Journal of Naval Science and Engineering
2009, Vol. 5 , No.3, pp. 104-116

CHALLENGES OF WEAPON SYSTEMS SOFTWARE DEVELOPMENT

Ph.D. Kadir Alpaslan Demir, Lt.
Department of Computer Engineering
Turkish Naval Academy
Tuzla, Istanbul, TURKEY
kademir@dho.edu.tr

Abstract

This paper describes some of the challenges faced today in developing weapon systems software. The weapon systems software context brings unique challenges to the software development effort. In this paper, we list related publications and briefly talk about major characteristics of weapon systems software. Finally, we discuss some of the challenges of weapon systems software development.

SİLAH SİSTEMLERİ YAZILIMLARININ GELİŞTİRİLMESİNDEKİ GÜÇLÜKLER

Özetçe

Bu makalede, günümüzde silah sistemleri yazılımlarını geliştirirken karşılaşılan güçlüklerden bazılarını incelenmektedir. Silah sistemleri yazılımlarını geliştirmek, yazılım geliştirme sürecinde ve ortamında kendine has güçlükler yaratmaktadır. Bu makalemizde konuyla ilgili yayınları listeleyip silah sistemleri yazılımlarının temel karakteristiklerini sıralamaktayız. Son olarak, silah sistemleri yazılımlarının geliştirilmesinde karşılaşılan güçlükleri kısaca tartışmaktayız.

Keywords: *Weapon Systems, Weapon Systems Software, Development of Weapon Systems Software, Challenges of Weapon Systems Software Development*
Anahtar Kelimeler: *Silah Sistemleri, Silah Sistemleri Yazılımları, Silah Sistemleri Yazılımları Geliştirme, Silah Sistemleri Yazılımları Geliştirme Güçlükleri*

1. INTRODUCTION

A significant portion of weapon systems is safety-critical real-time reactive systems [11]. In today's environment, these systems rely more and more on software-based components [12,13]. Table 1 shows the system functionality requiring software for a typical weapon system, a combat aircraft. The table emphasizes the increasing importance of the software portion in weapon systems development. The Crosstalk–Journal of Defense Software Engineering–published an article entitled "Now More Than Ever, Software Is the Heart of Our Weapons Systems" [13] in its January 2002 issue. The article provides examples of software intensive systems that increase the U.S.A.'s national defense capabilities. Today, the success of a weapon system is becoming more and more dependent on the success of the software portion of the system. Therefore, this paper focuses on the challenges posed by defense software development.

It is important to note that the software product itself is not different whether it is for military or commercial purpose. However, the software development process is not about the software product only. It is also about the manner in which the software is produced. Thus, management aspects have effects on the software development process and on the approaches to technical issues.

The rest of the paper is organized as follows. Section 2 provides a brief list of related publications on the topic. The definition of weapon systems software can be found in section 3 and the characteristics of these systems are explained in the next section. Section 5 lists the challenges and finally we conclude the paper in section 6.

Weapon System	Year	% of Functions Performed in Software
F-4	1960	8
A-7	1964	10
F-111	1970	20
F-15	1975	35
F-16	1982	45
B-2	1990	65
F-22	2000	80

Table 1: System Functionality Requiring Software [Source: USAF "Bold Strike" Executive Software Course, 1992]

2. RELATED PUBLICATIONS ON MILITARY SOFTWARE

The USA's Defense Science Board (DSB) [6] produces the major reports and publications on this issue. The board prepares reports related to military software from time to time. Some of the related reports are "Military Software" (1987), "Acquiring Defense Software Commercially" (1994), "Open Systems" (1998), "Defense Software" (2000). There are also other reports prepared by the science panels of the military services and the National Research Council. Examples include "Adapting Software Development Policies to Modern Technology" [7] (1989), "Report of the AMC Software Task Force" (1989), "Scaling Up: A Research Agenda For Software Engineering" by Computer Science and Technology Board Research Council (1989).

Crosstalk – The Journal of Defense Software Engineering is also another source of information. In every issue, the journal addresses specific issues related to defense software [8].

3. DEFINITION OF WEAPON SYSTEMS SOFTWARE

The term, weapon systems software, is in fact self-explanatory. However, this term is not in common use in the literature. Instead, the terms "Military Software" or "Defense Software" are generally used to encapsulate weapon systems software. The phrase "Defense Software" refers to software developed for a uniformed military service [1]. The term also includes software developed for the U.S. Department of Defense, or the equivalent branches in other countries. Jones [1] provides a broad definition of defense software and the main attribute that distinguishes defense software from other types of software:

The broad definition of defense software includes a number of subclasses such as software associated with weapons systems; with command, control, and communication systems (usually shortened to C3 or C cubed); with logistical applications; and also with software virtually identical to civilian counterparts such as payroll applications, benefits tracking applications, and the like. The main attribute that distinguishes defense software from other types of software is adherence to military or Department of Defense's standards.

As a result, defense software encapsulates weapon systems software. Examples of weapon systems include missiles, torpedoes, artillery systems, combat aircrafts, fire control systems etc.

4. CHARACTERISTICS OF WEAPON SYSTEMS SOFTWARE

Weapon systems are built in-house by government branches or contracted to eligible companies. System acquisition in this context has significant overhead when compared to commercial systems acquisition.

Currently, a significant portion of weapon systems software is safety-critical real-time reactive software. Also, many weapon systems are

complex systems. Following characteristics are found in most weapon systems software:

Real-Time Systems: Real-time systems are concurrent systems with timing constraints [2]. A real-time system generally consists of sensors, actuators, and real-time system core. Weapon systems are real-time systems. For example, a missile consists of sensors that capture signals from targets, actuators in wings that will help to navigate, and a decision-making component acting as the real-time system core. Meeting timing constraints between these components poses significant challenges. Also, real-time systems require real-time control that makes control decisions based on input data without any human intervention [2] or with little intervention. Development of such systems requires significant effort on all aspects of the development.

Safety-Critical Systems: A safety-critical system is a system whose failure may cause injury or death to human beings. Actually, weapon systems are intended to cause destruction on targets. However, the key phrase in a weapon systems context is "to intended targets". Therefore, weapon systems should not cause harm to its own users. For example, a torpedo is incapable of differentiating signals from its target and from its mother ship. Thus, system developers incorporate a feature in the design such as arming after a safe distance. The failure of this feature may cause the torpedo to attack its mother ship. Ensuring the correct implementation of safety-criticality to weapon systems is another challenging part of weapon systems software development. Safety-critical aspects of these systems are also found in some commercial applications such as medical systems.

Mission-Critical Systems: Mission-critical systems are systems whose failure will cause significant loss in terms of money, trust, or defense capabilities of a nation or of a military entity. For example, if a particular weapon does not work in a combat airplane or in a ship, the airplane or the ship will be subject to destruction by the enemy. The development of mission-critical systems poses similar challenges as in the case of real-time systems.

Embedded Systems: Embedded systems are systems that are parts of a larger hardware/software system. These systems often require special purpose hardware entailing increased optimization. The software associated with these systems has to work with this hardware and requires more attention. Signal processing components (hardware, software or firmware) found in weapon systems are examples of such systems.

Interaction with External Environment: Real-time systems typically interact with an external environment that generally does not involve humans [2]. In this environment, the interactions are generally not periodic; therefore timing constraints can be more complex. For example, a combat system may include a close-in weapon system to protect the ship from incoming missiles. This requires monitoring the signals in the environment and initiating the necessary components of various weapon systems.

Reactive Systems: Reactive systems are the systems that respond to external stimuli in a limited time period. Many real-time systems are reactive systems [3]. This is also the case for many weapon systems. These types of systems are event-driven and must respond to external stimuli [2]. In most cases, the system has to keep a history of previous events to determine the corresponding response. For example, in a torpedo system, the number of captured signals from a target within a specific time period may initiate a number of actions. The correct implementation of reactive systems poses different challenges than many other systems due to the unpredictable nature of their environments.

High Quality Systems: Weapon systems must be high quality systems. We do not have well-established quality metrics at the current state of the art. Quality includes attributes such as reliability, performance, fault-tolerance, safety, security, availability, testability, and maintainability [4]. Some of these attributes have metrics, but the software engineering discipline is far from having a widely accepted quality metric. The quality in weapon systems software may be understood via user satisfaction that is mostly after the fact. Generally, weapon systems must be reliable, safe, available, maintainable, and fault-tolerant. In short, they have to incorporate

many quality attributes at the same time. Software Productivity Research Incorporation has been measuring software quality and productivity since 1985. Their findings indicate that defense systems projects rank at the top in software quality [1]. However, defense systems also rank last in terms of software productivity. Developing high quality software is a labor-intensive task and software development best practices should be followed at every step.

5. CHALLENGES OF WEAPON SYSTEMS SOFTWARE DEVELOPMENT

Addressing all the challenges of weapon systems software development is not the purpose of this study. Only a few of the distinguished issues, especially those different from commercial software development, will be addressed. There are inherent challenges due to the type of the software produced. A number of these inherent challenges are also observed in the commercial software developments.

The United States is the major producer and consumer of defense software in the world and by away the leader [1]. The majority of the researches and studies on related issues are also conducted in this country. However, it is important to note that the findings of this study are also applicable to other countries.

Weapon systems software acquisition and development is different than commercial systems acquisition. The main reason behind the difference is that having the government as a customer comes with serious overhead in the process. Jones reports that defense software projects rank last in terms of software productivity [1]. Meeting the Department of Defense standards create a number of extra tasks for developing defense software.

The major challenges of defense systems including weapon systems are due to acquisition policies. In acquisition, the government releases general requirements documentation. Potential contractors develop specifications based on this documentation. Then, the government grants the

projects on a better value bidding policy. After the project is granted, the requirements become inflexible. Even if the contractor wants to change some of the initial requirements, the process is so cumbersome that the changes require significant effort and likely to increase the cost.

Currently, we do not have a software development life cycle model that fits all. It is unlikely that we will have one. Waterfall software development life cycle model is not applicable to all software projects. This model only works well for custom-developed software where requirements should be fixed when the design begins. Most of the software projects still employ a waterfall model [1]. This may be a side effect of the software acquisition policies of the governments.

Waterfall model requires extensive documentation after each phase. This, in fact, works well with the acquisition policy. Documentation is also very important for maintenance. However, the documentation itself requires maintenance just as does the software. Whenever a change occurs in the software, the documentation also needs to be modified as necessary. If the documentation is more than adequate, then modifications create extra cost. How much documentation is enough is an open question. The answer to this question should be researched in the weapon systems software development context.

Having the government as a customer forces the contractors to comply with many standards, regulations, and policies. This, in fact, helps the customers to acquire high quality systems and helps the developers to develop high quality systems. However, compliance drives up the cost and likely to reduce productivity when compared to applications developed in the commercial context. The volume of defense software specifications and other paper documents has been three times larger when compared to documentation in the civilian sector [1]. Also, the software engineering discipline is a fast-emerging discipline. The standards must keep pace with current practices, and this issue is burdensome for government agencies.

The Ada programming language was enforced by the DoD in many projects for nearly two decades. Using Ada as the programming language in military software was a recommendation in the 1987 report [9]. Since then, the use of Ada became a choice in many weapon systems software development. The benefits of using Ada are also recognized by the commercial sector, especially in aviation systems. On the other hand, new programming languages have been introduced such as Java that offers a platform independent development environment. This property made the Java programming language a widely-used and productive environment. However, Ada has always found limited usage, which restricts the number of language experts. The choice of Ada and a limited number of programming languages in weapon systems software development should be discussed due to the newest technologies introduced everyday. Java standard technology was not able to meet the needs of developing mission-critical real-time systems, however there are recent developments. A real-time specification for Java (RTSJ) has been introduced. Using new programming languages may offer productivity gains that defense customers and contractors would like.

Lacking a widely accepted and reliable quality metric for software poses another challenge for weapon systems software development in which the quality of the product is unquestionable crucial. According to a study conducted by the Standish Group [10], only 16% of all IT projects are completed on time and on budget. The study includes a variety of projects from commercial and defense software. Having a successful quality metric will play a key role in increasing the rate of project success. With the current body of knowledge, software engineering discipline is far from creating such a widely-accepted, easily-applicable, and reliable metric.

Most of the weapon systems software development requires longer schedules when compared to similar developments in the commercial world. Just reaching a final decision on military software contracts results in a delay of between 6 to 18 months [1] and this is even before the work starts. When the schedule is long, naturally the number of requirements changes will increase. In a weapon systems context, software will work with

specialized hardware and the hardware technology advances very rapidly. Therefore, even only the changes in hardware technology will likely to necessitate requirements changes in software.

The requirements extraction in the weapon systems software process is particularly harder than many other commercial applications. The main reason behind this challenge is that there are many stakeholders in weapon systems software. The list starts with the government, the service, the standards, and the users. Even some of the weapon systems are developed with the participation of many countries. The F-16 combat airplane and Harpoon missile are examples of such weapons. This is hardly the case for many commercial systems.

Defense software is often more complex than commercial software. This is a consequence of requirements to provide greater functionality and higher reliability than commercial systems [5]. Figure 1 shows code size over complexity for various systems. The development of complex systems is an obvious challenge. Mostly, military software must integrate with many other systems and legacy systems.

Information security is an important issue for governments and government agencies. This issue also affects weapon systems software development. The development of these weapons must occur in a secure environment. In addition, the product must be resistible to malicious attacks. In other words, the weapon systems software must be secure. The development of secure applications has its own challenges and many scientists have been conducting research on this topic for many years.

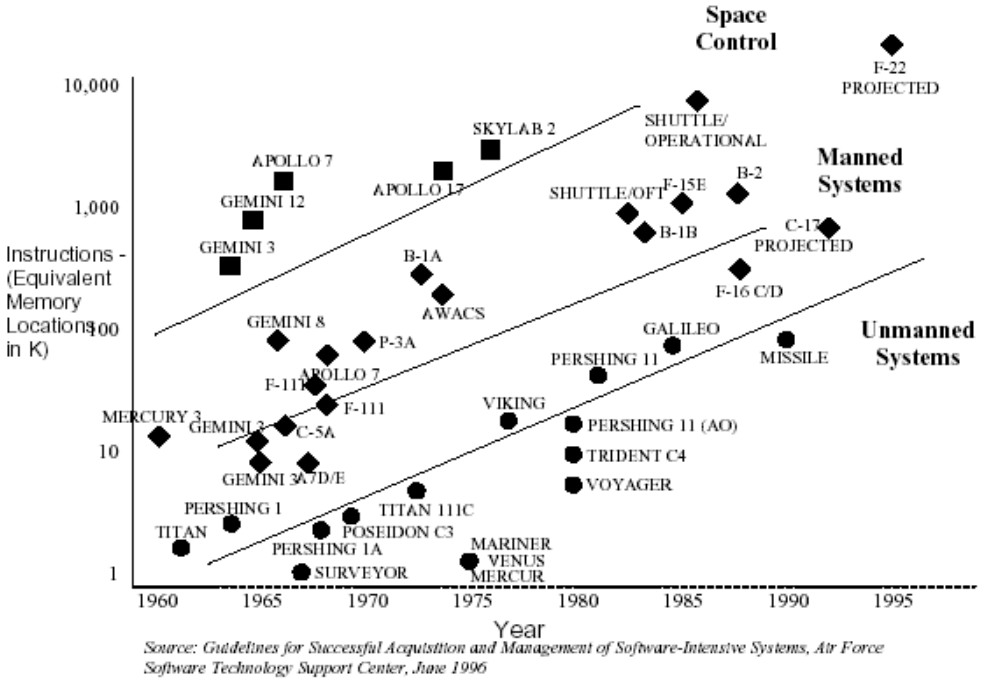

Figure 1. Code Size/Complexity Growth

[Source: Guidelines for Successful Acquisition and Management of Software Intensive Systems, Air Force Software Technology Support Center, June 1996]

6. CONCLUSION

Development of weapon systems software requires significant effort and special attention. The unique environment in which these systems are developed and the properties of these systems bring challenges separating them from most commercial software application development. Understanding the dynamics of this development activity will shed light to solutions of the problems faced. In this study, we try to identify some of the challenges of weapon systems software development. This list is not

exhaustive; however these are among the most important considerations. We believe that addressing these issues will led to more productive weapon systems software developments. In turn, this will help to enhance our war-fighting capability.

Even though most of the issues identified are extracted from the studies conducted in USA, these challenges are not unique to this country. The allied countries follow similar development procedures and standards. For example, U.K.'s Def Stan 07-85 (Design Requirements for Weapons and Associated Systems) refers to MIL-STD 498 (Software Development and Documentation) for most software related issues.

Future research may include identification of other challenges. Also, an in depth analysis of how each issue impacts the development effort may help us to find better solutions as well as increase productivity in weapons systems software development.

REFERENCES

[1] Capers Jones, "Defense Software Development in Evolution," *Crosstalk-The Journal of Defense Software Engineering,* November 2002.

[2] Hassan Gomaa, *Designing Concurrent, Distributed, and Real-Time Applications with UML*, p. 8, Addison-Wesley, 2000.

[3] Harel, D., and M. Politi. *Modeling Reactive Systems with Statecharts.* New York, McGraw Hill, 1998.

[4] J. Voas, W. W. Agresti, "Software Quality from a Behavioral Perspective," *IT Pro,* IEEE Computer Society, August 2004, pp. 46-50.

[5] "Report of the Defense Science Board Task Force on Defense Software," November 2000, Defense Science Board, pp. 11.

[6] Defense Science Board Official Web Site (http://www.acq.osd.mil/dsb/), October 2006.

[7] Beam, W., Chairman, Air Force Studies Board, *Adapting Software Development Policies to Modern Technology*, National Academy Press, Washington, D.C., 1989.

[8] Crosstalk – The Journal of Defense Software Engineering Web Site (http://www.stsc.hill.af.mil/crosstalk/about.html), October 2006.

[9] "Report of the Defense Science Board Task Force on Military Software," September 1987, Defense Science Board.

[10] CHAOS Study, Standish Group, 1999.
[11] Demir, K. A., *Analysis of TLCharts for Weapons Systems Software Development*, Masters' Theses, Naval Postgraduate School, December 2005
[12] Nelson, M., Clark, J., Spurlock, M. A., "Curing the software requirements and cost estimating blues", *PM Magazine,* November-December 1999, pp. 54-60
[13] Spruill, N, "Now more than ever, software is the heart of our weapons systems", *Crosstalk,* January 2002, pp. 3.