Towards Software Engineering Process for C4I Systems

Tazar Hussain, Khalid Naseer Al-Mutib, Abdullah Sharaf Alghamdi
Department of Software Engineering
College of Computer and Information
Sciences, King Saud University,
P.O. Box 51178 Riyadh 11543 Saudi Arabia
Phone: +966 1 4676988
{tazhussain, muteb, ghamdi}@ksu.edu.sa

Abstract— **Command Control, Communication Computer and Intelligence (C4I) systems enables modern military forces to achieve information superiority in the battlefield. C4I are complex System of systems (SOS) where individual systems interact locally to achieve global SOS behaviors. To build software for C4I systems conventional software engineering SwE process and practices have shortcomings and are not capable to support certain aspect of these systems. If C4I systems fail to operate as required due to the fact that SwE process was unable to fulfill its requirements, the consequences may not be tolerated because of the criticality of the mission of these systems in information warfare (IW). This paper highlights the distinguished characteristics and operational requirements of C4I systems which poses challenges to SwE process and practices. This paper also discuss the possible future research areas in order to enhance SwE process so that better software could drive these complex systems as required.**

Keywords- System of systems (SOS), C4I systems, Software Engineering (SwE) process, interoperability, NetCentric

I. INTRODUCTION

Software engineering (SwE) process, methods and models most often concentrate on software systems which operate in standalone fashion or have few interactions with other systems. The requirements of today's software systems in areas ranging from transport to defense cannot be met by standalone, monolithic system. Instead the required functionality of SOS can only be achieved by arrangement of systems that result when independent, heterogeneous and useful systems are integrated into larger system that delivers unique capabilities.

Information Warfare (IW) seeks to achieve information superiority by affecting adversary information, information-based processes, and information systems while defending one's own information, processes and systems [1]. Command, control, communication, computer and intelligence (C4I) system is a complex, geographical dispersed information systems that provide situational awareness about the location and status of friendly and enemy forces [2]. C4I provides the relevant information, superior decision making ability and thus enables the commander to achieve information superiority in the battlefield. To achieve information superiority C4I systems exploits the power of modern information and communication technology infrastructure. Earlier C4I systems were design independently with specific purpose but today the military operation are conducted in net centric environment with high interoperability. Modern weapon systems such as Future Combat System (FCS) or army battle command system (ABCS) poses new challenges for system and software engineers. C4I as SOS uses software which drives the information and communication technology but crucial point is that traditional software development processes and methods are not fully capable of supporting C4I systems. To develop software for C4I systems, software development process faces issues such as C4I systems have complex business logic and takes many forms [3], also mission needs of C4I applications evolves and user expects an ability to adapt their system accordingly.

From software engineering perspective major problems and issues arise because of certain principal characteristics of SOS that distinguishes it from other systems. In this paper we explore the SOS challenges associated with software engineering practices, processes and acquisition process and the limitation of these processes and techniques.

II. CHARICTERISTICS OF C4I SYSTMS

C4I systems are complex and complicated therefore different architecture frameworks have been develop e.g. Department of defense architecture framework DODAF [4], Ministry of defense architecture framework MODAF [5] and NATO architecture framework NAF [6]. The main purpose of these architecture frameworks is to provide guidelines and break complexity. C4I systems are difficult to be managed in terms of software/system engineering because of its unique requirements and challenges [7]. There are four types of C4I SOS Virtual, Acknowledged, Directed and Collaborative [8]. In this paper we concentrate on acknowledged SOS which has recognized objective, designated manager and resources.

As the operational requirements of C4I systems changes rapidly and new systems needs to be integrated. For example US Army Battle Command system (ABCS) is a digital battlefield environment and is made up of various Command and control C2 systems. But the operational requirement of ABCS has changed and now eleven new systems need to be integrated to this system [7]. To engineer software for these kind constituent systems is big challenge in term of design, requirements and implementation.

Following are the key features of C4I systems of systems which harden and limit the process of system and software engineering.

978-1-61284-486-2/11/$26.00 ©2011 IEEE

A. Operational Independence

If the system-of-systems is disassembled into its component systems the component systems must be able to usefully operate independently. The C4I systems composed of components which are independent and useful in their own right.

B. Managerial Independence

SOS component operate independently without central management in force. The component systems consist of independent legacy systems to achieve SOS end to end objectives.

C. Evolutionary Development

The behavior and functionality of system-of-systems is not predictable. Its development and changes occurs evolutionary with functions and purposes added, removed, and modified dynamically.

D. Emergent Behavior

C4I system performs operations such that the required capability can only be achieved when individual systems interoperate. These behaviors are emergent properties of the entire system-of-systems and cannot be localized to any component system. The principal purposes of the systems-of-systems are fulfilled by these behaviors. The figure 1 shows how local systems interact to achieve goals at global level. Different systems process information locally but these systems also interact with other systems and the capability of SOS is greater than the sum of constituent systems. By understanding the behavior and functions of individual systems the global behavior cannot be predicted.

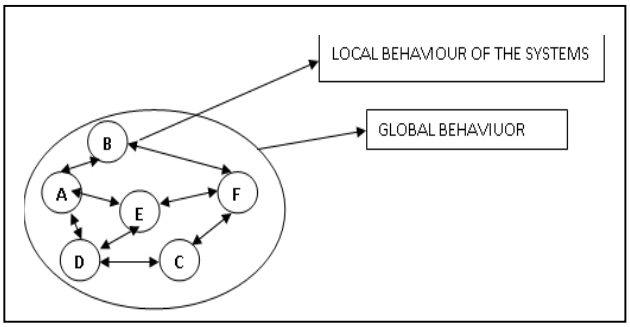

Figure 1. Local systems interacts to achieve global objectives.

The fact that the constituent systems exchange data may give rise to dynamic behavior, software may fail if these requirements have not been designed in.

E. Geographic Distribution

Systems of systems are geographically dispersed connected through network and communication technology. Distribution and communication increase complexity and uncertainty of SOS.

F. Network Centricity

Network centricity also known as net centricity is an emerging theory of war that seeks to translate an information advantage into a competitive war fighting advantage through the robust networking of well informed, geographically-dispersed forces allowing new forms of war fighting [9]. The term networking in Net centric warfare (NCW) means the network connection between people and systems in information and cognitive domain [10] to share information they need, when they need, it also protect information from those who should not have it. Ideally NCW is suppose to provide seamless communication internetworks composed of tactical radios, satellites, shooters, sensors, wireless and landline links to share information as it required. Technically this kind of setup offers many challenges to design and develop software systems for the constituents system which is interacting with other system. In NCW all the systems interact dynamically therefore actual participating components are not known until runtime. There is no central control over process or functionality because all components are selected dynamically.

G. Interoperability

Interoperability is the ability of systems, units and forces to provide services and accept services from other systems and operate effectively together both at operation and technical level [11]. C4I systems and services must interoperate to share information in reliable manner that is operationally useful. To implement the interoperability in C4I software is very difficult because of the unknown interfaces, legacy and diverse systems [12, 13]. In C4I system the data relies on dispersed heterogeneous systems different mechanisms are required such as mechanism for issuing requests, mechanism to describe data, mechanism for performing data format translation, mechanism for performing joins, Rules and a protocol for issuing requests and receiving responses in a structured manner [14].

III. SOFTWARE ENGINEERING ISSUES IN C4I SYSTEMS

Software engineering is the application of a systematic, disciplined approach to the development, operation, and maintenance of high quality and affordable software. SwE process must have to fix bugs and vulnerabilities of C4I components e.g. jet aircrafts, wireless sensor networks (WSNs) shooters tanks, manned and unmanned aerial vehicles. Any vulnerability in weapon systems software if exploited successfully may result in complete disaster e.g. a denial of service attack on anti missile system. The components of C4I weapon systems can only achieve the mission objective if the software code is free of bugs, vulnerabilities and meet the operational requirements. For the reason that future of C4I is dependent on reliable software engineering principles defense industry promote collaboration with software engineering institutes [15].

Following section identifies some core software issues which are specific to C4I systems and other complex systems; we also tried to identify the limitation of the current software engineering methodology and practices.

1. In SOS context the constituent system (system under construction) that would be part of a larger system has independent management and funding authorities, this greatly affects the planning and requirements phases because all the related requirement cannot be gathered specially at initial stage.
2. While gathering the requirements for development the team will have not only to document the requirements for the constituent system but also have to understand the interfaces, code structure of other systems that interoperates with constituent system.

3. The evolution towards system of system approach increase the number and diversity of stakeholders, in SOS stakeholders have different background, interests and priorities that creates serious technical and management conflicts. Non SOS systems have clear stakeholders but in SOS context there are stakeholders for both constituent systems and SOS level and often the stakeholders for SOS has little knowledge about constituent system. For example to achieve the overall goal of SOS system A may require to access system B but this in this process system A may be a security threat for system B. Similarly in service oriented architecture services are shared may have conflicting requirements. Similarly stakeholders also may be disagree on what should be the important e.g. for system engineer functionality is important that creates security hole.
4. Emergent behavior is a behavior of C4I SOS that cannot be predicted by understanding constituent systems in isolation. It poses the challenge that developers cannot design in or incorporate logic, features in constituent software system that can be predicted. These kind of systems needs to be more adaptable in order to respond to changes but if one constituent system respond to changes that might have unintended effect on other constituent system or on SOS. The dynamic and unanticipated nature of C4I systems make them difficult to be upgraded easily because any changes to these systems effect more than one systems and also effects SOS objective. Unlike traditional systems software team must not only evaluate operational reliability and security for constituent system but also have to consider the system under development at SOS level. Software engineer also cannot predict what user demands would emerge when constituent systems interact with other systems because these systems configure dynamically.
5. C4I systems are composed of diverse systems, services, components, technologies and infrastructure therefore integration and configuration management becomes uncertain and complex. Integrating incompatible systems and unpredictable behavior of components are the increasing cause of failure in SOS. Dispersed distributed, interconnected and independent networks increases inconsistencies. Single error could have multiple sources and can affect multiple systems and any changes made to resolve the error may results in new technical and management issues.
6. Software provides mechanisms to make SOS function but decentralization and complexity increase the spectrum of failure because complex systems break in complex ways. Thus software requirement process must incorporate strategies to analyze and evaluate potential risks throughout the life cycle. The strategy must capture the major requirement of SOS, complexity and must devised planes and mechanism to avoid conflict.
7. Risk analysis is the key to any software as part of requirement analysis but complexity and evolving nature of the SOS limit to identify risks and incorporate mitigation in advance. In addition, the use of COTS, legacy systems, and independently developed and managed systems limits full system understanding [16]. In SOS development team responsible for constituent system normally have knowledge only about their system but may have little or no knowledge of other systems.
8. In SOS context the full requirement of constituent system cannot be gathered because system will interoperate with independent COTS or legacy system already in operation. SOS development thus provides limited freedom of development because all the relevant requirement and risks cannot be anticipated.
9. To meet the complex requirements of war fighting C4I systems uses commercial of the shelf (COTS) software, civilian communication and network systems. Legacy systems are the integral part of C4I systems and to replace these systems is not an easy task. COTS and legacy systems contain software bugs and vulnerabilities because these systems were not developed as part of C4I systems. In order to develop robust software for C4I weapon systems legacy systems and cots limit the SE process in many ways e.g. to interoperate COTS and legacy systems transfers and consume data from other secure systems but these systems can also inject the inherit vulnerabilities and software bugs.
10. In conventional systems boundaries and interfaces are known while in C4I based systems the boundaries and interfaces may not be known and will make it difficult to design software for these systems. C4I systems include diverse, incompatible, critical and geographically dispersed components which also limits the full understanding of systems requirement.

IV. REASEARCH AREAS TO ENHANCE SOFTWARE ENGINEERING

The challenges identified in earlier section of this paper make the task of software engineers hard to develop software for C4I systems to meet the expectations. This section discusses and suggests the option that the software engineering community has to enhance and produce better software for complex C4I systems. In Advance software engineering (ASER) lab at King Saud University we expect

to work on the following areas in to overcome the shortcomings of software engineering.

A. Modeling and Simulation

Modeling emerges one of the useful tools because nonlinear systems like C4I are difficult to predict but these systems can be simulated through models and results can be obtained from simulation. However simulation may not evaluate all aspects of C4I systems and may fail when comes under certain environmental stress. Modeling and simulation (M&S) have some drawbacks but still can be proved very useful tools. In one of our previous effort [17] we worked on how to improve C4I systems in terms of security through threat modeling.

M&S helps understanding how interaction take place at specific time, how vulnerabilities arise as result of interaction, it also helps in identifying possible failure points. M&S are also essential for understanding runtime dynamic behaviors, information sharing, and resource limitation of constituent system as it operate in SOS context. Simulating network centric capabilities would help in identifying what software can handle before implementation. M&S also facilitate the communications and shows relationships of different stakeholders by emulating systems interactions among its components. Simulation plays its prime role in requirements analysis as alternate solutions can be suggested and also handle change and evolution process. A discrete simulation model and other areas of software engineering where M&S can be applied have been proposed in [18].

Assurance cases are the useful modeling tools that plays crucial role in software assurance but it can also be applied in C4I based systems. Assurance case can ensure that the constituent systems get develop in the context of global objective and assurance case can also enable different stakeholders to see systems from their own perspective so that conflicts may be eliminated.

B. Open Based System Approachs

In order to understand C4I systems and its relation to the environment open based approaches from biology, Social sciences and other natural inspired techniques are useful. As we human have the capacity of self organizing and as one of the best example of system of systems where different local systems interacts to achieve larger objectives.

One example of this approach is that of IBM initiative towards autonomic computing such systems will have the ability to self organize and self repair [19].

C. Software survivability and resilient

The equations are an exception to the prescribed C4I are the backbone systems for information superiority which enables the forces to make superior decision, failure of these systems or failure of some of it components may be having disastrous affect on the mission objectives. It is impossible to make them free of all defects but survivability engineering and resilience nature can be embedded in software systems so that the critical parts continue to operate despite failure and defects. Already research has been started in this direction [20] [21] with the prime purpose that highly distributed and unpredictable systems provide essential services and resists to software failures.

D. Netlogo

NetLogo is a programmable modeling environment for simulating complex systems authored by Uri Wilensky in 1999 and has been in continuous development ever since at the Center for Connected Learning and Computer-Based Modeling [22]. This tool has mainly been used in biology and social sciences but due to some of its fantastic feature it could be a very useful in modeling and simulating C4I based complex systems. Thousand of independent entities can be modeled and instructed through netlogo which make it possible to explore the interactions among micro level pattern behavior of constituents system and macro level patterns that emerges from the interaction of many component systems. In this way netlogo could produce useful results as this tool enables to test different individual systems under different conditions. Emergence and interoperability and systems can be easily simulated through netlogo and will enable the designer of C4I systems to understand these systems under different scenario.

V. CONCLUSION

Modern C4I systems and application are more emergent, unpredictable, dispersed, complex and distributed and thus cannot be built through conventional software engineering process and methodologies. In this paper we tried to identify some core issues and hurdles which make the work of software engineering difficult and challenging.

This paper brings some challenging issues in front with the hope that it will enable us to analyze specific problems and improve software practices, process, techniques and methodologies. This work in this paper is just the beginning and in future we aim to work continuously on the future research topics such as modeling, simulation and netlogo to recommend best practices for software engineering.

REFERENCES

[1] P. William Secretary of Defense. C4I surveillance and reconnaissance. DOD Annual Report to the president and congress, http://www.dod.mil/execsec/adr96/chapt_27.html last accessed November 28th 2009.

[2] Committee to reviw DOD C4I planes and Programme “Realizing the Potential of C4I, Fundamental Challenges” available at http://www.nap.edu/openbook.php?record_id=6457&page=28.

[3] W,lee “ Why is C4I software hard to develop”, 12TH ICCRTS “Adapting C2 to the 21st Century.

[4] Dapartment of defense architecture framework DODAF V 2.0, August 2010 available at http://cio-nii.defense.gov/sites/dodaf20/.

[5] UK Ministry of defence architecture framework MODAF revision version 1.2, september 2008 available at http://www.mod.org.uk.

[6] Nato architecture framework NAF version 3.0 avaialble at http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf.

[7] Director, Systems and Software Engineering, Deputy Under Secretary of Defense (Acquisition and Technology) Office of the Under Secretary of Defense, "Systems Engineering Guide for Systems of Systems version 1.0" 2008.

[8] M.Mark "Architecting Principles for Systems-of-Systems" available from http://www.infoed.com/Open/PAPERS/systems.htm.

[9] C. Amine, D. James “The Implications of Network-Centric Software Systems on Software Architecture: A Critical Evaluation”, Proceedings of the 45th annual ACM southeast regional conference, 2007.

[10] R. Scott “Building information system for network centric warfare”, the MITRE Corporation, June 2003.

[11] Department of Defense Directive 5000.1, "Defense Acquisition," 15 March 1996; available from http://www.acq.osd.mil/ar/doc/dodd5000-1.pdf; Internet; accessed 15 February 2010.

[12] C. Anthony A. B. Arleigh " The Future Combat System What Future Can the Army Afford?" Febraury 2009.

[13] C. Radu and K. Marta “ Software Engineering techniques for the development of systems of systems” computing laboratory, university of oxford.

[14] M. Madiajagan, and B. Vijayakumar " Interoperability in Component Based Software Development", World Academy of Science, Engineering and Technology 22 2006 www.waset.org/journals/waset/v22/v22-13.pdf.

[15] R. Sandy, The Redstone rocket "Software engineering is core for future weapon systems" October 2002 available from http://www.redstone.army.mil/history/bios/dodgen_photos/Pages%20from%2010_02_02.pdf.

[16] C.Rita and E.Bob "System-of-Systems Influences on Acquisition Strategy Development" https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/acquisition/981-BSI.html#dsy981-BSI_maier1998.

[17] A. Ghamidi, H.Tazar, K. Gulfaraz, “Enhancing C4I security using threat modeling”, 12th International IEEE Conference on computer modeling and simulation, March 2010.

[18] C. Alan “Simulation: An Enabling Technology in Software Engineering” available at www.sei.cmu.edu/.../simulation-enabling-technology-sw-engineering.pdf.

[19] J. O. Kephart and D. M. Chess. The Vision of Autonomic Computing. Computer, 36(1):41–50, January 2003.

[20] E. Robert at el, Survivability Assurance for System of Systems, May 2008 available from http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA482312.

[21] Protecting your critical systems, CERT, Oct 2005 available at http://www.cert.org/archive/html/protect-critical-systems.html.

[22] Netlogo "multi-agent programmable modeling environment" http://ccl.northwestern.edu/netlogo/docs/.