

Chapter 2

The Domain of Interoperability in the U.S. Department Of Defense

By
Captain Jerome D. Rosen, US Air Force Reserve
Captain Jennifer L. Parenti, US Air Force
Dr. John A. Hamilton, Jr., Auburn University

The views expressed in this chapter are the opinions of the authors, and do not reflect the official opinions of any U.S. government agency.

2.1 BACKGROUND

Interoperability is a hard problem and there has been no shortage of searches for quick fixes. Comparatively easy problems are fixed as “proofs of concept” which then wither when unable to scale up to the huge size of the DOD enterprise. During the Clinton Administration, the JFPO was tasked by the Under Secretary of Defense for Acquisition, Technology and Logistics (USD AT&L) to study interoperability tools.

There are many so-called interoperability “tools” on the market; some are actually useful. Most do not scale well. That is, they perform well for small demonstrations then choke when confronted with a communications architecture of the magnitude typical of a US Combatant Command. In order to perform any meaningful analysis of individual interoperability tools, Captain Rosen and Captain Parenti developed a methodology and analyzed the domain against which the tools were to be evaluated.

The results were and are instructive. The research team determined that in many cases, the tools themselves were driving processes. In our view, tools should be developed to support processes, not drive processes. First, we will describe the methodology developed and then move into the domain analysis.

2.2 GOALS, ASSUMPTIONS AND APPROACH

Based on the broad guidance received from USD(AT&L), the research group established three goals for the study:

1. Ensure tools are developed to address required functions
2. Avoid redundant development efforts
3. Provide the *right* connections between tools

Achieving these goals required a rational framework for evaluation and a respect for potential users throughout the department. In other words, the tools had to meet or exceed a rational set of criteria and should be reasonably usable for the DOD workforce. Early experiences with some interoperability tools with horrific user interfaces sensitized the team to the importance of a user-friendly front end.

In order to attain these goals, we applied the process depicted in Figure 1. The two upper blocks show a set of “one-time” conceptual work that was required to reach truly general conclusions. By defining the domain over which interoperability is applied in the DoD, we have laid out the battlespace so that we can position tools within it. In order to support this placement, we developed a functional model of the tasks within this domain, against which we can conduct tools.

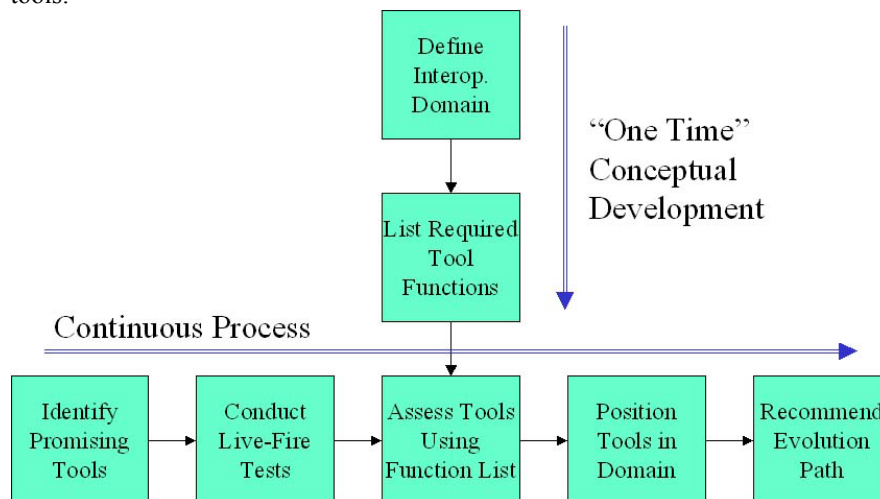


Figure 2-1 *Interoperability Tool Study: Approach*

Finally, we recognized that you cannot evaluate tools without using them. Having seen more than one “content-free” analysis, we opted to eschew conclusions

drawn without data. We also observed that from a tool perspective (as well as other perspectives, architecture and interoperability are related.)

Leveraging on existing efforts within our offices, we identified the tools that had the most promise, conducted live-fire tests of these tools, and then assessed the tools against the functional model. Using the results of this, we were able to better understand how the tools are related and can make some recommendations about how future development should proceed.

The process described across the bottom should really be a continuous process as the capabilities of tools evolve, but we believe this will receive only intermittent attention from our office as major changes occur.

The context for evaluation recognized that different users have different needs. For example, tools useful to a staff officer in a service acquisition command may not be useful to an operational planner serving on a Combatant Command staff. Tools and tool users often have agendas and the research group worked to avoid a “not invented here” syndrome.

2.3 INTEROPERABILITY DOMAIN IN THE DOD



Figure 2-2 The Rosen-Parenti Model Domain

The Rosen-Parenti Model, depicted in Figure 2, is divided into four quadrants. The upper half of the model represents the domain of the planning and acquisition communities. The lower half of the model represents the operational community. The left half of the figure represents the warfighting Combatant Commands and JTF commanders. Activities which fall in the Joint warfighting arena will be shown there. The right half of the figure represents the Services and the Defense Agencies. Activities which fall in their areas will be shown there. Finally, representative

Brigadier Tim McKenna, Ph.D., Royal Australian Artillery, observed at the 2001 Pacific Architectures Conference that Americans duplicated joint and service staff functions. This observation was obvious to us during the domain analysis. Recognizing that function duplication does exist, the research team used observed that the clear trend since the 1986 Goldwater-Nichols Act has been for the Combatant Commands (CINCs) to assume more and more of the war fighting tasks while the Service Staffs focus on providing and supporting forces to the Combatant Commands.

Further confusing matters, there are a large number of players engaged in acquisition activities. Many of the worst interoperability problems come from “rogue” acquisition efforts, that is, home grown systems developed independently from various commands and agencies. For the purposes of this domain analysis, we opted to follow the officially prescribed acquisition process that protects the oft-quoted but rarely understood Title 10 responsibilities of the services.

2.4 CYCLING THROUGH THE DOMAIN

Many problems and most interoperability problems start in requirements. A full discussion on the merits of the CJCSI 3170 Requirements Generation Process is beyond the scope of this chapter. However, we next map that process against the previously defined domain in figure 2-3.

2.4.1 From CRDs to SPOs

We start in the upper left hand quadrant. This quadrant, the intersection between the CINCs and the Planning and Acquisitions Agencies, represents the Joint Planning Community, which includes the Joint Staff, US Joint Forces Command, the JROC, and other functionally-oriented agencies such as JTAMDO. Here, the requirements process begins with the formulation of functionality-based requirements in the form of Capstone Requirements Documents (CRDs). As CRDs are nearly always overarching requirements documents with many underlying applications, they easily fall into the realm of joint cooperation, and are rarely service-centric. They are formed along functional lines.

Requirements flow from the CRDs to the Operational Requirements Documents (ORDs). ORDs are organized by systems which may be joint or service-specific. As such, they straddle the line between the CINCs and the Services.

Note that since CRDs are functionally organized, and ORDs are organized by system, there is a “many-to-many” relationship between them. A CRD will generally require a set of ORDs to implement all required capability. An ORD may perform functions from multiple CRDs, and must trace its requirements back to all CRDs that it supports.

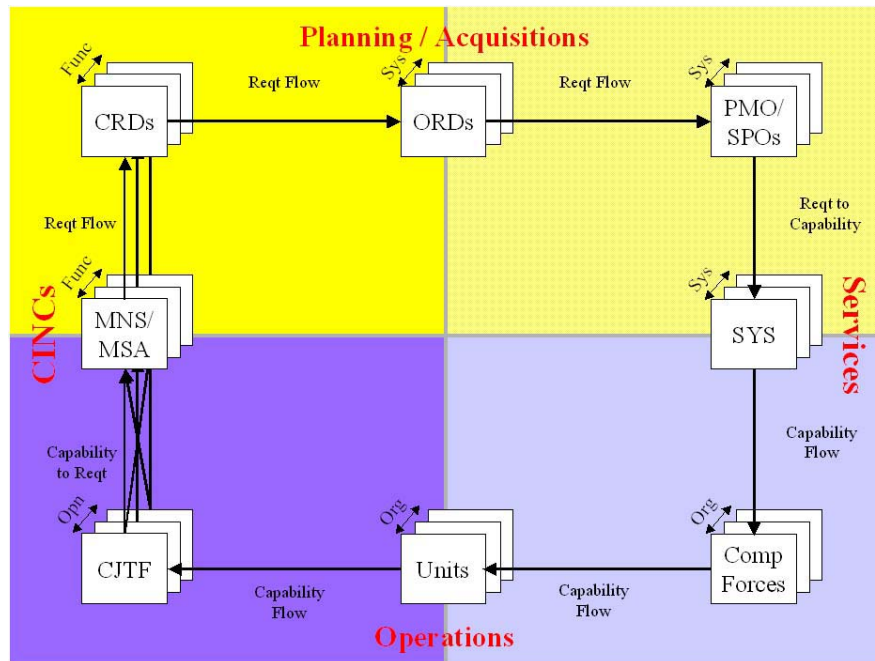


Figure 2-3 The Rosen-Parenti Model Domain

Requirements flow from the ORDs down to the Program Management Offices (PMOs) and System Program Offices (SPOs) for implementation. SPOs are most often organized by system, although there is a recent move to make them more functionally oriented. Likewise, these activities most often fall in the realms of the services, even in the acquisition of joint systems, which usually have a lead service for acquisition. However, we recognize there are exceptions to this model.

2.4.2 From SPOs to Component Forces

The systems which are produced by the SPOs represent the transition from requirement to capability. It also represents the line between the acquisition community and the operational community.

As systems get fielded, capability flows from the SPOs and acquisition agencies to our component commanders and forces. Forces are hierarchically organized by command structures, or, rather, along organizational boundaries. This is another “many-to-many” relationship, as systems are integrated into multiple organizations, and each organization will use multiple systems.

2.4.3 From Service Components to Combatant Commands

The transition from our home-based and service-organized forces to our deployed Joint Task Forces occurs through the deployment of individual units, organizationally structured, which come together to form a joint warfighting force. For example, Battery F, 7th Field Artillery is an Army component unit assigned to the 25th Infantry Division (Light) at Schofield Barracks, Hawaii. Battery F could be assigned to either a standing Joint Task Force (most likely) under US Pacific Command or assigned to a specially created JTF to support a specific joint or combined operation.

2.5 COMPLETING THE CIRCLE

JTF Commanders can turn capabilities back into requirements through the formation of Mission Need Statements, Mission Solution Analyses, Mission Area Analyses, and Mission Need Analyses. These, in turn, feed back into the creation

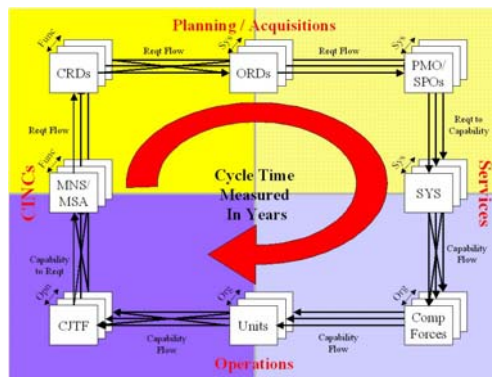


Figure 2-4 Cycle time measured in years

of the CRDs. In theory, the flow of capabilities and requirements start and stop in the Combatant Commands. In practice, the cycle is a long one with many detours and delays and a cycle time of many years as illustrated in Figure 4. That it takes years to complete this cycle is in direct conflict with the accelerating rate of technology change. This can only bode ill for a technology dependent military force.

Ideally, it is the concept of operations that ties all the functional areas together, from requirements generation to deployment. But there are really many different CONOPS – some that are functionally-oriented, some that are system-oriented, some that are organizationally oriented, and some that are oriented toward a particular operation or engagement of force. The relationships between the various concepts of operation are very complex. They must all be consistent with one another -- and the intersections between them govern their relationships. We can think of each of them as slices through a “master” CONOPS that covers the entire domain.

Since, in some respects, the operational architecture is the embodiment and implementation of a CONOPS, we might better understand the role of the CONOPS if we understood how operational architectures are used throughout this process. In addition, we will examine how system architecture, in tandem with operational architecture, is used throughout the process to support interoperability.

2.6 ARCHITECTURE IN THE ROSEN-PARENTI DOMAIN

This study looked at architectures in terms of the Rosen-Parenti domain and the users of the architecture. Simplistically, we can say that the Operational and System Views establish **what** systems must connect and the System and Technical views establish **how** systems must connect. For this reason, the study evaluated System and Operational views in terms if each of the four quadrants in the domain as shown in Figure 5.

Three architectural views were defined in the early versions of the Army Technical Architecture and later the Joint Technical Architecture. A more detailed treatment of the Operational, System and Technical views was set forth in the C4ISR Architecture Framework Document. A detailed treatment of these three views occurs later in this volume in the chapters on Software Architecture, Bilateral Interoperability and Interoperability Process.

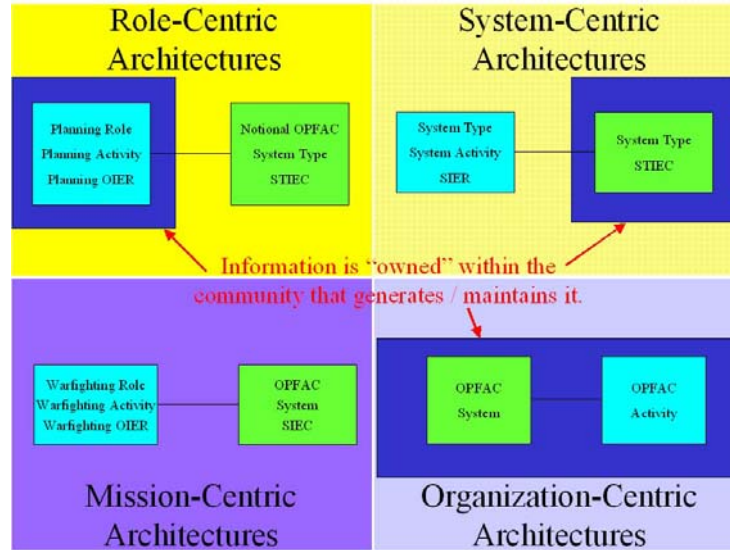


Figure 2-5 Architectures aligned with quadrants

2.6.1 Role-Centric Architectures

In the role-centric operational architecture (Figure 6), a functional area is defined in terms of the roles required to perform it and the activities allocated to those roles. Since these activities are the ones that we *plan* to be performed with the roles, we refer to them as planning activities. Operational Information Exchange Requirements (OIERs) document the required interactions between planning activities. Again, since these are planned exchanges, we refer to them as planning OIERs.

In the role-centric system architecture, notional Operational Facilities (OPFACs), which are manned system-of-systems, are defined within the context of their ability to meet the roles defined in the corresponding role-centric operational architecture (corresponding by functional area). The system types that are part of a notional OPFAC have the ability to exchange information with other systems. We refer to the ability of one system type to exchange a particular piece of information with another system type as a system type information exchange capability (STIEC). This capability is exactly analogous to an IER in that it includes the same information, but represents a *can* rather than a *must*. (It is worth observing, at this point, that in a real implementation, we would expect the STIEC to be associated with information about *how* the information is exchanged, such as the network or protocol used. We omit these details from this discussion for simplicity.)

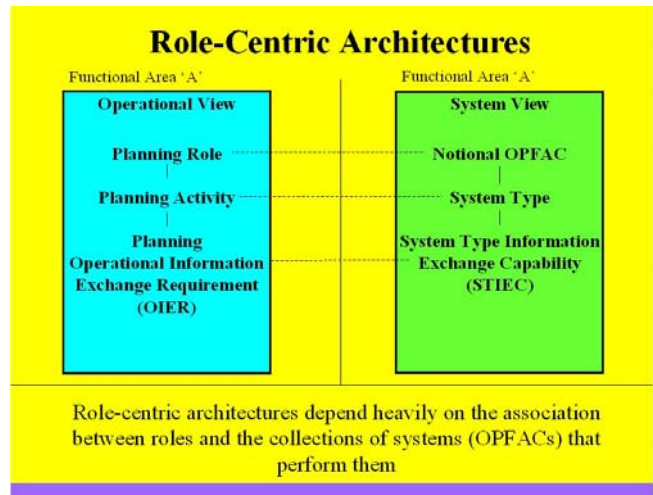


Figure 2-6 *Role-Centric Architectures*

Notional OPFACs are assigned to the roles which they can fulfill from the operational architecture. Each system type within the OPFAC is bound to the planning activities it will perform. Finally, STIECs can be associated with the planning OIERs they implement in the Operational Architecture. In the role-centric case, the system architecture is meaningless without the context of the operational architecture. (I.e. the purpose of the system architecture is its ability to define systems that can meet the objectives of the operational architecture.) The role-centric architecture can be viewed as a set of associated bindings between roles and OPFACs (i.e. the operational and system architectures are not really separable; but are rather two views of a larger, integrated architecture.)

2.6.2 System-Centric Architectures

Clearly, the system-centric architecture is closely related to the role-centric architecture. The planning OIERs specified in the role-centric architecture need to be allocated to the system types that will perform those roles. Similarly, the system capabilities, which are the responsibility of the service acquisition community, (rather than the CINC planning community) must be used by the CINC planning community in the development of role-centric system architecture shown in Figure 7.

In more detail, we see the requirements architecture showing what a system is required to do, by the parameters of the role-based architecture. The system type has a set of system activities that it is required to perform. Between sets of these activities are system information exchange requirements. These are similar to the

planning OIERS, but they are phrased in terms of the need for *system types* to exchange data, whereas in the role-based architectures, the IERS (OIERS) were between *roles*. This implies that functional allocation has occurred, and this is part of the process of developing requirements for system types within the overall system-of-systems context specified at the CRD level.

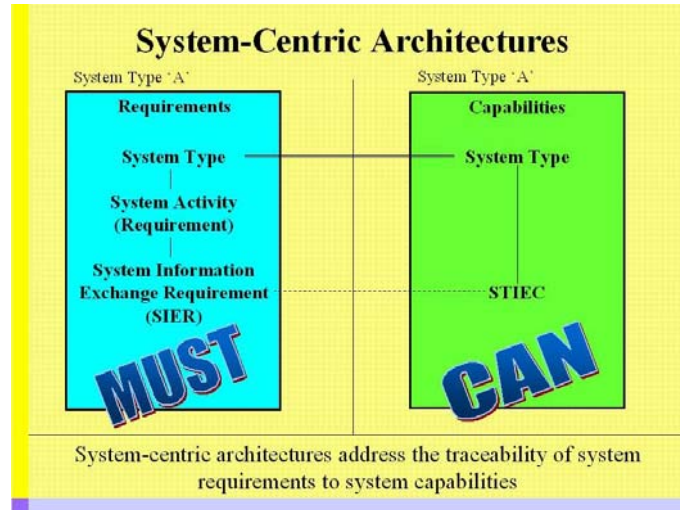


Figure 2-7 *System-Centric Architectures*

The capabilities architecture shows what a system type can do, based on the capabilities of the system. It does so in exactly the same way as the role-centric architecture – using system type information exchange capabilities (STIECs). It also provides traceability to the requirements tree represented by the operational architecture.

Combining the requirement and capability views, we can think of the system-centric architecture as a set of associations between system requirements and capabilities.

The system-centric operational architecture defines, to a PMO/SPO, how their system is being employed (or will be expected to be employed) in the field.

The system-centric system architectures, showing system capabilities, must be kept up-to-date—so that planners creating the role-centric architectures are using the best information to assign systems to roles. These architectures, like the role-based architectures, must factor time into account as well. It is important both to define the capabilities of today's systems and to project the capability expected to be achieved by the systems of tomorrow.

2.6.3 Organization-Centric Architectures

Within the services, organization-centric architectures focus on the specific capabilities and systems within an organizational construct. This can be accomplished at many levels, from the CINCs down to the individual units, at commander discretion.

The organization-centric architectures capture the capabilities and systems that each organization can contribute to an operation as shown in Figure 8.

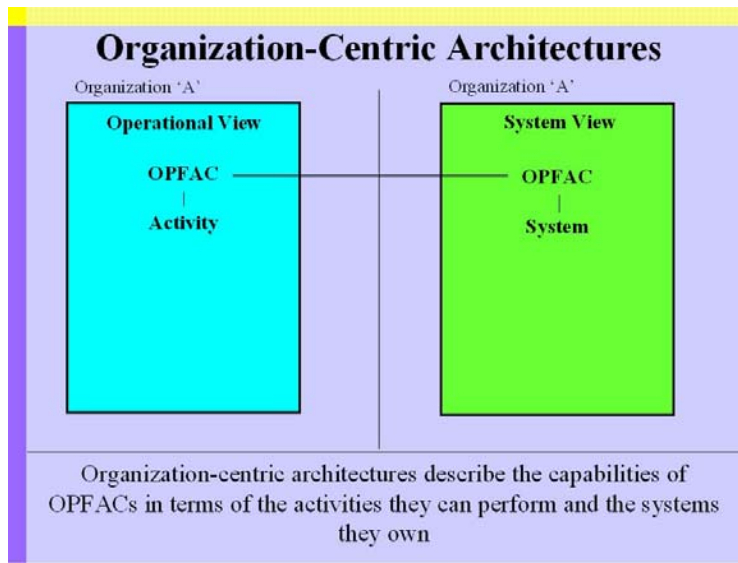


Figure 2-8 *Organization-Centric Architectures*

Organization-centric architectures focus on the specific capabilities and systems within an organizational construct. Operational Facilities (OPFACs) are identified within the command structure as locations where activities take place. In contrast to the notional OPFACs in the Role-Centric architectures, these OPFACs represent specific, actual OPFACs.

The Operational Architecture will identify the activities each OPFAC is capable of performing. Typically, these activities can be mapped to some higher-level UJTL or JMETL task. Since these are activities a unit can perform, we refer to them as unit activities.

The System architecture will identify all the systems located within that OPFAC. The capabilities of those systems will be known, as they are instantiations of

(particular instances of) system types in the system-centric and role-centric architectures.

That we are talking about specific instances of systems, and not just types of systems, is an important point that bears emphasis. If an OPFAC has 12 Global Command and Control System (GCCS) terminals, it is important to document all 12 instances of GCCS, and not just that they have GCCS capability. This is for the purposes of contingency and deployment planning where the number of systems plays a key role.

From the point of view of interoperability, it is vitally important that these architectures are kept up-to-date, so the JTF planners know a unit's capabilities when the unit is added to the JTF.

2.6.4 Mission-centric Architectures

This brings us to the mission-centric architecture, the responsibility of the JTF Commander within the CINCs. Mission-centric architectures focus in on a very specific mission. Choosing functional areas from the role-centric architectures forms them. Then mapping the requirements of these functional areas against OPFACs form the organization-centric architectures. This process of allocating units to functions is clearly the core of CJTF formation.

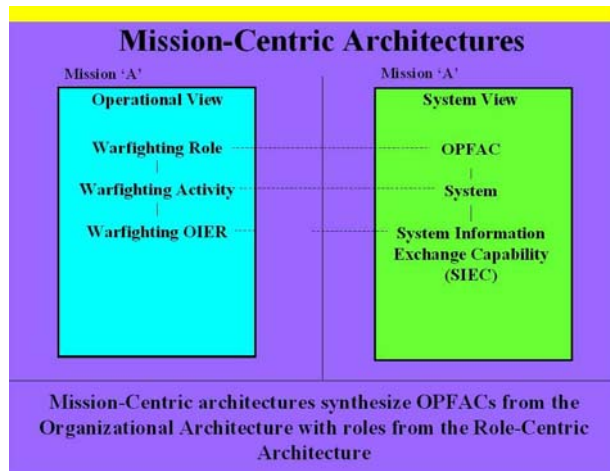


Figure 2-9 *Mission-Centric Architectures*

Looking more closely, the mission-centric operational architecture is a compilation of roles, activities and required information exchanges determined necessary to meet the specific mission at hand as shown in Figure 9.

In this case, the roles are specific instances of the generic roles described in the role-centric architecture. To show this difference, we refer to them as mission roles. In the same way, the activities are specific activities required to perform a mission, so we refer to them as mission activities. Similarly, we refer to the OIERS as mission OIERS.

The mission-centric system architecture is a compilation of OPFACs and systems that are available to fulfill the roles identified in the operational architecture. Based on the system types of the systems, we can identify the system information exchange capabilities (SIECs) as necessary from the STIECs in the role- and system-centric architectures.

The operational and system sides of this architecture are explicitly linked, as a change in one must force a change in the other in order to fulfill the mission. Any disparities between the two must be abated either by adding organizations or systems, or by eliminating activities. Otherwise, there will be activities and functions assigned on the operational side that cannot be supported on the system side.

2.6.5 Validating a Mission-Centric Architecture

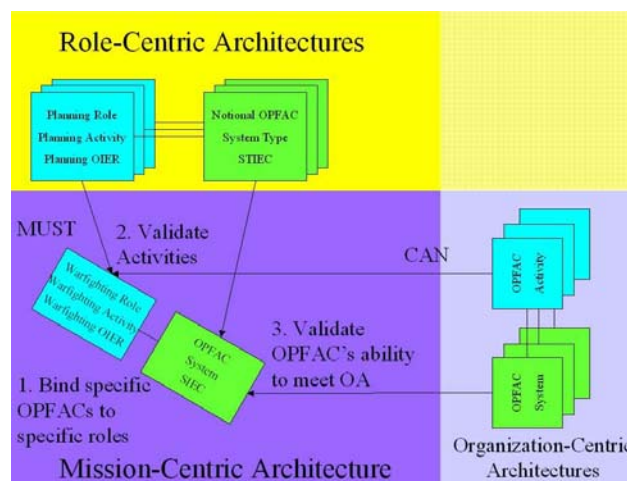


Figure 2-10 Mission-Centric Architectures

When building the mission-centric architecture, we are drawing information from role-centric and organization-centric architectures. We are selecting (and instantiating) roles from role-centric architectures and binding them to OPFACs from organization-centric architectures. We must then perform two concurrent validation processes to ensure usefulness as shown in Figure 10.

First, we must validate the list of activities required to complete a specific mission (the mission activities, which are instantiations of planning activities in the role-centric operational architecture) against the list of activities that can be performed by assigned units (the unit activities in the organization-centric operational architecture). This ensures that the units selected are appropriate for fulfilling the mission assigned mission. (This is not an interoperability driven task, but rather a check of functionality.)

Second, the list of system types capable of performing the assigned roles (from the role-centric system architecture) must be validated against the systems owned by the forces (from the organization-centric system architecture) to ensure that these units are properly equipped to fulfill these missions. (An example of a unit that might be capable of fulfilling an activity, but not have the proper systems: a unit capable of providing Close Air Support may be able to fulfill the activity of CAS, but may be inappropriate to a given scenario if it is not equipped with, e.g., the ability to talk to NATO forces.)

If there are any discrepancies in either validation loop, the mission-centric architecture must be modified to accommodate whatever capabilities are available—this may mean a quick deployment of systems from other units (or acquisition centers), the deployment of different forces from other areas, or may require a change in the mission altogether, if it is determined that we just do not have the capability to support that mission at that given moment.

We can now look across the entire process and see the importance of architecture throughout the system's lifecycle, in all parts of the DoD (Figure 11). We see how architecture is used to determine the interoperability requirements at the functional level in the role-centric architecture; how these requirements are translated into system requirements and implemented in the system-centric architecture; how the locations of the systems throughout the force are maintained in the organization-centric architecture; and how the CJTF commander draws on the role-centric and organization-centric architectures to match force with function in a way that is supported by the systems.

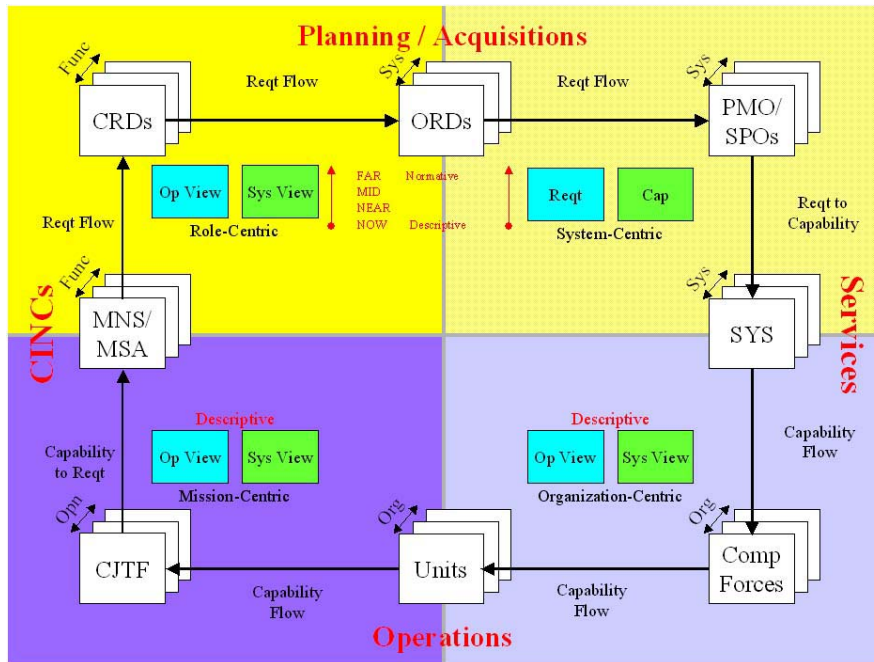


Figure 2-11 Lifecycle Architectures across the Rosen-Parenti Domain

2.7 SUMMATION OF THE ACTUAL TOOLS STUDY

At this point we have now completed the “One Time Conceptual Development Phase” of Figure 1. Having laid out the interoperability domain, we will describe how the tools we have examined fit into the domain. Note that we are omitting the important steps that show you how we arrived at this conclusion – essentially, we developed a functional model by examining the data objects and relations between them from the interoperability domain model. We then compared the functions performed by the tools with the functions in that model, and used this to determine which tool best supported each process. This brings us to the point indicated in Figure 12.

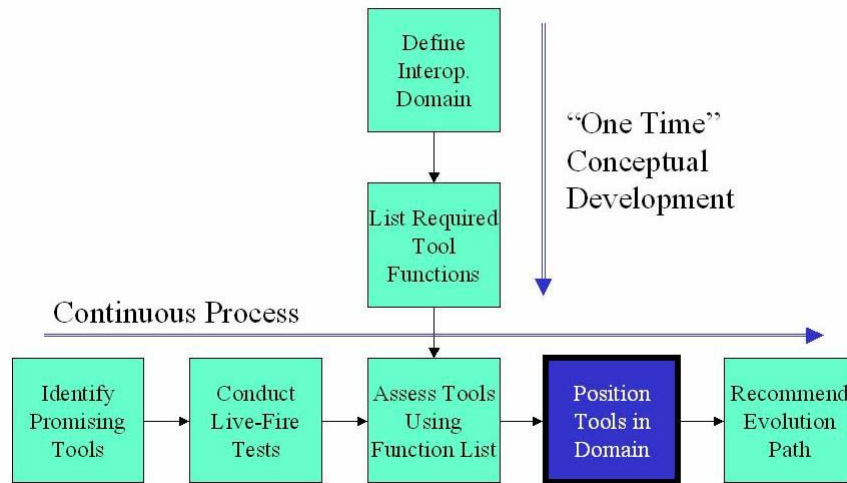


Figure 2-12 *Interoperability Tool Study: Approach*

Three tools were found that roughly mapped into three of the four quadrants. The tools evaluated were LISI (discussed later in the chapter on interoperability process), InterPro and JCAPS (discussed later in the chapter on bilateral interoperability.)

Looking at how tools support this process, it was our tentative assessment that the then front-running architecture tools, InterPro, LISI and JCAPS fit into this model as depicted in Figure 13.

This is not to say, however, that the tools, in their existing states, satisfy all the suggested requirements for each of these activities. It is more to say that the direction of these tools is, or should be, in the direct support of these activities. Note that there exists no tool today to support the creation of the mission-centric architectures. Although JCAPS could certainly be used to document a mission-centric architecture, it does not provide any capability that facilitates the integration of role-centric and organization-centric architectures. A tool filling this role needs to have the capability to extract information from the other architectures and bring that information together for JTF planning and analysis.

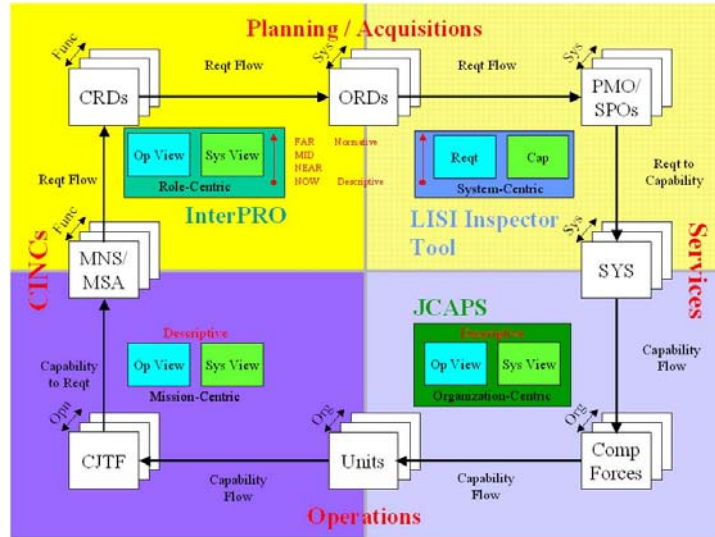


Figure 2-13 Interoperability Tool Study: Approach

2.8 CONCLUSIONS

As we looked more closely at these tools, we realized that the tools could not be considered outside of the context in which they were used. Since interoperability issues are driven by large amounts of data, one had to consider what data was available within a tool, in addition to the functionality of the tool, in order to consider its utility. When looking at long-term utility, it was important to consider how the data was managed and maintained – and, in fact, whether it was likely to be managed and maintained. This could only be done by looking at the way the tool was embedded in the business practices of the tool-using community. Thus, the environment in which the tool is used is as important, perhaps even more important, than the tool itself.

Also, as reported earlier, we recognized that there were four distinct communities and that each had different requirements for tools.

Ultimately, to achieve interoperability goals across the department, these communities must interact using a process driven by exchange of architectural information, rather than by an exchange of static documents. Only through the use of collaboration at the data level can the interests of all the parties be realized in a dynamic environment driven by rapidly changing commercial capabilities. Although providing tools for this type of collaboration will help, better tools are

not enough to achieve this revolution in business affairs. High-level direction will be required, as we will discuss on the next slide.

To specifically address the tasking, the LISI Inspector Tool shows promise for treating interoperability problems involving implementation. Therefore, it should help address many “simple” legacy problems, which deal with small groups of known systems where the operational requirements are fairly well understood. It should be noted that the LISI Inspector Tool should not be confused with the LISI Reference Model which is addressed in detail in the Interoperability Process chapter.

For more complicated problems requiring better operational requirement definition, other tools are needed, and they cannot be used by the CIPOs in a vacuum. ASD(C3I)’s Architecture Directorate is undertaking an effort to develop a suite of architecture and interoperability tools that will work together to help the broader community address this type of problem. We strongly support this effort. However, we feel that this work will only be successful if someone is given both the resources and the authority to integrate these efforts.

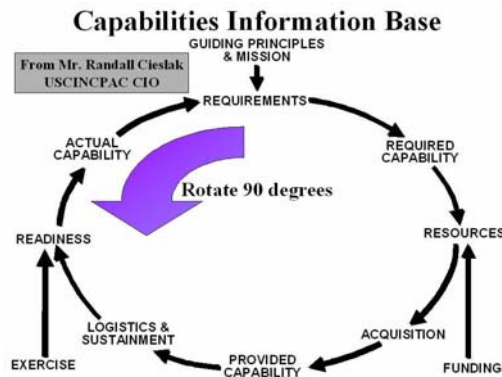


Figure 2-14 *Capabilities Information Base (Cieslak)*

One interesting result of this research project was that the Rosen-Parenti Domain was developed independently of some similar work being conducted in the US Pacific Command. Interestingly, the domain definitions were quite similar as shown in Figure 14.

This model is remarkably similar to our depiction; the mapping can be found by rotating Mr. Cieslak’s slide 90 degrees counterclockwise and superimposing it on our domain as shown in Figure 15.

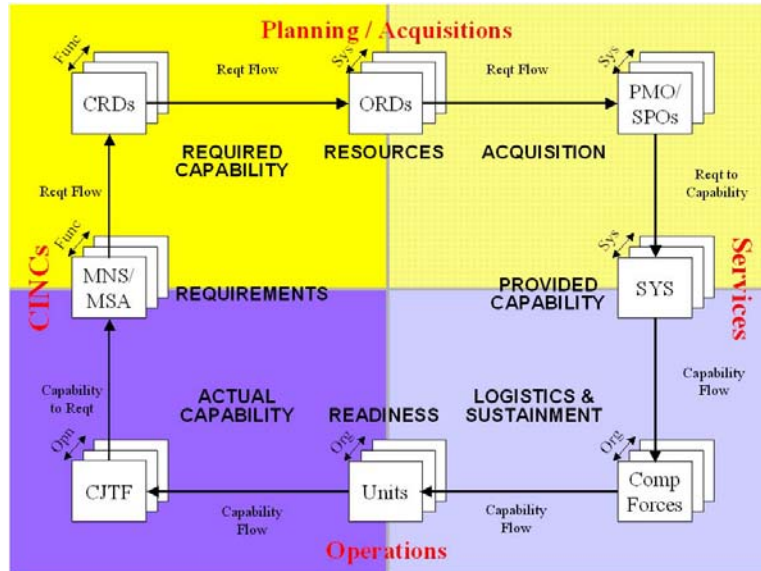


Figure 2-15 Cieslak's Capabilities Information Base mapped to the Rosen-Parenti Model Domain

As can be seen, the two models are virtually identical. Mr. Cieslak's external arcs are:

- Funding, which feeds into resources, now at the top of the model;
- Exercise, which feeds into readiness, now at the bottom of the model; and
- Guiding Principles and Mission, which feeds into requirements, now at the left of the model.

As with most serious research projects, the "Tools Study" took many different directions. The long-lasting impact of this study is not the fleeting snapshot of high cost proprietary tools that could be fitted into one of the domain quadrants.

First and foremost, serious, unbiased analysis of the domain space demonstrated that lack of tools was not the problem. Rather, lack of a defined domain space was a problem. The Rosen-Parenti model remains a useful means of illustrating the acquisition versus operational and joint versus service push-pulls that make any sort of large-scale interoperability solution very difficult. Finally, this project pioneered the concept of user-oriented architectures, recognizing that different organizations and staff use architecture for different purposes.