

Complex Systems, COTS and Knowledge

Author name: Stephane Collignon
Business Affiliation: Senior Information Technology Officer
Defence Science and Technology Organisation
Address: Electronic Warfare and Radar Division
Post Office Box 1500
Edinburgh
SA 5111, Australia
Phone number and fax: Phone: +61 8 8259 7357
Fax: +61 8 8259 5976
E-mail: stephane.collignon@dsto.defence.gov.au

Author name: Professor Stephen Cook
Business affiliation: DSTO Professor of Systems Engineering
Director, Systems Engineering and Evaluation Centre
Address: University of South Australia
Room F2-24, Building F
Mawson Lakes Campus
Mawson Lakes
SA 5095, Australia
University of South Australia
Mawson Lakes, SA 5095, Australia
Phone number and fax: Phone: +61 8 4321-9876
Fax: +61 8 4321-0000
e-mail: stephen.cook@unisa.edu.au

Abstract

The very high cost of developing software has encouraged the use of Commercial-Off-The-Shelf (COTS) components to realize complex systems. However this approach is not without problems, because the needs of a complex system are often different from the design imperatives of COTS components. Thus, integration of COTS components is problematical.

The first part of the paper will briefly survey the most common problems types encountered with the application of COTS software in huge complex software development, and the reasons for the existence of these problems. The second part will review the solutions currently employed by industry and government, and their efficiency from technical from technical, economic and cultural viewpoints. Subsequently the paper will propose an optimal solution for the use of COTS in the production of complex systems.

Introduction

Aristotle stated: “the whole is greater than the sum of the parts”. This is a primary tenant of the systems movement. Systems comprise components and are characterized by the relationships between the components and the control and communication within these relationships. Systems tend to be hierarchical in nature thus a complex systems comprises components that are also systems but at lower levels of complexity. It is useful to make the distinction between monolithic systems and what is becoming increasingly known systems System-of-Systems (SoS). SoS are characterized by Maier (1999) as systems that:

- Comprise components that are designed as independent entities;
- Display operational and managerial independence of the components;
- Tend to evolve with time through the asynchronous upgrading of components;
- Tend to be very large, e.g. air traffic control, the Internet, military command and control systems.

The functionality of SoS is determined by software and the size and complexity of the software is steadily growing. The cost of developing software is driving the trend to select Commercial-Off-The-Shelf (COTS) products as the core components of SoS.

However, COTS software products, while highly capable, are not ideal building bricks for SoS. Firstly, COTS software has to be thought of as a black box component as there is usually little or no information about the product design or detailed run-time resource requirements (Cook & Collignon, 2002). Secondly, COTS products contain hidden risk associated with the probability of executing either undesired functions or hidden code within the product. This type of unexpected feature is likely to modify the expected behavior of any system component of a complex system.

The first part of the paper will briefly survey the most common problems encountered when COTS software is integrated to form a SoS, and the reasons for their existence of these problems. The second part will review the solutions currently proffered by industry and government, and their efficiency from technical from technical, economic and cultural viewpoints. Subsequently the paper will propose an optimal solution for the use of COTS in the production of complex systems..

SOFTWARE and COTS Problems

a. The roots of the problem

The main problem for defence (and other builders of high-reliability, professional SoS) is that defence is a minority COTS user group (about 0.1% - Ciufu, 2003) and hence the majority of COTS products are not tailored for this market. Increasingly, COTS software is being targeted for the cost-sensitive home market or the equally cost-sensitive small business market. Table 1 illustrates that there is significant non-alignment between the software attributes required for the home market as opposed to the professional market.

Software Attribute	Characteristics of COTS software in a Domestic Environment	Characteristics of COTS software in a Professional Environment	Alignment
Cost	Important	Secondary	Yes
Reliability	Secondary	Vital	No
Support	Restricted to 'How To Use'	Ability to get more than 'How To Use' is Essential	No
Information on Package Internals (I/O, interfaces..)	Rarely available	Essential	No
Availability of Package Functions through APIs	Rarely available	Essential	No
Code Integrity and Trust Level	Irrelevant	Primary	No
Duration of support	Warranty only, at most a few years	Life-Cycle Critical	No

Table 1: Alignment of COTS software characteristics across application environments

In addition, design documentation about COTS products is not released to the public. Similarly, performance figures and testing scenarios are generally not normally available. This situation implies that:

- COTS code cannot be comprehensibly tested;
- Undocumented code is present that may cause erratic behavior at the system level, if unintentionally invoked.

Finally, time to market is the driving force that shapes COTS product development cycles and hence attributes such as integrity, that are so important to professional users, are assigned secondary importance. Rechtin (1991) also highlights that in order to produce a system with a low failure rate, the components from which it is assembled must be ultra reliable.

b. Some mitigation strategies

The challenges presented by using COTS software as components in professional SoS can be partially solved by the following actions:

- Reverse-engineering (disassembling) of a software package to collect adequate information;
- Purchase of extra monitoring software to map the resources used by a software package;

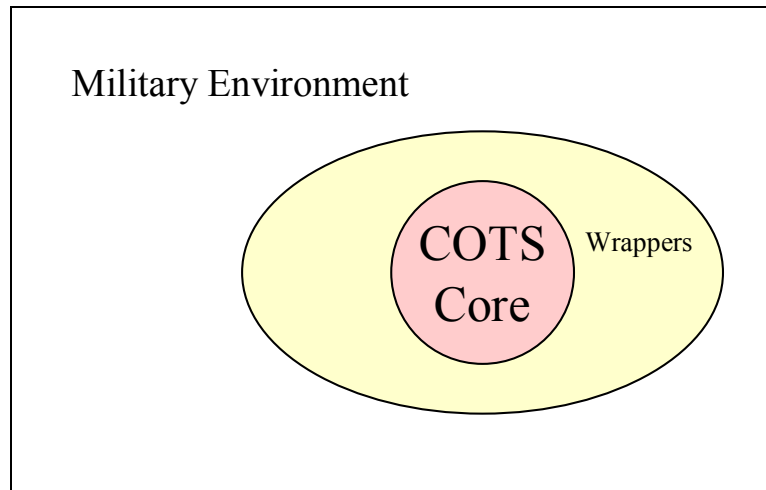


Figure 1: Wrapper Implementation (White, 2000).

- Implementation of wrappers to eliminate out-of-boundary behaviours (White, 2000) as shown in figure 1;
- Intense Test and Evaluation (soaking) of the selected software components to identify the risk areas;
- Addition of glue code (Abts, Boehm and Bailey, 1999) to synchronize the COTS packages with the monitoring tools and wrappers;
- Comprehensive Test and Evaluation of the resulting application code, including glue code, against the system requirements.

This list of actions results in a suite of products, information streams and processes that have to be smoothly integrated together as illustrated in figure 2 where the arrows represent the relationships between the original COTS code, the glue code, the monitoring code, and the host platforms.

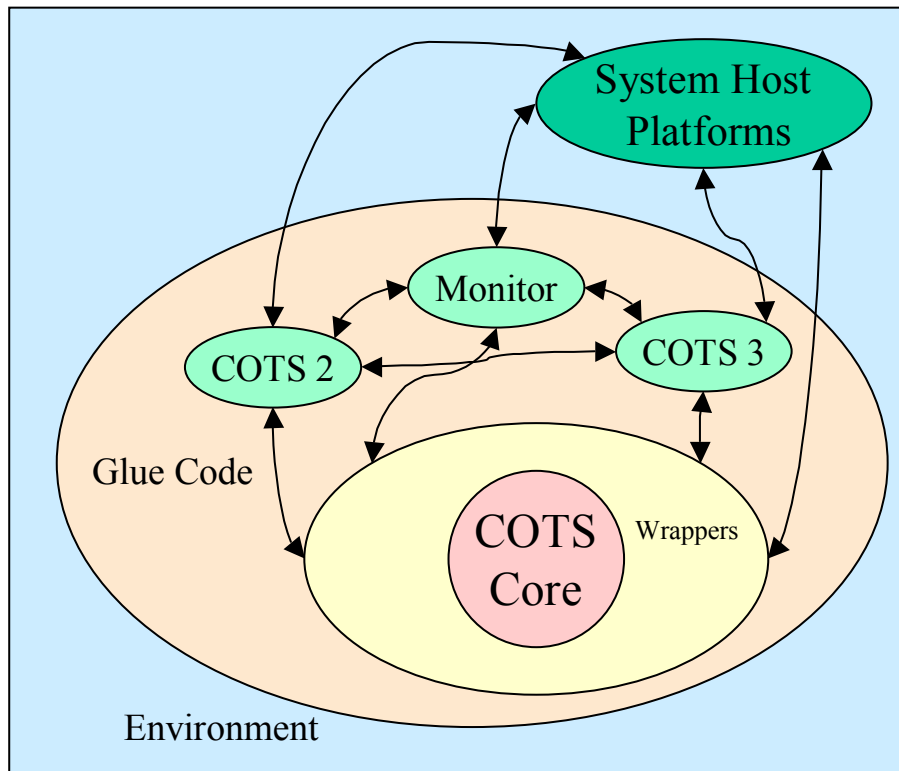


Figure 2: A SoS product synthesised from COTS products.

To implements these strategies, the following skills are needed:

- Deep knowledge of the operating system of the computer host;
- Enhanced trouble-shooting skills on both application and operating system layers;
- Reverse-engineering skills;
- Knowledge of low-level language;
- Software engineering skills, to perform risk evaluation; and
- Advanced software developmental and coding skills to implement wrappers and glue code.

The type of people required will be those who have acquired their skills in a traditional developmental software environment and have not only deep technical knowledge but also an appreciation of systems and software engineering.

Solutions and Behaviours

The analysis of a NATO conference (Collignon & Cook, 2002) gave a distribution of the type of solutions chosen to solve the problems met when developing Defense systems with COTS products. This analysis is introducing a new factor in the problem-solving process. This factor is a cultural one, reflecting the technical environment where the problem occurs on one side, and the resources and political processes involved in problem resolution on the other side. Therefore if we admit the classification resulting from the initial analysis (Collignon & Cook, 2002) we have in Table 2 the following results:

PAPERS	PROPOSED SOLUTION	PROPORTION
1,11	Error Control Implementation	8.3%
15,20,21	COTS product modification	12.5%
2,4,5,6,7,8, 9,10,12,17, 18,19,22,2 3,24	Use of standards, Procedural Evaluation and Selection	62.5%
13,16,	COTS Code Examination	8.3%
1,3	Use of Open Sources Software	8.3%

Table 2: Grouped distribution of the proposed solutions (Collignon&Cook,2002)

This grouping means that that the majority of the COTS users in these examples (79%) decide to adapt their environment to the errors whereas only 13% try to modify the COTS products. Only 9% proposed to use Open Source Software as their development base from which to build their system. Now, let us discuss the solution types:

a. Adaptation of the system environment to accommodate the errors

This choice implies that the system requirements will be adjusted to match the limits of the COTS products capabilities. Three types of action are possible:

- Examination of COTS product source code, to check coding standard level and detect coding mistakes. This solution is resources intensive, in terms of skills, time and complexity. Also copyrights and commercial restrictions normally prevent this inspection from happening. Under extreme pressure due to unsolved problems in their products, the code manufacturers may release some of the COTS sources but this is not a usual situation.
- Use of standards, Procedural Evaluation and Selection.
 - Standards: Examination of the development standards used by COTS manufacturer, against the standards expectations of the customer. Checking of the products compliance with internationally agreed standards (IEEE).

- Procedural Evaluation: Benchmarking exercise of the different COTS code products, using pre-established procedures representing the operational scenarios.
- Selection: Analysis of the results collected during the previous evaluation step, followed by a recommendation to select a particular COTS product. This recommendation normally shows a coefficient of compliance of the COTS product to the original system requirements, and the list of the requirements that COTS products will not meet. This situation will consequently lead to
 - Modification of the user environment by dropping some of the system original requirements or
 - Addition of glue-code to the COTS products to provide additional functionality.
- Error Control Implementation. Implementation of wrappers, filters, glue-code to force the faulty COTS product to return function status codes within expected system and application boundary values.

b. COTS product modification.

This step is only possible when communication is well established between the COTS producer and customer. COTS products are modified after consultation with the customer, to fit the customer 's requirements. This solution is normally available when the COTS manufacturers wish to acquire a part of a market they do not presently have. The cost of such an exercise is commercially high. For the user it means the opportunity to have normally large-scale COTS products (Operating Systems, Networks, specialized applications) scaled to their requirements. Collaborative work between the two parties is a principal condition to the success of the operation. The Software Engineering Institute (Peeling N. and Satchell J., 2001) warns that producing variants of commercial products is to be avoided as it may create configuration management issues. However the soft implementation of modifications, as configurable options, in the commercial products may solve this type of issue.

c. Use of Open Sources Software (OSS).

An active OS users community maintains OSS and the source code is available, which removes the lock imposed by Copyright agreements precluding debugging at the source. OSS could be an alternative solution at medium and long term. Advantages:

- The availability of the source code to the public allows for:
 - Comprehensive quality awareness through code reviews undertaken by the substantial user community;
 - Informal testing on a wide range of platforms;
 - Easy modification of the products; and
 - A cadre of knowledgeable professionals arises that are familiar with OS products and their characteristics.

These advantages for the software developers (Peeling and Satchell, 2001) are:

- Supply of a zero-cost platform to develop their product. ;
- Can run concurrently with current products;
- Can be tailored to a full solution (Operating system + applications)

- No licensing restriction
- Continuous access to the body of Open Sources software; and
- Continuous access to the development tools and expertise of the growing number of developers.

The disadvantages for the developers are: mainly that there no direct marketing drive or budget for the open sources products, and that the products release may occur on a ad-hoc basis. There is also no product support from the vendor but this need not to be an issue if in-house expertise is available.

On the user side, in addition to the fact that there is no single organization committed to support the products, a few negative perceptions exist at management level:

- Free software is unreliable;
- You have no control on the developers that you do not pay;
- Developers may not work on your project.

These perceptions can be easily corrected:

- Software sources are available, they can be checked and modified;
- Participate in the development/modification of open sources products;
- It is your project. Have staff working on it.

In most of the proposed solutions the user environment was modified one way or another to accommodate COTS products. This modification is made possible by the supply of IT skills to bring the COTS product to a level of control and performance suitable to the system requirements. However the control obtained by the user is linked to the lifecycle of the COTS product. Hence when the COTS product lifecycle ends, the COTS replacement problem remains.

An Optimum solution: from “COTS Plug and play” to selective Developmental Items?

a. APIs, interfaces and open sources

Due to COTS use, the IT professional profile has considerably changed during the last twenty years. From the development of full applications and systems, the emphasis has moved to the ability to integrate COTS products and merge their capabilities.

In order to integrate successful SoS, it is necessary to have high quality system components and robust and well-documented interfaces (Maier, 1998). However, the low quality interfaces are the main reason for the glue code and the presence of external performance monitors. Nevertheless the availability of APIs with functional interface capabilities to other commercial COTS is an efficient way to bring visibility to the COTS within the SoS (Figure 3).

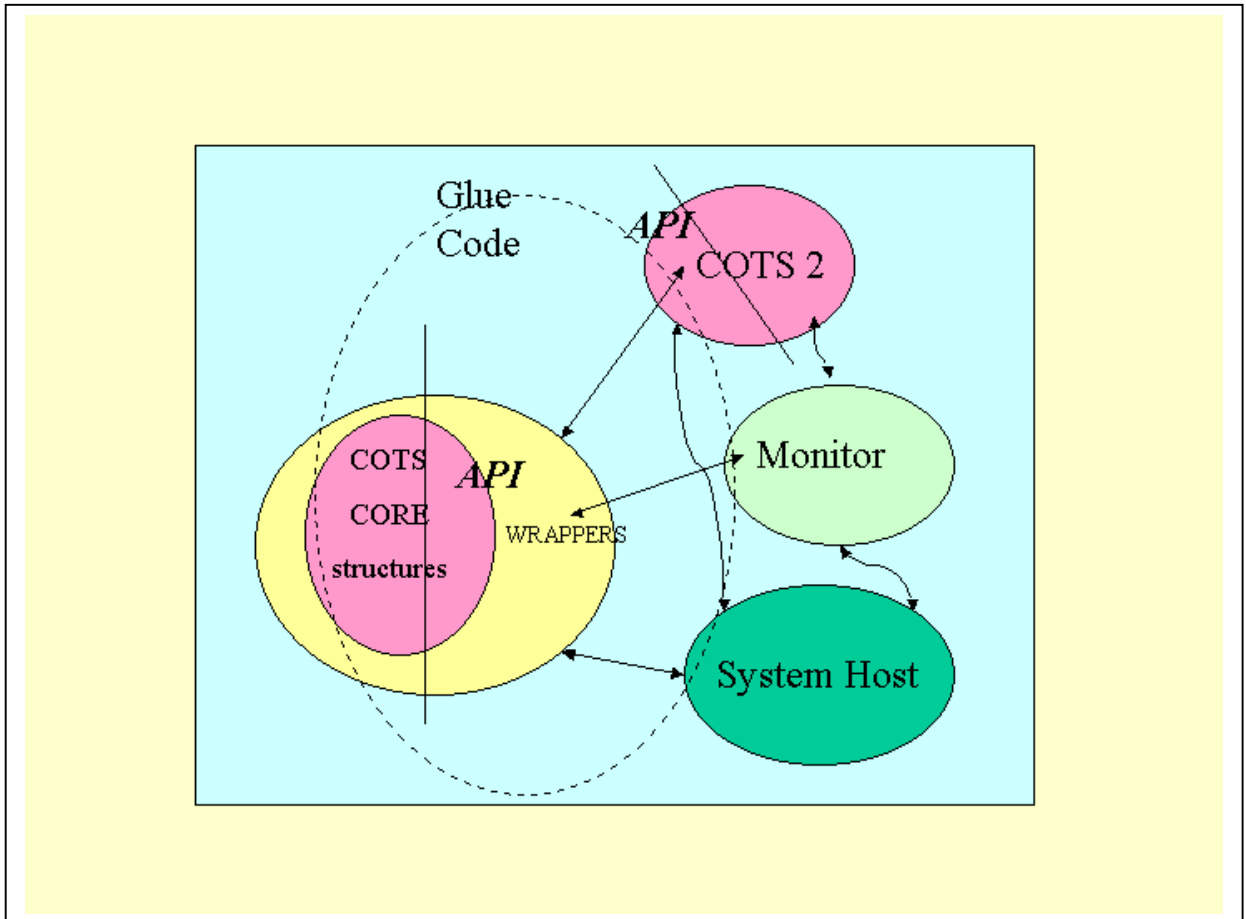


Figure 3: APIs in COTS

b. Principal decision elements

The following elements will determine the decision path taken by the SoS design authority to produce complex systems:

- Existence of a software integration team;
- Nature of the normative guidance on the use of COTS;
- Existence of a functional software development environment;
- Ability to have COTS components modified by the vendor (or in-house);
- Access to Open Sources resources
- Ability to generate Complex Systems
- Sustained ability to generate Complex Systems

These elements are combined to show the decision process between COTS and Open Sources in Figure 4:

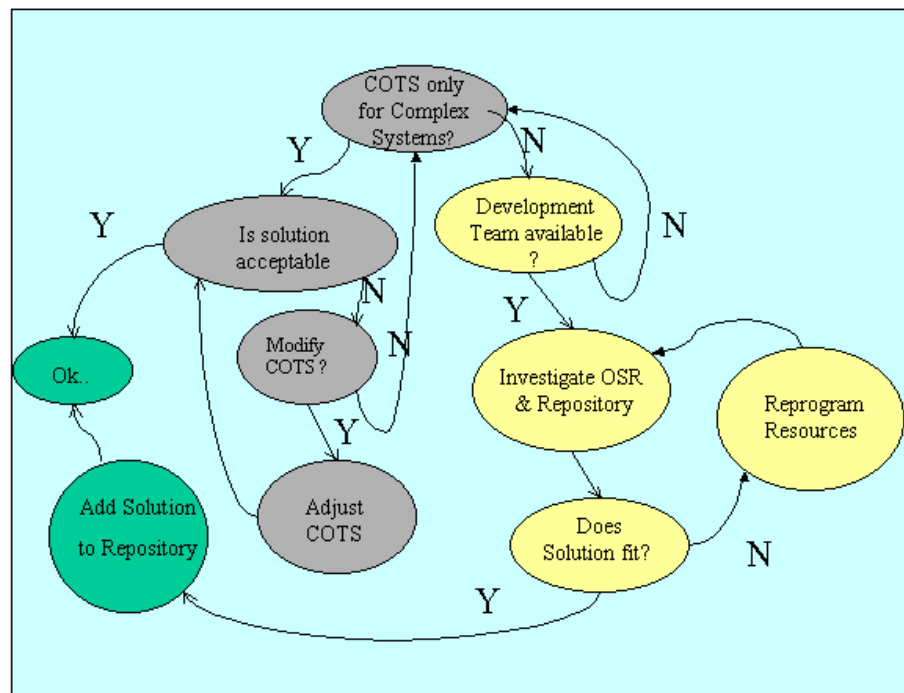


Figure 4: Decision Flow

The decision shown in Figure 4 will contain an important organisational resource component: a solution repository. The consequences of the decision on the creation of this component will influence the quality of the decisions taken after the first iteration of this cycle.

c. Some examples of use

Open Sources Resources (OSR) in different countries and contexts, act as a substitute or a complement to COTS products. Some of examples are:

- UK: GNU C Compilers modified to produce parallel architecture code. The main target is university and research, with a small business window (Aspex Microsystems, UK);
- Germany: substitution of Windows platforms by Linux. Main target is small business, working at IT pc level. The market is expected to jump from Euro 152 millions to day to 307 millions by 2007 (Blau, 2003).
- USA: The US has published a statistical report on the use of Free and Open-Source Software (FOSS) in the U.S. Department of Defence (MITRE, 2003). One of the report recommendations is selective approval of OSR.
- Japan: has been using Open Sources resources to learn, gain control of, and integrate their IT activities in their industry (Tohma and Jacoby, 1997, Nakakoji K., Yamamoto Y., Nishamika Y, Kishida K., and Ye Y., 2002).

The decision to use Open Sources resources is mapped in Table 4:

Country	Area-of-Interest (AOI)	Ability to generate systems	Importance Of the AOI	COTS Usability	Principle Activity In the AOI	Next step of the Open Source Initiative
UK	University	Yes	Marginal	No	Promotion	Commercial
Germany	Small Business	Yes	Substantial	No	Substitution of COTS	Diffusion
USA	DoD	Yes	Primary	No	Alternate Solution	Evaluation
Japan	Industry	Yes	Integration	No	Control & Quality	Production

Table 4: Open Source Use

The variety of these decisions reflects the complexity of the decision made in the selected field of interest. Most of the countries in Table 4, except Japan which used OSR as a transition platform, had a strong IT culture, with the necessary expertise to produce operating systems and complex application software. Therefore the evaluation of alternative solutions to the dominant use of COTS products could be done on an add-on approach, which implied a controlled and gradual growth of the use of open sources resources, in the selected areas of interest. All countries in Table 4 have now indeed a sufficient technological background and supporting institutions to be able to build systems from scratch. Developing countries are also a favorable target for the implementation of development models based on the use of Open Sources.

Conclusion

The use of Open Sources is an alternative to the use of COTS software products, for the design and building of a complex system. However this alternative comes with a cost of providing a development environment similar to the pre-COTS era. No license fee is mandatory for the software, but strong system and application software skills are required. The decision to use this path for complex system development must be carefully made, starting with the evaluation of the available IT skills and cannot proceed without a educational commitment to promote these skills at all professional levels. One very positive aspect of this constraint is the gradual acquisition of the necessary expertise to design and implement reliable, complex systems.

Bibliography

- Abts C., Boehm B and Bailey B., "COCOTS Software Integration Cost Model: Insights & Status"(1999). USC Center for Software Engineering, Southern California SPIN U.C., Irvine, CA .
- Avison D. and Fitzgerald G. (2003), "*Information Systems Development*", 3rd Ed., ISBN: 0-07-709626-6, McGrawHill, UK.
- Blau J.(July 02, 2003), "*Update: High growth rates for open source in Germany*", Infoworld, IDG News Service.
- Ciufo C.(2003), "Quiet Success: NSWC Corona Naval Base's COTS Influence Extends Beyond the Fleet", *COTS Journal*, February 2003, P. 51.
- Collignon S. and Cook S.(2002), "*Review of Software Integration and T&E Techniques of COTS products in the Commercial environment*" , Proceedings of SETE2002, Sydney.
- Collignon S and Cook S (2002), "*Review of Software Integration and T&E Techniques from a NATO Perspective*", Proceedings of SETE2002, Sydney.
- Cook, S. (1999, "*Towards a Unified Systems Methodology for Australia Defence Systems-of-Systems*". Proceedings of INCOSE 1999 Annual Symposium, Brighton, UK, July 1999.
- MITRE Corporation (2003), " Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense", MP 02 W0000101.
- Nakakoji K., Yamamoto Y., Nishamika Y., Kishida K., and Ye Y. (2002), "*Evolution patterns of Open Sources Software and Communities*", white paper, SRA Key Technology Laboratory Grad, School of Information Science, NAIS T3-12 Yotsuya, Shinjuku, Tokyo, 160-0004, Japan. RESTO, JST Japan Society for the Promotion of Science, 8916-5, Takayama, Ikoma, Nara, 630-0101, Japan.
- Peeling N. and Satchell J, (2001), "*Analysis of the impact of Open Source Software*", (SEI draft), QinetiQ, 2001.
- Rechtin E. (1991), "*Systems Architecting: Creating and Building Complex Systems*", ISBN 0-13-880345-5. Prentice Hall.

Tohma Y. and Jacoby R., "*Parameter Value Computation by Least Square Method and Evaluation of Software Availability and Reliability at Service-Operation by the Hyper-Geometric Distribution Software Reliability Growth Model*" (1997), white paper, Department of Computer Science, Tokyo Institute of Technology, Ookayama 2-12-1, MeguroKu, Tokyo 152

Waier M.(1999), "*Architecting Principles for Systems-of-Systems*", John Wiley and Sons, 1998.

White, Ian (2000). "*Wrapping the COTS Dilemma*",. Proceedings of the Conference "The Ruthless Pursuit of the Truth about COTS". NATO, RTO (2000),Belgium.