Clarify the Mission: A Necessary Addition to the Joint Technical Architecture

Ingmar Ögren
Tofs Inc.

The Joint Technical Architecture (JTA) was published to provide the Department of Defense with the basis for needed seamless interoperability across its systems. The JTA contains three architectural views: operational, technical, and systems. The operational architecture view shows the tasks and activities for a system, while the other two views show supporting elements. It is important to understand how the three views relate to each other and how the parts of the technical architecture view support the overall system's missions and operators. This article discusses how a simplified and extended version of the Unified Modeling Language "component diagram" can be used to connect the three JTA views and consequently create a clarified environment for a system's software with a resulting increased probability that the right software will be built.

The Joint Technical Architecture (JTA) was created to establish a standard to provide interoperability among the Department of Defense (DoD) systems [1]. The JTA is a very ambitious and important document with great impact on creating and updating military systems in the United States. Over the long term, it also has considerable impact on civilian and foreign systems since its objectives and approaches go well beyond the U.S. defense sector. The objectives and main content of the JTA are best explained with a few citations from the standard itself:

- The JTA provides DoD systems with the basis for needed seamless interoperability.
- The JTA core contains the minimum set of JTA elements applicable to all DoD systems to support interoperability.
- The DoD JTA provides the minimum set of standards that, when implemented, facilitates this flow of information in support of the warfighter. The JTA standards promote the following:
  - A distributed information-processing environment in which applications are integrated.
  - Applications and data independent of hardware to achieve true integration.
  - Information-transfer capabilities to ensure seamless communications within and across diverse media.
  - Information in a common format with a common meaning.
  - Common human-computer interfaces for users, and effective means to protect the information.

The JTA defines three interrelated views within the architecture as shown in Figure 1: operational architecture (OA) view, technical architecture (TA) view, and systems architecture (SA) view.

> *"... as a software developer, it is not sufficient to understand the technical system; you must also understand the mission(s) and the expected and possible behavior of the operating roles ..."*

These three views are described in the following subsections (citation from the JTA). The article continues with the difficulties in consolidating these views.

The OA View

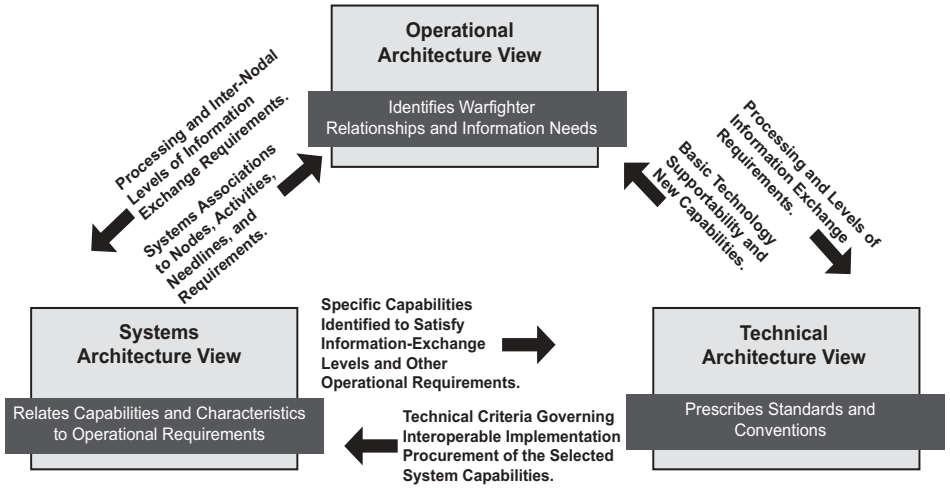
The OA view is a description of the tasks and activities, operational elements, and information flows required to accomplish or support a military operation. The OA contains descriptions (often graphical) of the operational elements, assigned tasks and activities, and information flows required to support the warfighter. The OA defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges in detail sufficient to ascertain specific interoperability requirements.

The TA View

The TA view is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements, whose purpose is to ensure that a conformable system satisfies a specified set of requirements.

The TA view provides the technical systems-implementation guidelines upon which engineering specifications are

Figure 1: *Three Interrelated Views of Joint Technical Architecture*



based, common building blocks are established, and product lines are developed. The TA includes a collection of the technical standards, conventions, rules, and criteria organized into profile(s) that govern system services, interfaces, and relationships for particular systems architecture views, and that relate to particular operational views.

The SA View

The SA view is a description, including graphics, of systems and interconnections providing for, or supporting, warfighting functions. For a domain, the SA view shows how multiple systems link and interoperate, and may describe the internal construction and operations of particular systems within the architecture. For the individual system, the SA view includes the physical connection, location, and identification of key nodes (including materiel-item nodes), circuits, networks, warfighting platforms, etc., and specifies system and component performance parameters (e.g., mean time between failure, maintainability, and availability). The SA view associates physical resources and their performance attributes to the operational view and its requirements following standards defined in the technical architecture.

Inherent Problems with Interfacing

When you talk to anyone responsible for a complex system with multiple operators and a high content of software, you are likely told: "We concentrate on the mission, and our qualified people are our most important resource."

Modern defense systems include personnel (warfighters) to complete missions, with complex interaction between operator roles and technical system parts. This means that, as a software developer, it is not sufficient to understand the technical system; you must also understand the mission(s) and the expected and possible behavior of the operator roles, which are required to complete the mission(s) together with the technical system parts.

Furthermore, when you look at Figure 1 and the accompanying text, you see a multitude of relationships. However, they are rather informal and do not really tell you much about the common core behind the three views. Still further, when you look at the extensive definitions and standards for human-computer interfaces, they are lopsided, meaning that they limit their descriptions to the computer part of the interface and leave the human part to the reader's imagination.

The conclusion is that we have the following set of problems defined when working with the JTA in systems design:

- How do you really connect the three views to each other?
- How do you show how the elements of a system support the system's mission(s)?
- How do you clarify the human part of the human-computer interface?

Introducing Mission and Operator Objects

One way to solve these problems is to start from the Unified Modeling Language's[1] (UML) component diagram with its basic relationships: *depends on* and *included in*. Furthermore the diagram should be extended to include not only software objects but also mission objects, operator objects, and hardware objects. As a result you get a set of object categories as shown in the entity-relationship diagram[2] in Figure 2.

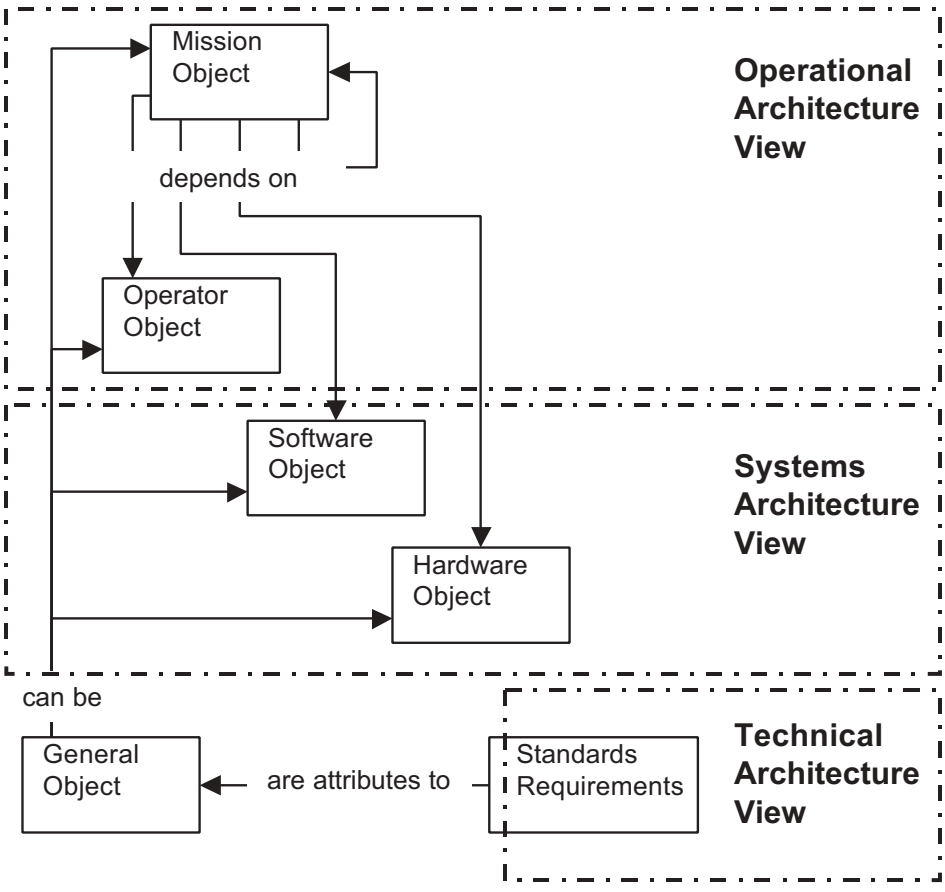
Figure 2 is a general entity-relationship diagram with objects and attributes to objects as entities drawn as boxes. Relationships are drawn as arrows and the diagram should be read along each arrow: <entity> <relation> <entity>. The diagram also applies the principle of *boxes within boxes* to show how smaller entities are parts of larger entities (the three JTA views). Note that the standards included in the TA view of the JTA concern all object categories through the *general object*, although they are primarily applicable to hardware objects.

This diagram shows one way to solve the problems identified above:

- How do you really connect the three views to each other? As shown in Figure 2, the elements in the OA view depend on the SA view, and all elements are derived from the general object. Consequently, the standard requirements of the TA view concern all objects as applicable.
- How do you show how the elements of a system support the system's mission(s)? *Mission objects* are introduced in the OA view to define missions on the system level and also on lower levels.
- How do you clarify the human part of the human-computer interface? Through introduction of operator objects, it is possible to define not only the computer side of human-machine interface, but also the human side with definition of the human operator's behavioral space to match the software's behavior.

The main message of Figure 2 is that

Figure 2: *Object Categories and JTA Views*



you can work with a general object concept that includes objects of categories: mission, operator, software, and hardware. To understand how a mission depends on system parts for completion, you use dependencies to define how each mission depends on other missions and on objects of other categories. This diagram also shows how this view of systems complies with the three views of the JTA.

Command and Control Example

Let us look at a hypothetical command and control (C2) example to illustrate the principles described above. (It is hypothetical since real systems are most often classified and too large for a short article.) While the JTA talks about command, control, computing, communication, intelligence, surveillance, and reconnaissance, this example will concentrate on the core mission, C2, without the support technology, which may or may not be included in a particular C2 system.

Figure 3 shows the top part (main missions) for a C2 system as four mission objects, drawn as a simplified UML component diagram. Note that the methods in the component diagram are represented by *abilities* required for the missions.

Figure 3 also shows how the component diagram can be compacted as a tree or as an indented list to show only the object types involved and their dependencies. This compact form, called the *Tree Graph*, can be used to show a somewhat more detailed view of the C2 system, as shown in Figure 4.

The Tree Graph is where you can see how the three aspects of the JTA can be managed together, clearly connected in the dependency tree as follows:

- The missions shown (C2 supported by plan tactical mission, train personnel, and develop tactics) together with the operator roles shown (planning officer, training officer, and tactics development officer) belong to the OA view in the JTA.
- The software system parts shown (planning support, training support, and tactics development) belong to the SA view in the JTA.
- The hardware parts shown (C2 computing system) are governed by the standards contained in the TA part of the JTA.

Here, the experienced reader may have the following objection: "You need much more to build a C2 system!" That is correct, and there are two reasons why so little of the C2 system is included in Figure 4. One is to limit the information to a reasonable amount for a journal article. The other is that this figure also concentrates on demonstrating how the Tree Graph can be used to show how a system depends on another system (visualizing *systems of systems*).

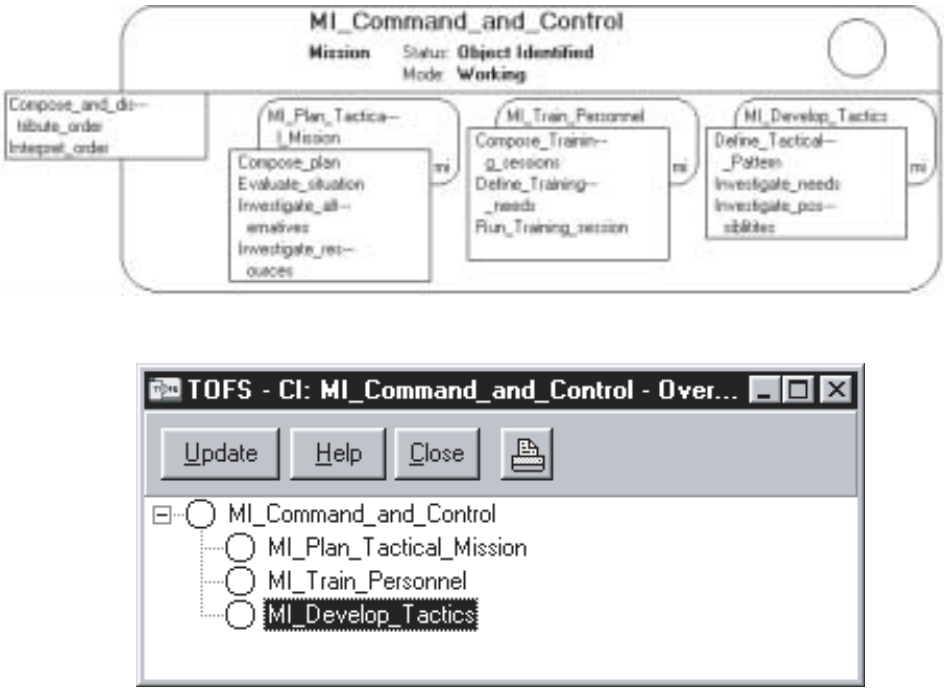


Figure 3: *Component Diagram and Tree Graph for the Top Missions of a C2 System*

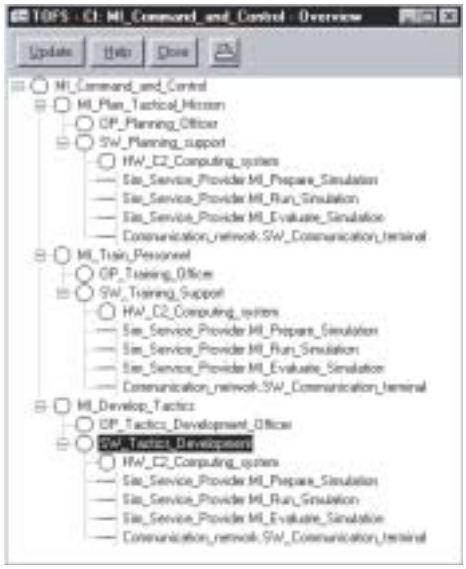
You can see how the three software objects shown depend on a communication network and on some external system to provide simulation services. This makes it possible to keep the C2 system and the simulation system separate and still clarify their interdependencies.

Practical Experiences

The principles related here have been applied in various real systems, primarily C2, communication, simulation, and avionics. The experience is that the principles work and result in system descriptions that are understandable both to end users and developers. However, since this is a new way of viewing a system, the following indications of uneasiness have been noted:

- Software engineers find it strange to work with the *human part* of human-computer interfaces. However, after some time they will most often accept that this is needed to build the necessary understanding to create the right software.
- Some end users might be afraid that working with the missions together with a contractor might result in *too much knowledge* of classified principles on doctrine with strategic and tactical principles. This may be a very real problem. Managing it requires an understanding that it is next to impossible for a contractor to produce a useful system without knowledge of why it is built, and how it should be used.
- Operator end users may be hesitant when contractors try to describe their work as system components (operator role objects). However, as soon as they really study these descriptions, they often get fascinated and produce some extremely valuable comments and criticism, which will be a real help to building the right system.
- People will consider it unnecessary to define the mission since "everyone

Figure 4: *Tree Graph for a C2 System*



knows what the mission is." This is correct, but when you start defining the mission, it is sometimes surprising how many different understandings you find within what *everyone knows*.

Perhaps the most important result of introducing mission objects, operator objects, and dependency trees is that they provide a common ground for end users and technical system developers to meet, which results in an increased probability of common understanding of the system.

Summary

The problems concerning the JTA's difficulties to consolidate its *three views* and to clarify system missions and operator roles in system architecture have been discussed. A possible solution based on a simplified and extended UML component diagram has been presented with a small C2 example. Furthermore, some experience from practical application of the principles presented has been described.

From the software engineer's point of view, this means that the software's environment is investigated and clarified prior to software design and programming. This will increase the software engineer's understanding of the software requirements and consequently also increase the probability that the right software is built.◆

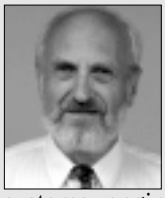
Reference

1. JTA Development Group. Joint Technical Architecture. Ver. 3.1. U.S. Department of Defense, 31 Mar. 2000 <www-jta.itsi.disa.mil>.

Notes

1. The following Web sites provide information on the UML component diagram: <www.sparxsystems.com.au/EAUserGuide/component_diagram.htm> and <http://jliusun.bradley.edu/~jiangbo/uml/Booch_uml/sld019.htm>.
2. The Tofs Web site provides information about the tool used for diagrams in this article <www.toolforsystems .com>.

About the Author



Ingmar Ögren has worked with the Swedish Defense Material Administration and various consulting companies in systems engineering tasks associated with communications, aircraft, and command and control. He is currently a partner and chairman of the board for Tofs Inc. and Romet, a systems engineering consulting company mainly utilizing the Objects for Systems development method. He also teaches systems and software engineering and has had several papers accepted at international conferences. Ögren is a member of Modeling and Simulation in Sweden and International Council of Systems Engineering. He has a master's of science in electronics from the Royal University of Technology in Stockholm.

Tofs AB
Fridhem 2
S-76040 Veddoe, Sweden
Phone: (+46) 176-54580
Fax: (+46) 176-54441
E-mail: iog@toolforsystems.com