and environment and developer. Several testing methods are there to make sure the developed software system operates safely in a given environment. The software developed for the avionics require lot of efforts to make it successful. The avionic software system developed for Saturn V flight software system consist of eight programs, including the software operating system, which consist of one half million 32 bit words of data and executable instructions [2]. This software system is considered to be the crucial part to the shuttle operations. They perform various operations such as guidance, navigation, flight control functions performed during all flight phases, gathering data from the environment, sensor input, issuing of commands to the vehicle/ground interface functions [2]. Apart from these operations the software system takes care of management and monitoring of onboard systems, fault detection, annunciation, preflight, pre-entry checkout and safing procedures [2]. The unpredicted behavior of the system could be determined using various testing methods. Testing depends on the type of software it is developed for. High quality software system needs tested with various extreme conditions to make sure the software operates in the proper manner. Model based testing [9] is one the most common approach used to test the software system.

Lack of Proper Communication

One of the common challenges which are faced by the software community is because due to the erroneous communication or miscommunication. Communication failure could be possible at any point of development which might rise during any of the following requirement phase, documentation phase, development phase, implementation phase, maintenance phase & etc. If the requirements provided are incomplete & not clearly specified, the team would develop software which might be inappropriate. Developing space systems usually involve various factors:

Multicomponent systems, elements of AI, autonomous systems, evolving systems, high-risk and high cost systems, rigid design constraints, potential for extremely tight design space and highly risk driven systems [3]. If any of the factors is not correctly built the system would ultimately result in disaster. Proper communication plays a major part from building requirements to the final software system. Most of the space development system team involves various professionals and people. Proper communication media should be provided to ensure the information is communicated effectively and in time bound.

Regulations and Law

Various critical software involved in military, space exploration, defense, nuclear weapons, chemical industries and etc. should strictly follow several rules and regulations imposed by local government. Safety critical system tend to have reliability requirements ranging from 10^{-5} to 10^{-9} over a given time period [1]. Federal Aviation Authority (FAA) rules require that any failure condition that could be catastrophic must be extremely improbable [1]. Certain software systems are identified as critical systems by various standards and criteria by the government.

Software Maintenance

In today's world maintenance of software is considered to be one of the importance challenges for most of the software professionals. Around 50 to 70 percent of a software engineer's time is spent making changes to mission critical software [17]. Some of the problems associated with software maintenance are: poor system design and structure, excessive system complexity, limited system flexibility, limited or nonexistent documentation, inadequate project and process management, inadequate change and version management, inadequate release management and inadequate maintenance tools [17]. Implementing Knowledge Management Systems in software engineering introduces various challenges such as: Software Engineering is a vast domain, convince software engineers to use KMS, impacts are difficult to measure, KMS integration, technology related knowledge obsolescence and KMS should support software processes [34].

Software Engineering Education

Software engineering institution plays a key role by producing effective software professionals who could take-up job after the studies. The software engineering educators should be well qualified & trained. So that they could identify and train the students towards the global need. It is a great challenge for teaching community people to meet the personal preference and technical complexities. As educators how do we teach students to build security into their software and to implement software consistent with privacy policies? [10] The current education trend totally differs in commitment, interaction, learning style, and these create new challenges for education and usage of the technology [10]. Each and every day new software emerges with sophisticated functionalities & features making people learn more easily without training.

Apart from these challenges [23] provides 13 challenges with respect to software engineering namely: software quality, return on investment, process improvement, metrics and measurement, standards confusion, standards interoperability, legacy software, testing stoppage criteria, interoperability and compos ability, operational profiles, designing in, product certification and services.

Conclusion

Thus with the increasing software development, the challenges tend to be higher & complex. These complexities and challenges should be considered by the software professionals. We tried our best to describe the various complexities involved in developing software.

Abbreviations

FAA	- Federal Aviation Authority
NASA	- National Aeronautics and Space Administration
ACM	- Association for Computing Machinery
IEEE	- Institute of Electrical and Electronics Engineers
HRSM	- Hubble Space Telescope Robotic Servicing Mission
ANTS	- Autonomous Nano Technology Swarm
DS1	- Deep Space 1
CSE	- Control System Engineering
DOD	- Department Of Defense
COTS	- Commercial Off The Shelf
MPSoC	- Multiprocessor Systems-on-Chip
SPRUCE	- Software Producibility Collaboration and Experimentation Environment

References

Nancy G. Leveson, “Software Safety: Why, What, and How”, ACM Computing Surveys, Vol.18, No.2, June 1986, Page No. 126-163.

William A. Madden and Kyle Y. Rone, “Design, Development, Integration: Space shuttle primary flight software system”, Communication of the ACM, Sep 1984 Vol 24 No.9, Page No.914 – 925

Emil Vassev et al., “Swarm Technology at NASA: Building Resilient Systems”, Published in IEEE Computer Society, IT Pro Mar/Apr 2012, ISSN No.1520-9202, Page No. 36-42

Tom Mens et al., “Software Evolution”, Published by the IEEE Computer Society 0740-7459/10, Page No.22-25

Issa A.D. Nesnas et al., “CLARAty: Challenges and Steps Toward Reusable Robotic Software”, International Journal of Advanced Robotic Systems, Vol.3, No.1 (2006), ISSN 1729-8806, pp. 023-030.

Roy Sterritt et al., “Next generation system and software architectures challenges from future NASA exploration missions”, Science of Computer Programming 61 (2006) pp. 48-57, DOI: 10.1016/j.scico.2005.11.005.

Yuzhong Sun et al., “Green challenges to system software in data centers”, Front. Comput. Sci. China 2011, 5(3): 353-368, DOI: 10.1007/s11704-011-0369-3.

Thein Than Tun et al., “Specifying features of an evolving software system”, Softw. Pract. Exper. 2009;39:973-1002. Published online 8 May 2009 in Wiley Inter Science, DOI: 10.1002/spe.923.

Alexandre Petrenko et al., “Model based testing of software and systems: recent advances and challenges”, International Journal of software tools technology transfer (2012) 14:383-386, DOI: 10.1007/s10009-012-0240-3.

Hossein Saiedian, “Software engineering challenges of the Net generation”, Published in The Journal of Systems and Software, 82 (2009) 551-552.

Karim Derrick, “Developing the e-scape software system”, International Journal of Technology Des Educ (2012) 22:171-185, DOI: 10.1007/s10798-011-9193-1.

David J. Atkinson, “Constellation Program Return to the Moon: Software Systems Challenges”, Proceedings of the third IEEE International Workshop of Engineering of Autonomic & Autonomous Sytems(EASE 2006), 0-7695-2544-X/06.

Rikard Land et al., “Integration of Software Systems”, Proceedings of the 29th EUROMICRO Conference “New Waves in System Architecture”(EUROMICRO 2003), 1089-6503/03.

C. Stephen Kuehl, “A process direction for common avionics developments using commercial hardware and software components: the avionics systems engineering challenge”, 0-7803-4150-3/97, 1997, pp. 6.4.1-6.4.9.

H.S. Gill et al., “Software Engineering in Power Systems: Practices and Challenges”, 0-7803-7107-0/01, 2001, pp. 25-30.

Manfred Broy, “The ‘Grand Challenge’ in Informatics: Engineering Software-Intensive Systems”, Published in the IEEE Computer Society, ISSN: 0018-9162/06/2006,Computer, vol.39, no.10, pp. 72-80.

David Sharon, “Meeting the challenge of Software Maintenace”, ISSN: 0740-7459/96, pp. 122-125.

David C. Sharp et al., “Challenges and Solutions for Embedded and Networked Aerospace Software Systems”, Proceedings of the IEEE, ISSN: 0018-9219, pp. 621-634.

Jurgen Mossinger, “Software in Automotive Systems”, Published by the IEEE Computer Society, ISSN: 0740-7459/10, pp no.92-94.

Patrick Lardieri et al., “SPRUCE: A Web Portal for the Collaboration Engineering of Software Intensive Systems Producibility Challenges Problems and Solutions”, ISBN: 978-1-4244-8/09, pp No. 276-283.

V.Rahimian, J.Habibi, “Performance Evaluation of Mobile Software Systems: Challenges for a Software Engineer”, Proceedings of 5th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE 2008), ISBN: 978-1-4244-2499-3/08, pp No. 346-351.

Mats P.E. Heimdahl, “Safety and Software Intensice Systems: Challenges Old and New”, Proceedings of Future of Software Engineering(FOSE ‘07), ISBN: 0-7695-2829-5/07.

Jeffrey Voas, “A Bakers’s Dozen: 13 Software Engineering Challenges”, IT Pro Mar/Apr 2007, ISSN: 1520-9202/07, pp No. 48-53.

Anthony Cleve et al., “Data Intensive System Evolution”, Published by the IEEE Computer Society, ISSN: 0018-9262/10, pp. 110-112.

John C. Knight, “Safety Critical Systems: Challenges and Directions”, Proceedings of ICSE ’02, ACM 1-58113-472-X/02, pp no. 547-550.

Mazeiar Salehie, Ladan Tahvildari, “Self-Adaptive Software: Landscape and Research Challenges”, ACM Transactions on Autonomous and Adaptive Systems, Vol.4, No.2, Article 14, DOI 10.1145/1516533.1516538.

Martin et al., “Software-engineering Challenges of building and deploying reusable problem solvers”, Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2009), 23, 339-356, doi: 10.1017/S0890060409990047.

Joe W. Heil, “Addressing the Challenges of Software Growth and Rapidly Evolving Software Technologies”, American Society of Naual Engineers, DOI: 10.1111/j.1559-3584.2010, pp no.45-58.

Robyn Lutz, “Software Engineering for Space Exploration”, Published in the IEEE Computer Society, ISSN: 0018-9162/11, pp no.41-46.

Ian Gray, Neil C.Audsley, “Challenges in Software Development for Multicore System-on-Chip Development”, ISBN: 978-1-4673-2789-3/12, pp no. 115-121.

Jonathan Barker, Janet Thornton, “Software engineering challenges in bioinformatics”, Proceedings of the 26th International Conference on Software Engineering (ICSE ‘04), ISSN:0270-5257/04.

Matt M. Eskandar and Vahid Garousi, "Engineering Control Software Systems: A Multi-Disciplinary Challenge", ISBN: 978-1-4673-0750-5/12, pp no.1-6.

Sommerville Ian, "Software Engineering", Addison Wesley, pp 5-6, ISBN: 978-0-321-31379-9.

Mostefai Mohammed Amine and Mohamed Ahmed-Nacer, "Implementing Knowledge Management Systems in Software Engineering: Opportunities and Challenges", Proceedings of 36th IEEE International Conference on Computer Software and Applications, ISSN:-730-3157/12, DOI:10.1109/COMPSAC.2012.10.

http://www.managedmayhem.com/2009/05/06/sashimi-waterfall-software-development-process/

http://spectrum.ieee.org/computing/software/why-software-fails/0

http://www.gophoto.it/view.php?i=http://newspaint.files.wordpress.com/2012/05/softwarevhardware.png#.UZDft8o1eSo