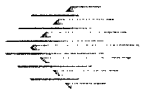


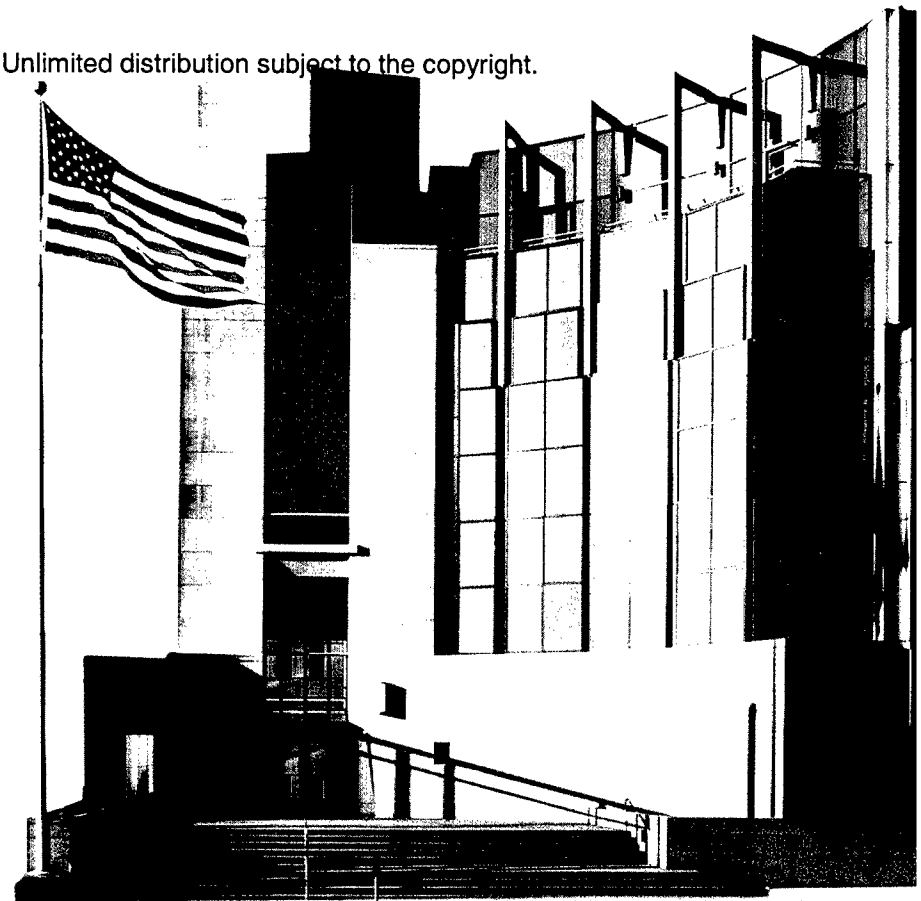
Carnegie Mellon
Software Engineering Institute

Topics in Interoperability: Concepts of Ownership and Their Significance in Systems of Systems

David Carney
William Anderson
Patrick Place

November 2005

Integration of Software-Intensive Systems Initiative



Technical Note
CMU/SEI-2005-TN-046

Topics in Interoperability: Concepts of Ownership and Their Significance in Systems of Systems

David Carney
William Anderson
Patrick Place

November 2005

Integration of Software-Intensive Systems Initiative

Technical Note
CMU/SEI-2005-TN-046

This work is sponsored by the U.S. Department of Defense.

The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Contents

List of Figures

Abstract

This technical note is a brief examination of the concept of ownership and the ways in which it might apply to systems of systems. It first analyzes ownership itself from a number of perspectives and then describes how ownership is generally understood in the context of computer systems. Next, the note outlines some implications that different notions of ownership will pose for large-scale, complex systems of systems, particularly such systems as those envisioned in network-centric warfare. The note describes several real-world examples of ownership issues that exist in existing systems of systems and posits some areas in which research is necessary.

1 Introduction

"How can you buy or sell the land? The idea is strange to us."—attributed to Chief Seattle

This technical note is one in a series concerning various aspects of interoperability in heterogeneous systems of systems. In the first of these documents several properties were proposed as significant characteristics of interoperability [Brownsword 04]. The premise was that a greater understanding of these properties would bring a fuller knowledge of how to develop successful techniques for analyzing, predicting, and designing interoperable systems of systems. Among these properties are evolution [Carney 05], communication, data management, semantic consistency—and ownership, the focus of the present note.

By *ownership*, we refer in general to the different modes of possession, authority, and control that exist over both individual systems and collections of heterogeneous systems. Our purview also includes ownership of the processes necessary for the interoperation of those systems. In some cases a single entity may completely control all elements of a system of systems. However, the more likely scenario, as the world of interconnected systems grows more and more complex, is that no single entity is responsible for the overall composite system.

The topic of ownership is of paramount importance in many domains. In such areas as law, economics, telecommunications, land usage and rights, taxation, industrial dynamics, public policy, and many others, scholars have devoted considerable energy to studying how different nuances of ownership may apply in their fields. In the field of computer systems and software, the topic is currently receiving significant interest, particularly from the vantage point of intellectual property.

The topic is also of signal importance to the Department of Defense (DoD), since the DoD is presently engaged in a large-scale paradigm shift to network-centric warfare (NCW), a concept for which many issues of ownership will arise. Perhaps the most significant of these is that, by our definition of interoperability, the central aspect of NCW is a relationship **between** or among systems. Hence, although the owner of each system (i.e., the agency with authority over the system and its evolution) may be clearly defined, the ownership of the relationship between systems might well be ambiguous. In a context of only two systems, this may not be a significant issue. But assuming the kinds of massive interoperation envisioned in such contexts as NCW, ambiguity about ownership may well become a major barrier to success. We note that while in this report, we use examples from DoD systems to focus on NCW and the likely ownership issues that confront it, our discussions will, in large part, be applicable to other domains.

Given the richness of the topic, we caution that the present report is a high-level overview, not an exhaustive study. We do, however, propose some particular areas in which more detailed and software-specific research is needed, and offer some thoughts about how that research will be applicable to current problems of interoperability.

Before considering our major topic, we begin with a brief general discussion of interoperability that sets out several key concepts that underlie this technical note.

1.1 Interoperability as a Relationship

The term *interoperability* has many definitions; a reasonable one is

> the ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to a shared operational semantics in order to achieve a specified purpose in a given context [Carney 05].

The corollary to this definition is that interoperability is a **relation** between **systems**, where systems are the entities in the above definition. While our focus will be on computer-based systems, the definition extends beyond the world of mechanical systems to organizational and other contexts. To interoperate, one system must provide a service[1] that is used by another. This cannot be achieved without, at a minimum, communication from the provider to the consumer of the service.

1.2 Boundaries of Systems and Systems of Systems

Almost every discussion of interoperability is plagued by one annoying reality: any construct that we label a *system* may in fact be composed of several constituent systems, and this may be true recursively at several levels. In other words, anything that at one level we can call a "system" may actually internally be a "system of systems," and any "system of systems" may itself be part of some larger "system of {systems of systems}," and so forth.

To illustrate, imagine some hypothetical data systems that interoperate in some manner. These data systems could all be elements of a military aircraft's avionics system (e.g., communication or navigation), which together with many other systems (weapons system, mission management system) compose the total aircraft, which itself can be viewed as a single system. To continue to even higher levels, the aircraft is an element in a larger system of systems, since it interoperates with other aircraft and other military units in combat. The process can continue recursively through ever larger systems of systems of systems of systems.

To facilitate our discussion of interoperability, we need to define some level of immediate interest. To do so, we must choose one of these many levels as that of "the system" and the next higher level as that of the "system of systems." The level we choose is, in a sense, arbitrary, since it is only one vantage point within the potentially large scope of this recursive sequence. But it is useful to focus discussion and analysis.

[1] While it seems obvious, it must be stated that provision of service includes the provision of data.

Thus, if our concern at the moment is with issues related to low-level data or semantics of data, we could choose the data systems noted above and their interoperability relationships as our level of interest. At some later time, if our concern lies in some other sphere (e.g., real-time factors relating to the avionics and weapons systems) then our level of discourse could well be the interoperability relationships at that level, and so on.

We illustrate this as shown in Figure 1:

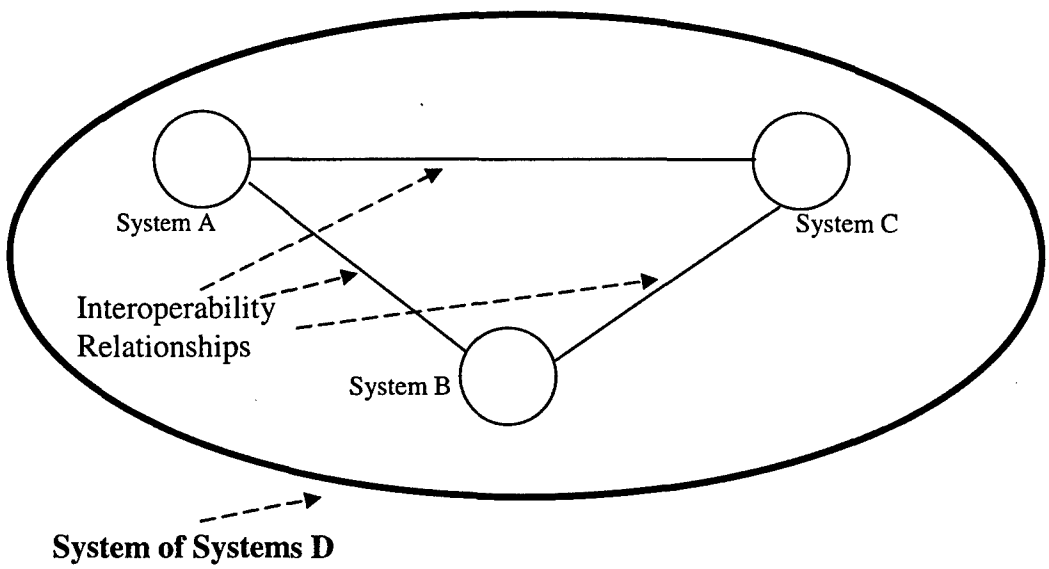


Figure 1: Systems and Systems of Systems

Graphically, the three smaller circles in Figure 1 represent the individual data systems in our hypothetical avionics example. Each is related to two others by some interoperability relationship. The three together as a related unit—that is, the "system of systems"—is depicted by the large, darker oval; this would be the hypothetical avionics system. The smaller circles may themselves each comprise several systems, and the large oval may itself be a single system in some larger context. We temporarily ignore those possibilities and focus only on the interrelationships between A, B, and C that bring about D. By choosing this particular vantage point, we are able to consider the precise nature of the three constituent systems, their interrelationships, and the principles by which the system of systems (D) is brought about.

1.3 Relationships Implemented by Systems

We use this common vocabulary regardless of the mechanism by which a relationship is implemented. For example, let us suppose two systems (A and B) whose relationship is such that they must communicate data back and forth. Let us also suppose that the relationship is implemented by a complex communication system. Since that communication system is, by

definition, a system in its own right, any discussion of these systems may easily be complicated by two different opinions. One opinion sees a system of systems of three entities (A, B, and the communication system). The other opinion sees a system of systems of only two (i.e., by disregarding that the communication system is a **system** and viewing it only as implementing the relationship between A and B).

We argue that either view is possible, depending on the issues of interest and the questions to be answered. For instance, if interested only in the semantics of shared data between A and B, we may be unconcerned with the manner in which the data is communicated. In that case, we can rightly consider the communication system simply as the mechanism that implements the A-B relationship. On the other hand, if we are concerned with the specific details of how System A locates System B, with the significance of timing constraints, and other such questions, then we well may consider that the relationships between System A, System B, and the communication system are all interoperability relationships in their own right.

This issue of relationships and how they are implemented is of particular interest in any discussion of ownership, since it is in the relationships between systems that difficulties of authority and control often arise. We now turn to the topic of ownership and its implications for interoperability. Section 2 considers various notions of ownership, and then puts these notions in the context of interoperability. Section 3 discusses some of the implications and problems that heterogeneous system ownership poses for network-centric warfare. Section 4 sets forth some potential areas of solution to the problems described in Section 3, as well as some areas in which further research is necessary. Section 5 provides a brief summary.

2 Ownership and Interopable Systems

"... He and the cook alone
Receive the morning on their old estate,
Possessing what the owners can but own."

—"A Summer Morning," Richard Wilbur

In this section, we present some perspectives on what *ownership* means in a variety of contexts. We first consider the concept as it applies to computer systems and software in general and then consider how it might apply to systems of systems. An expanded discussion of this material, including the various ways that ownership has been defined historically, is provided in Appendix A.

2.1 Common Concepts of Ownership

Throughout most discussions of ownership, several basic concepts reappear. Among these are possession and property, authority over what is owned, and the issue of roles and responsibilities regarding what is owned; each of these somehow involves the relation between ownership and rights. Each also has some bearing for our consideration of ownership of interoperable software systems in Sections 2.2 through 2.4.

Property and Possession—It is commonly thought that what is owned is one's property, whether tangible or intellectual; that is certainly the basis for most dictionary definitions of ownership. Defining ownership in terms of how rights may be exercised over one's own property is the basis of many legal and economic discussions of ownership. Thus, Bergstrom notes a common view that

> More specifically, ownership is defined as residual control rights to assets, that is, the right to determine the uses of assets [Bergstrom 00].

And Oliver Wendell Holmes holds that

> . . . the rights of ownership . . . are substantially the same as those incident to possession [Holmes 00].

Possession is most easily demonstrated in the case of tangible property. However, in the case of intellectual property (which is critical for software), some questions arise. Most descriptions of intellectual property regard it as similar to, but not the same as, tangible property. A key difference, from a U.S. government definition, is that

> . . . intellectual property is intangible, that is, it cannot be defined or identified by its own physical parameters. It must be expressed in some discernible way to be protectable [Hefter 05].

Authority—Authority is typically defined as:

> The power to command, enforce laws, exact obedience . . . freedom or right granted to another [HM 82].

Given the condition of ownership, some authority is typically either resident or conferred. But authority is not absolute; it exists in gradations and hierarchies, and is thereby a variable aspect of ownership. For instance, one might own a house, but not have complete authority over who may reside there (e.g., there may be a zoning stipulation that no more than two unmarried adults may inhabit a house in a certain area). Therefore, owning and possessing something do not automatically convey the right to use it in any way one wishes; only certain rights are granted to the owner. From this, we note that the limitation on an owner's **authority** modifies the earlier **property**-based definition of ownership (i.e., "ownership [is] the right to determine the use of assets").

Roles and Responsibilities—For both tangible and intellectual property, ownership resides in some person or agency. But there may be a division of roles and responsibilities within that agency. With computer systems, there are usually a number of interested parties, called *stakeholders*, for whom different modes of ownership exist, at least notionally. Thus, users of a particular computer system may not be the legal owners of that system. But their enterprise and its success may be entirely dependent on that system, and those stakeholders will likely have at least some sense of ownership over the system. This suggests the need to consider a number of roles that refine the notion of ownership (e.g., users, maintainers, purchasers, and so forth).

2.2 Ownership of Computer Systems

There are varying notions of what is meant by *computer system*. We will use the term to mean the combination of four things:

1. users and other stakeholders of the system
2. hardware
3. business processes that are employed
4. software

Thus, ownership of "a computer system" aggregates hardware, software, and processes, and is apt to be divided among a number of parties. We shall consider each of these elements separately.

The first element presents little difficulty, since users and stakeholders are not themselves "owned" but are the owners, in some manner, of the other three elements—processes, hardware, and software. In the case of a single system, the human element may not be a complex matter. But given that ownership connotes authority, the human element is of paramount importance when we consider the many different kinds of interoperability

relationships **between** systems. We shall consider this more fully in the paragraphs that follow.

Ownership of the hardware is equally straightforward, since hardware is physical and may be treated as tangible property. For that reason, the discussions of tangible property in the preceding sections are pertinent and applicable to the ownership of hardware. Thus, to the extent that ownership of any tangible property can be reasonably well defined, those concepts apply to hardware as well. The simplest view is that the owners of hardware are, by and large, the persons or agencies that have purchased it.

When we consider ownership of processes, we enter the realm of intangibles, which, while no less real than physical property, may alter how we apply the concept of "ownership." One of the first questions to arise is this: precisely **what** of a process can be owned? One view distinguishes ownership of the process design from ownership of the process execution. Equally critical is the question of how the users own, at least in some manner, the process that they are executing.

Ownership of software poses the greatest difficulty. Software is ephemeral; in one sense, at least, it has no existence outside of its operation in the context of a system. Indeed, it is not obvious what we should consider to be software: is it the source code, the compiled executable, or both? Is it the design or the architecture? The algorithms? All these elements and more have been conjectured to be elements of software that can be owned.

Software thus offers a dilemma for such definitions of ownership as that made by Oliver Wendell Holmes ("...the rights of ownership ... are substantially the same as those incident to possession"), since possession of software is not always well defined. For instance, many software companies sell executables to their clients, who, it would seem, thereby possess them. But those clients rarely own the source code that produced the executable. And it may not even be clear who owns the executable. Many legal rulings, for instance, have been made about "software" that do not distinguish between source code and binary executable code. Thus, in a ruling about taxation of corporate property, the state of Louisiana ruled that

> Company A's software . . . is comprised [sic] of prewritten modules, or programs, that are combined and installed to properly integrate the communications system. . . . Ownership of [Company A's] software does not pass to Company B. Instead, the company is granted a perpetual software license agreement, which allows Company B to use, but not own, the software [Louisiana 04].

By this ruling, ownership of Company A's software, presumably including the binary executables, does **not** pass to Company B despite the fact that Company B purchased it and owns the overall system in which the modules operate.

Software lacks most attributes of tangible property (though not all, as we shall see later), and the major consideration for software is that its intellectual property is the critical aspect of its

ownership. In most cases, the owner of a copy of some software owns only the right to use that software and seldom owns the software itself. And unlike most tangible property, where possession generally implies ownership by one party at a time (e.g., in the case of real estate), software may be replicated at little or no cost. Multiple instances of a software component are essentially equal, yet can be owned differently by many persons simultaneously.

2.3 Ownership and Systems of Systems

However complex or nuanced the different notions of computer system ownership may be, real systems are still, in some manner, owned. Someone has purchased the hardware, some (or many) persons have the right and authority to use its software, and the system enacts processes to benefit users and other stakeholders. So although a system's "owner" may be an aggregation of many different parties, the system's ownership is quite real. Further, the particular manner of ownership affects the way that a system will interoperate with other systems, which may be owned in very different manners.

In Section 1, we described interoperability as a relation between systems, as shown in Figure 2:

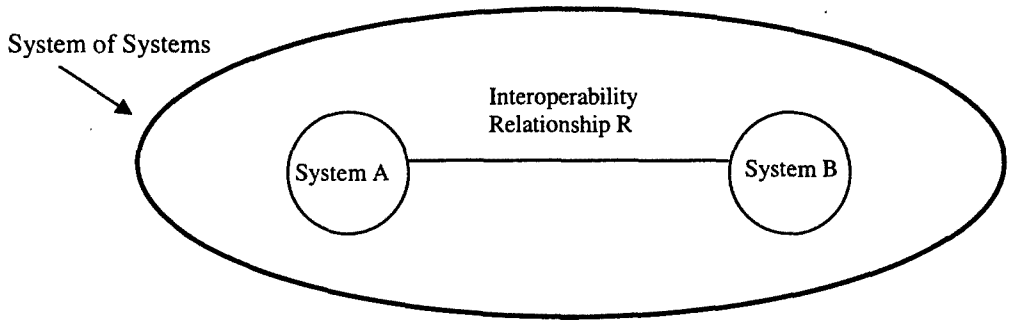


Figure 2: Two Systems in an Interoperable Relationship

For simplicity's sake, let us assume that the relationship R is implemented by something like the Common Object Request Broker Architecture (CORBA). From the machine-machine view of interoperability, therefore, this illustration, though simple, may appear reasonable.

However, there are two flaws with the illustration. First, we have defined a *computer system* to be an aggregation of several things, including its human stakeholders. So while it may seem intuitively obvious that the circles in Figure 2 could represent both hardware and software, the collective human community on each side is missing. And since we are now concerned with ownership (of the two systems and also of the aggregate, interoperating system), we must at least include some representation of the owners and the relationships between them. Thus, a more accurate illustration of the overall system should be as shown in Figure 3.

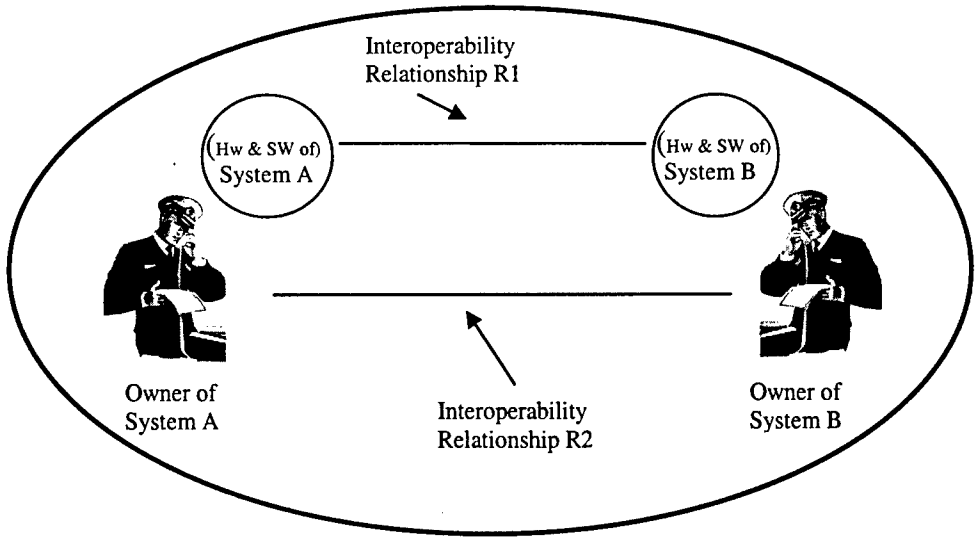


Figure 3: System and Owners in an Interoperable Relationship

The point is that interoperability is not merely a relationship between hardware and software elements but also consists of human, programmatic, and organizational relationships as well. Said more strongly, any significant interoperability will **necessarily** involve multiple relationships in both the machine-machine and the human-human domains.[2]

This illustration also raises questions, however. First, while the ownership of the two separate systems may be (relatively) clear, what can we assume about the overall system of systems? Who owns the relationships between the systems? Using the concepts from section 2.1, who **possesses** the relationships, and in what sense can one possess a relationship? Who has **authority** to make a change to migrate, for instance, the machine-machine relationship from a CORBA implementation to a Web services implementation? What ownership **roles** are played by the users of the aggregate system, and its owner (if any)? In short, what rights exist to the overall system of systems, who has those rights, and how do they influence the interoperation of the whole?

These considerations are not purely academic. Though apparently simple in the context of Figure 3, they will be quite complex in a network-centric environment of hundreds, perhaps thousands, of independent, interoperating systems. If we are to truly succeed in creating and sustaining such large-scale heterogeneous systems of systems, we must address these questions. There are no universally accepted answers obvious at present. But we can begin by considering the most immediate issue, namely, ownership and interoperability relationships.

[2] Note that the illustration is still incomplete, since we have omitted representing the processes and have made no attempt to distinguish human "owners" from users, other stakeholders, and so forth.

2.4 Ownership and Interoperability Relationships

If we speak of *owning a relationship*, it is probably not apt to assume any sense of possession or property. But it is certainly reasonable to consider the element of authority: someone must have the right to alter or sever a relationship. We shall focus on this concept as a means to understanding what the phrase *own a relationship* might mean.[3]

As shown in Figure 3, no authority over either relationship, R1 or R2, is apparent. In the case of R1 (i.e., CORBA), the relationship might even be implemented by another system, which could bring yet another person or agency into the overall mix. In any case, assuming such a relationship exists and supposing the owner of System A wishes to stop supporting CORBA and migrates to some other mechanism, does he or she have that right? What authority would the owner of System B have to prevent this? This question really asks: Who has rights and authority over preserving the relationship? Three answers seem possible: (1) it may be a third party; (2) it may be only one of the two parties; or, (3) it may be both parties.

Underlying this question is the matter of incentive, since the incentive to establish and preserve any relationship must somehow be sufficiently compelling to whoever had authority to bring the relationship into existence in the first place. If we consider the notion of a third party owning the relationship, then the incentive could be simply a matter of command. Hence, some third-party person or agency with authority over both systems, A and B, could have ordered that the relationship be created; the same party could order that it be changed. But this solution is inconsistent in the context of truly independent systems. And even where it has been applicable (i.e., where some person had absolute authority over both systems), this solution has proven inefficient and, in many cases, has simply failed.

The second answer (i.e., one of the two parties owns the relationship) is illogical. If System A's owner really has the sole authority over the relationship and also has the right to sever the relationship (thus affecting the behavior of System B), then System A's owner is, in some fashion, also a stakeholder in System B; in effect, he or she is a partial owner, and the two systems are not actually independent. This solution is a variant of the third-party solution discussed above.

The logical conclusion is that, assuming a relationship between truly independent systems, some degree of mutual self-interest is most effective: each system owner must realize a sufficient benefit that outweighs the cost (e.g., in resources or control) of creating and preserving the relationship. This is not only more logical, it is a more accurate description of reality and is necessary if the relationship is to succeed. Hence, we posit that joint ownership of a relationship, with mutual responsibility for its maintenance, is the most realistic scenario for interoperation between independent systems.

[3] We shall continue to use the term *ownership* for consistency. Where its precise bias (e.g., authority, possession, or rights) is significant, we shall clarify the specific meaning.

In practice, however, there may be some difficulty in defining the benefit to each system of establishing the relationship and then of preserving and maintaining the benefit as both systems evolve. Furthermore, as the aggregate system becomes larger (i.e., as additional systems are added, creating a network of systems), the difficulty will grow for two reasons. First, authority over relationships will become more complex (i.e., some will inevitably become third-party ownership). And second, in very large systems of systems the need for truly compelling incentives becomes more pressing, since the value of the additional interoperability relationships may serve functionality quite distinct from the stand-alone functionality, goals, and stakeholders of each individual system.

3 Implications of Ownership for Network-Centric Warfare

"Charley glanced astern at the fishermen with a look of ownership in his eye which till then had been missing."—"Tales of the Fish Patrol," Jack London

Within the Department of Defense (DoD), one driver for the current interest in large-scale interoperability is the idea of network-centric warfare (NCW). We first provide a brief and high-level description of NCW and then consider NCW in light of some of the ownership issue we have raised in the preceding section.

3.1 An Overview of NCW

The concept of NCW, described in many sources, is one of ubiquitous, ad hoc interoperation among a very large number of heterogeneous systems [Alberts 99, Alberts 03]. Connections between systems (i.e., interoperability relationships) will be made as mission need arises: new relationships formed, operations carried out, and relationships dissolved—all with great frequency.

NCW is based on the assumption of a wide and diverse network of systems and organizations; interoperability is its essential ingredient:

> The basic tenets of NCW begin with the existence of a robustly networked force. Such a force can only be achieved if there is a high level of interoperability among mission participants and the systems that support them. Interoperability, the ability to work together, needs to simultaneously occur at a number of levels or layers to enable entities to communicate, share information, and collaborate with one another. The degree to which forces are interoperable directly affects their ability to conduct network-centric operations [Alberts 03].

In this vision, what we call a *system of systems* is short-lived and ephemeral. The interoperability relationships are not static; they are created dynamically as the context of the required capability dictates. The vision is one that permits users "at the edge" to pull data and information as required, depending on circumstance. Using our terminology, the "edge user" will rapidly and in real time form and dissolve new interoperability relationships as need arises. One account of the NCW vision is

> . . . [the NCW warfighter] is a creature of the Information Age. This is a junior noncommissioned officer who must be able to function across a range of missions and make decisions that have implications far beyond his local responsibilities. For example, [the warfighter] might be responsible for a roadblock late at night during a peace operation. He (or increasingly, she) may have to decide what to do about a civilian vehicle that approaches the roadblock at a high rate of speed and does not appear to intend to stop. If the

> occupants are innocent civilians, then firing on the vehicle may result in casualties (and very unfavorable media reporting) and a serious loss of trust among the local population. However, if the occupants are hostiles, then failure to stop it may result in an attack on his unit, a later violent event (bombing or assassination), or the loss of control over the road. The corporal must make his decision based on his situation awareness (Have there been similar incidents? What kind of vehicle is it? What types of occupants does it appear to contain?), his orders, the rules of engagement, and his judgment or common sense [Alberts 03].

A scenario like this assumes the warfighter's ability to have rapid, real-time access to a wide variety of data, chosen on an ad hoc basis; this will require a very complex technological environment for support. In such an environment, it becomes quite difficult to pinpoint where "the system of systems" actually is, since the system's existence (or rather, the existence of many systems of systems) may be quite brief and may not be the same from the perspectives of any two participants.

3.2 Ownership Issues and NCW

How does our consideration of ownership and interoperability affect the NCW scenario? For one thing, the NCW vision is implicitly predicated on a widespread, global sense of shared ownership over all of the system assets of the entire DoD enterprise. The authority over relationships would not stem from any localized agreements between a particular set of system owners, since it cannot be known in advance from which collections of systems the user will need cooperation. Instead, the authority (e.g., to form new relationships between systems) would be derived from the overall mission needs of the DoD.

This is essentially a vision of ownership that we described above as a "third-party solution," where the authority to create and dissolve interoperability relationships lies not at the individual system owners' level but with a higher authority. In practical terms, to achieve this goal will require that all parties collaborate—the services, and the broad array of DoD acquisition programs—far more than they do presently, to orchestrate acquisitions toward a jointly owned collection of systems. This will necessitate a shift away from the current concept of single-service system ownership and can only succeed in a context of fragmented possession with common responsibility. There is, at present, little evidence that we have the incentives properly aligned with this goal; yet without it, the NCW vision will be extremely difficulty to achieve.

Another ownership implication for NCW is that the dynamic **system** structure that is envisioned will need a correspondingly dynamic **authority** structure as well. To be sure, many writers on NCW have already stressed that the constantly changing and reconfiguring systems of systems will require an organizational shift away from a hierarchical command toward a more distributed and networked command structure called "power to the edge." But this concept needs something far more complex than a simple redistribution of authority.

Instead, NCW will require a dynamic and changeable authority structure, parallel to the expected dynamic system structure, where different degrees of authority are granted "to the edge" at some times and only "halfway to the edge" at other times. Said differently, this means that human-type interoperability relationships would be reconfigurable and dynamic, just as the machine-type interoperability must be.

In this manner, authority could be changed, even withheld; these shifts will differ, depending on changing mission need and circumstance. For instance, a common scenario used for NCW depicts a warfighter on the ground who observes enemy ground troops, dynamically connects with several external communications systems, and calls in air support to make a strike on the enemy. With no other context, this is reasonable. But context may alter this: at the global level, where overall strategic and policy issues must reside, there may suddenly occur some overriding reason, at that particular moment, **not** to permit the users at the edge to form any relationship they wish. There may be some condition where this would give away vital knowledge to the enemy (i.e., that we had spotters on the ground at all). There could conceivably be a greater need, unknown just then to the spotter on the ground, to convince the enemy that its troops were not observed. And, should that be true, it would be necessary, based on that unique circumstance, to withhold "power to the edge" authority at least as long as that circumstance were true.[4]

Finally, there are large-scale economic issues as well. There is a cost in achieving the ubiquitous interoperability required for NCW: Who pays this cost? And not only for achieving, but also for sustaining the interoperability. This question asks, in effect, who has ownership of the whole vs. ownership of the parts. This cannot be solved purely at the level of acquisition funding, since it is an operational issue as well. For instance, it is inevitable that much of the NCW systems' makeup will have commercial components, many of which will entail license and usage fees. This problem is already complex for individual systems; it is rendered even more complex in the ephemeral world of NCW, since "the whole" is always fluctuating. This may create conflict with licensing agreements that have a temporal basis. Thus:

> During the execution of software . . . functions performed by many pieces of software—residing perhaps on many different machines—may participate in accomplishing a specific task. . . . Such fluidity and interoperability among software applications will undermine existing pricing systems. If I use three pieces of software but only a small part of the functionality of each . . . whom do I pay? How do I get charged? The answers are not clear. We are not at all sure we know how to control the assets. . . . Even more problematic are questions about the rights of ownership accorded to each of the application

[4] The classic example of exactly this scenario occurred in the Second World War, where Churchill learned in advance that the Luftwaffe was planning to bomb Coventry but was compelled to permit it to happen. Otherwise the German High Command would have realized that the English had broken the Enigma code [Winterbotham 74]. Note that Lewin disputes this story [Lewin 78].

components that are combined and recombined to create a customized work of software at the user's behest [Branscomb 91].

These ownership issues, and likely many others as well, are not insuperable, but they require resolution before the vision of NCW will be achieved. Also, it is perhaps significant that these issues tend to fall into the organizational, programmatic, or management sphere rather than that of thorny technical matters. But this does not render them more tractable; in fact, solving such problems is no less difficult a task than in the technical area and may prove even more difficult.

3.3 Ownership Issues in Actual Systems of Systems

We now examine three instances of actual DoD system-of-systems projects that exemplify one or more of the ownership issues that we have raised in this report. We shall disguise their real identities, but the problems that each faced are real. For all three, some of the ownership issues we have identified in this technical note were factors in the systems' development or operation and in each case were barriers to success. The systems in question span the full spectrum of the life cycle, from pre-deployment through long-term evolution. While none of these systems manifest the full character of NCW, each of these has at least some of the intended qualities that the vision of NCW will embody.

3.3.1 System A: Pre-Deployment

In the first example, we studied a system during development that is expected to be deployed within two or three years. The system is large and complex with two primary functions: data collection and prediction. It is one of several similar DoD systems, each of which has a primary focus on some subset of the relevant data and is presently stand alone. The domain is critical to DoD operations.

The program to develop this system is executing in conjunction with a number of other like programs, all of which are either modernizing existing systems or creating replacement systems. Taken together, these systems are excellent candidates to be interoperable components in a very large system of systems, which would permit the separate subsets of data to be viewed, studied, and analyzed from a single, global perspective. To that end, a high-level decision has been made for each system to be part of such a system of systems.

The ownership issue we observed was that of authority, or rather, the lack of authority, to ensure the interoperability relationships between the various systems. Each of the constituent systems is a separate acquisition, falls under a different part of the DoD, has a separate funding stream, and follows a well-defined schedule and budget for a stand-alone system. There is no apparent provision, in any of the separate acquisitions, for the planning and cooperation activities; nor is there any indication that defining the technical requirements of interoperability among several systems is part of the work effort. The high-level decision—that is, that each system should cooperate—was not accompanied by any allocation of funds,

compromise of schedules, or ongoing oversight to ensure that the decision would be implemented.

Using our terminology, no one has either taken ownership of or applied resources to the **relationships between** the various systems. Hence, each separate program manager is primarily concerned with his or her system's individual success; lacking any other incentive, this the only logical course for each manager. The success of the system of systems is largely an ideal, without any tangible steps being taken to reach it. As of the time of our study, it appears likely that this condition will be ongoing.

3.3.2 System B: Operations and Daily Maintenance

The second example consisted of two systems, both of which had been successfully operating for some time. Both systems operated in the financial domain and in combination processed vouchers, payments, and other financial transactions. System A was owned and operated exclusively by one of the armed services; System B was owned by a different government agency and provided its capabilities to many systems throughout the government. Both System A and System B were developed using technology of the 1970s and 1980s (COBOL and assembler code, batch processing, file-based structures, etc.) and were becoming increasingly difficult to maintain. The armed service decided to modernize its system (System A), using a relational database and providing real-time transaction processing to replace existing labor-intensive processes.

In creating the replacement for System A, the builders assumed that, since System B already performed its critical functions successfully, the new System A would interoperate with System B just as the old System A had. However, this assumption was deeply flawed: linking real-time transaction processing with batch processing led to significant errors, huge transaction delays, and erroneous behavior from both systems individually and in combination.

This kind of technical mismatch is not uncommon, and many similar technical lessons have been learned and documented. However, the major lesson from the ownership perspective lay in how difficult it was to remedy the mismatch. The two systems were owned by different agencies of the DoD. System B served several clients and was required by law to accommodate all ongoing updates to the tax code; these updates were to take precedence over all other maintenance. System A was responsible for a major financial process within one of the armed services, and the failing interoperation between the systems led to large-scale defaults on those financial processes.

Complicating this was that most attempts to repair the mismatch were not undertaken jointly but independently by persons from the two systems.[5] The resources available from System B's staff were applied towards fixing the mismatch, with the result that all other corrective

[5] There were some joint attempts to correct data entries within the two systems, but no joint attempts to correct code deficiencies were made.

maintenance of System B was abandoned; with the resultant loss of configuration control, defect fixes were not made. The resources from System A's staff were applied largely to perfective work on System A alone; their belief was that all data sent to the older system was consistent with the agreed interfaces and that the problems lay in the older system.

The above is an example of the issue raised in Section 2.4: that the needs of the individual systems and the needs of the system of systems can be in conflict. This example also highlights the fact that interoperability is more than a purely technical matter. There was, in effect, no relationship to promote interoperability between the owners of the two systems, which proved as critical as the technical mismatch.

3.3.3 System C: Evolution and Long-Term Maintenance

The third example is a large and complex system of systems that has been in operation for several years. It relies on an infrastructure component that is itself highly complex; and the architecture of the overall system consists primarily of interfaces between this infrastructure and all of the constituent systems. The infrastructure thus is the interoperability "engine" that unites the separate heterogeneous systems to perform certain DoD processes.

The constituent systems are under a wide variety of ownerships and authority, including all of the services and several other DoD agencies. Maintaining the infrastructure component and integrating the various application systems with it is the responsibility of one element of one of the armed services.

The infrastructure and its interfaces with the rest of the system are based on very old technology. There are redundant data stores, and the interfaces are inconsistent between the various applications and the infrastructure. Perhaps the greatest weakness is that the major processing task of the overall system of systems (gathering data from a large number of sources, analyzing the data, and then providing support for military operations) takes more than a day. Further, a limitation of the infrastructure dictates that tasks can only be done one at a time: the system cannot perform multiple analysis tasks simultaneously.

We observed this system when a project was underway to replace the infrastructure with a more modern, Web-based component. The new infrastructure component remedied the data redundancy and also provided the capability for more consistent interfaces with the other systems. Most importantly, the new component enabled multitasking.

Several ownership issues arose because the modernization project was not being executed by the service agency that developed the original infrastructure and that still had authority over the overall system of systems. Instead, the agency sponsoring the modernization was a strong proponent of agile acquisition and was intent on radically shortening system development times. But that agency only had authority over the short-term upgrade of the infrastructure; it did not have other authority (e.g., for integrating the upgraded infrastructure into the system of systems). The two agencies have repeatedly disagreed on issues: appropriate testing,

clarity of requirements, and which activities should be included in development versus integration.

Another ownership issue derives from the technical nature of the modernization, since many of the interfaces supported by the old infrastructure will not be supported by the new one. Given that the interfaces to a large number of the application systems will therefore be broken, there are several unknowns about bringing the system of systems back into operational condition: whose responsibility will it be to restore the broken interfaces, whose resources will be expended, and what is the detailed plan to bring this about.

Still another issue is contractual: one agency has responsibility over the software company modernizing the infrastructure and the other agency has responsibility over the company that is maintaining the overall system integration. This implies that virtually any authority and decision-making is distributed among four entities, two of which have displayed a marked lack of cooperation.

This program exemplifies the issue raised in Section 3.2, concerning the need for a large-scale shift in how the various services and agencies presently regard ownership and authority over their systems. At the moment, there appears to be little incentive for either of the sparring agencies described here to abandon their positions. Yet, unless the participants find a path toward truly shared authority and joint possession, this kind of project, whose technical dimension typifies many NCW modernizations, has little hope of success.

4 Some Practical Steps to be Taken

". . . pointing the way to encouraging possibilities of compromise and cooperation which, if bilaterally implemented with appropriate give and take on both sides might, if the climate were right, have a reasonable possibility at the end of the day, of leading, rightly or wrongly, to a mutually satisfactory resolution."— "Yes, Prime Minister", Jay & Linn

In this section, we suggest some steps that can be taken throughout the DoD to mitigate the risk that the problems detailed in section 3 will be repeated. These steps are not inclusive but are rather representative of the changes that must be brought about.

4.1 Education

There is an immediate need for a new mindset, focusing on organizational and programmatic issues, at all levels and in every part of the DoD. To date, the majority of the discussion of NCW has been at the technical level, and most of the scenarios (including the ones cited herein) have stressed how the technical aspects of the systems will interconnect. However, the "new mindset" we refer to must focus on other needs—in particular, the different types of ownership that the NCW world will necessitate. At the very least, experts from all services must begin to discuss how the current structure of single-service authority over systems can evolve toward the demands of a truly net-centric operational posture.

Closely related to this is the need for greater understanding about the reality of unfunded mandates: as long as directives that can never be met (because no resources are applied to following them) continue to be issued, the overall DoD culture is unlikely to change.

One final area requiring greater understanding and education is the increased need for disclosure. In defining interoperable relationships between systems, the owner of each individual system will inevitably face the need for some, and perhaps a great deal of, disclosure about that system. Systems of systems require significant disclosure of semantic details to enable their interoperation. One organization working toward understanding how this kind of climate can be achieved is the Defense Modeling and Simulation Organization, which is working to enable dynamically composable models in a service-oriented architecture. The constituent parts of a composed model require extensive disclosure of what is considered privately-held intellectual property. This has raised many additional questions of confidentiality, compensation, and exposure to competitors.

4.2 Web Services Solutions

In the commercial sphere, and particularly in the community working on Web services, considerable work is being done in the area of Web-based collaboration. Two of the promising concepts are *orchestration* and *choreography*. These terms relate to different ways

of connecting Web services, with the assumption that different connections may need different notions of ownership, or at least authority. Orchestration reflects a single-owner notion, while chorography reflects a joint-owner relationship:

> *Orchestration* refers to an executable business process that may interact with both internal and external Web services. Orchestration describes how Web services can interact at the message level, including the business logic and execution order of the interactions. These interactions may span applications and/or organizations, and result in a long-lived, transactional process. With orchestration, the process is always controlled from the perspective of one of the business parties.
>
> *Choreography* is more collaborative in nature, where each party involved in the process describes the part it plays in the interaction. Choreography tracks the sequence of messages that may involve multiple parties and multiple sources. It is associated with the public message exchanges that occur between multiple Web services. Orchestration differs from choreography in that it describes a process flow between services, controlled by a single party. More collaborative in nature, choreography tracks the sequence of messages involving multiple parties, where no one party truly "owns" the conversation [Peltz 03].

Standards and protocols are being developed to support these concepts and may have significant value to NCW.

4.3 Need for New Incentives

We have argued that success in realizing the vision of system-of-systems interoperability on a large scale will require a fundamental realignment of ownership perspectives. Historically, the sense of responsibility for a system is proportional to owning that system, particularly in the degree of authority one has over the system. For the emerging vision of NCW, however, a new and perhaps uncomfortable style of ownership must evolve, one that has two novel characteristics. First, in the context of the systems of systems, some of which may be unbounded, it may even be inappropriate to speak of "ownership" at all; it may be impossible to determine who owns (in the familiar sense) the aggregate system of systems. Second, for the individual systems (that will still have owners), the balance among ownership, authority, and responsibility must change. It may be that, for a given system, the sense of ownership requires a high sense of responsibility but a lessened degree of individual authority: the needs of the whole may demand that the individual system owner act in ways that benefit the system of systems even if such actions penalize the system that he owns.

This realignment will require creative incentives to foster a more global perspective; in effect, a large community will need to "think globally while acting locally."[6] Two examples show the need for this perspective in practice. The first is taken from *Power to the Edge*:

> Given the significant advances in technology, the primary barriers that remain are cultural and institutional. Finding ways to remove these impediments to progress is on the critical path to transformation. Education alone will not be sufficient. The reward and value structures need to change in order to establish cultural and institutional norms that (at the least) permit exploration of power to the edge principles (for example, desirable attitudes and behaviors about sharing information, collaboration, loyalty, and relationships within an organization and among organizations). There must be proper incentives associated with these desirable behaviors [Alberts 03].

The second is from the 9-11 Commission's final report:

> Recalling the Goldwater-Nichols legislation of 1986, Secretary Rumsfeld reminded us that to achieve better joint capability, each of the armed services had to "give up some of their turf and authorities and prerogatives." Today, he said, the executive branch is "stove-piped much like the four services were nearly 20 years ago." He wondered if it might be appropriate to ask agencies to "give up some of their existing turf and authority in exchange for a stronger, faster, more efficient government wide joint effort." Privately, other key officials have made the same point to us [NCTAUS 04].

And from the same report

> [Federal Bureau of Investigation] field offices other than the specified office of origin were often reluctant to spend much energy on matters over which they had no control and for which they received no credit [NCTAUS 04].

All of the ownership issues described in this technical note exhibit this dilemma. It is inevitable, given the historical attitudes of authority and ownership, that the logical first thought of an individual system owner is this: If there are some interoperability demands levied on my system which mean that I must relinquish control but receive no ownership or authority over the global benefit, relationships, emergent properties, or any of the other things, then why should I cooperate? The logical answer is "For the good of the overall community." But this response assumes far more altruism than is often found in practice.

Deeper and more creative incentives must be devised that will bring the reality of personal value (whether in the form of compensation, success in career path, or other comparable

---

[6] This phrase refers to the argument that **global** environmental problems can turn into action only by considering ecological, economic, and cultural differences of our **local** surroundings. This phrase was originated by Rene Dubos as an advisor to the United Nations Conference on the Human Environment in 1972.

rewards) into alignment with the new ownership paradigm. Collopy has proposed one such mechanism in postulating win-win contracting mechanisms for the Joint Strike Fighter.

> Value misalignment is an endemic problem in the current spec-flowdown system, where every part has performance specifications and a cost target. For example
>
> > Engineer A's part is over spec weight but under target cost. His best choice is to change to a lighter weight, more expensive material. Cost increases $5,000 and weight is reduced by 8 pounds. The design is now satisfactory. Engineer B's part is over cost but under on weight. A cheaper, heavier material increases weight 28 pounds, but reduces cost $2,000, meeting all goals. The net effect of both decisions on the system [however] is +20 lbs. and +$3,000.
>
> It is impossible for this problem to occur when both engineers are maximizing [global value ...]. Thus, design engineers can evaluate every design decision according to what maximizes [global value] for the overall system, that is, what provides the greatest collective benefit to the government and the contractor [Collopy 98].

Collopy then goes on to construct a global optimization model that simultaneously maximizes government benefit and contractor profit. As the majority of today's system-of-systems evolution must occur through an acquisition mechanism, these types of contractual incentives hold promise to stimulate global thinking in our local actions.

5 Summary

In this technical note, we have examined a number of ways in which different modes of ownership can affect the realization and maintenance of large, heterogeneous systems of systems. As with many other pertinent attributes of interoperability (e.g., evolution or semantics), these are not isolated concepts; rather, they jointly contribute to the complexity and difficulty of creating ubiquitous, dynamic, ad hoc relationships between systems.

This note presents four key arguments. First, ownership of any computer system is more complex than simple possession; in particular, ownership of a software-intensive system will likely have a certain ambiguity. Second, ownership in the context of interoperable systems implies some understanding and agreement, by the individual system owners, about authority and relationships between systems. Third, to achieve the hoped-for ideal of NCW, a considerable change is needed in the programmatic and managerial domain of DoD acquisition; a central aspect of this change involves the incentives to relinquish and share ownership and authority. Fourth, the difficulty of bringing about this change is at least comparable to any technological aspects of system-of-system interoperability.

Appendix A Definitions and Characteristics of Ownership

We discuss what *ownership* means in a variety of contexts. We first consider the concept in its most general sense and then focus on the various understandings of ownership in the context of computer systems and software.

Common Concepts of Ownership

The nature of ownership is complex. A straightforward definition is: "legal right to the possession of a thing" [HM 82]. However, the simplicity of that definition is misleading. Many serious discussions of ownership, dating back several centuries, point out the tangled distinctions between possession, property, and rights—as well as the interrelationships among the legal, philosophical, political, and economic dimensions of ownership. We shall cite some of these below. Some analyses of ownership seek to arrive at precise definitions of the key terms relevant to ownership (e.g., "attendant rights to property"). However, in much contemporary literature, particularly documents relating to information technology and business process management, the term *ownership* is used in a much looser fashion, favoring such phrases as "the users need to own the process" and "we must foster a sense of ownership of the system." In this current-day use of *ownership*, the goal is simply to denote shared responsibility, without making hard-and-fast distinctions about rights and possession of the sort described earlier.

Some Representative Descriptions

Simply to indicate the breadth of writing on the topic, we offer the following samples. First, from a report on privatization of community water systems:

> The goal of our analysis is to construct a decision-making framework for evaluating water system ownership and management alternatives [which] requires an understanding of economic, policy, social, legal, and environmental considerations [Cowan 04].

From Justice Oliver Wendell Holmes, often cited to justify a particular understanding of economic rights:

> But what are the rights of ownership? They are substantially the same as those incident to possession. Within the limits prescribed by policy, the owner is allowed to exercise his natural powers over the subject-matter uninterfered with . . . [Holmes 00].

From John Locke's *Second Treatise on Government*:

> Every man owns his own body. Nobody else has a right to one's own body. One also has a right to one's own labor and work. Whatever is unowned in the state of nature and is worked on by a person becomes the property of that person. Then no one else has a right to it [Locke 02].

And from an analysis of rights to genetic material:

> *[Refuting Locke's position that a man owns his own body:]* Why should we accept this? It is far from self-evident. Why should we not assume, rather, that nobody (not even the person in question) can own a person or the person's body? From a purely intuitive point of view, this is surely at least equally plausible. It would also explain the widespread conviction that no one is entitled to sell or give away himself [Bergstrom 00].

Common Elements of the Definitions

Throughout these, and many other, discussions of ownership, several elements reappear. Among these elements are possession and property, authority over what is owned, and the issue of roles and responsibilities regarding what is owned; all of these somehow involve the relation between ownership and rights. Below we offer a brief discussion of these elements. Each has some bearing for our consideration of ownership of interoperable software systems in Section 3.

Property and Possession—It is commonly thought that what is owned is one's property, whether tangible or intellectual; that is certainly the basis for the simple dictionary definition of ownership cited at the start of this section. Defining ownership in terms of how rights may be exercised over one's own property is the basis of many legal and economic discussions of ownership. Thus, Bergstrom notes a common view that

> More specifically, ownership is defined as residual control rights to assets, that is, the right to determine the uses of assets [Bergstrom 00].

In addition, Holmes, quoted above, holds that ". . . rights of ownership [are] incident to possession. . . ." Possession is most easily demonstrated in the case of tangible property. However, in the case of intellectual property (which is critical for software), some questions arise. Most descriptions of intellectual property regard it as similar to, but not the same as, tangible property. The two are quite similar, however, in that the question of rights is no different for tangible or intellectual property. The following is from a U.S. government definition:

> [There are] four separate and distinct types of intangible property—namely, patents, trademarks, copyrights, and trade secrets, which collectively are referred to as "intellectual property." Intellectual property shares many of the characteristics associated with real and personal property. For example, intellectual property is an asset, and as such it can be bought, sold, licensed,

> exchanged, or gratuitously given away like any other form of property. Further, the intellectual property owner has the right to prevent the unauthorized use or sale of the property. The most noticeable difference between intellectual property and other forms of property, however, is that intellectual property is intangible, that is, it cannot be defined or identified by its own physical parameters. It must be expressed in some discernible way to be protectable [Hefter 05].

Authority—Authority is an aspect of ownership that is most closely related to rights. A typical definition is

> The power to command, enforce laws, exact obedience . . . freedom or right granted to another [HM 82].

Given the condition of ownership, some authority is typically either resident or conferred. But authority is not absolute; it exists in gradations and hierarchies, and is thereby a variable aspect of ownership. For instance, one might own a house, but not have complete authority over who may reside there (e.g., there may be a zoning stipulation that no more than two unmarried adults may inhabit a house in a certain area). A document describing the city of Québec's authority over vehicular traffic states the following:

> In keeping with its jurisdiction over ownership and civil rights, [the city government] determines restrictions on the right to travel, traffic rules and vehicle ownership standards. Generally speaking, Québec alone is empowered to legislate with regard to road safety [Québec 99].

Thus, owning and possessing an automobile does not automatically convey the right to use it in any way one wishes; only certain rights are granted to the owner. And here, we see that the limitation on an owner's **authority** modifies the earlier **property**-based definition of ownership (i.e., "ownership [is] the right to determine the use of assets").

Roles and Responsibilities—Possession and authority both reside in some person or agency. But there may be a division of roles and responsibilities within that agency. With computer systems, there are usually a number of interested parties, called *stakeholders*, for whom different modes of ownership exist, at least notionally. Thus, users of a particular computer system may not be its legal owners, but their enterprise and its success may be entirely dependent on that system. Those stakeholders will likely have at least some sense of ownership over the system.

This suggests the need to consider a number of roles that refine the notion of ownership. For instance, the following is from a discussion of ownership of software data rights:

> [The speaker made] a distinction between Data Ownership and Data Stewardship. Ownership suggests that an individual is empowered to make

decisions alone, and that he can act to address the needs of his business unit. Owners design and control their own processes.

Stewardship, on the other hand, connotes a facilitation role. A steward uses a consistent, repeatable process to achieve alignment across the organization, and the needs of all areas are considered when making decisions.

A third role is that of Custodian. The Custodian is responsible for the physical security of the data. It is a role often played by IT people who work closely with Business Data Owners. Custodians administer the password access systems and provide backup and disaster recovery capabilities. These are people who create and enforce data standards and implement the physical data architecture [Schlenker 03].

Before examining the matter of ownership and rights in greater detail, we first must consider some issues specific to ownership of computer systems.

Ownership of Computer Systems

There are varying notions of what is meant by *computer system*. We will use the term to mean the combination of four things:

1. users and other stakeholders of the system
2. hardware
3. business processes that are employed
4. software

Thus, ownership of "a computer system" aggregates hardware, software, and processes, and is apt to be divided among a number of parties. We shall consider each of these elements separately. Of the four, the first two—users and other stakeholders, and hardware—are probably the easiest to consider.

The first element presents little difficulty, because users and stakeholders are not themselves "owned" but are the owners, in some manner, of the other three elements—processes, hardware, and software. In the case of a single system, the human element may not be a complex matter. But given that ownership connotes authority, the human element is of paramount importance when we consider the many different kinds of interoperability relationships **between** systems (see Section 3 for a fuller consideration of this point).

Ownership of the hardware is straightforward, too, since hardware is physical and may be treated as tangible property. For that reason, the discussions of tangible property in the preceding sections are pertinent and applicable to the ownership of hardware. Thus, to the extent that ownership of any tangible property can be reasonably well defined, those concepts apply to hardware as well. The simplest view is that the owners of hardware are, by and large, the persons or agencies that have purchased it.

Ownership of Business Processes

When we consider ownership of processes, we enter the realm of intangibles, which, while no less real than physical property, may alter how we apply the concept of "ownership". One of the first questions to arise is this: precisely **what** of a process can be owned? One view is that processes are things that are designed and then executed, each with corresponding owners:

> A simple view is that a Process Owner owns the *design* of the process, whilst a Process Manager owns the *operation* of the process. . . . The Process Owner may (or may not) own the overall operation of the process and thus may be responsible for ensuring that the Process Managers are capable of managing the process and also do so in practice [Syque 00].

This view makes no mention of the process **users**, who are certainly significant stakeholders and perhaps owners of the process in some sense; there are also many other views on process ownership. Nor does the above statement bring any clarity on precisely what owning a process really means. So we see that, notwithstanding the intangibility of a process, the complexity of its ownership can rival or exceed that of ownership of physical property.

When we consider business processes in the context of computer systems, we then realize that there is an intimate relationship between the processes and the software that supports and enacts them. The software is present to assist people in carrying out their business and the software thus reflects (or should reflect) the practices that the people employ. But often the software will dictate the business practices: if the software performs a necessary business function in a particular way, it thereby defines the practices the people must use.

Is it therefore the case that the process owner (whoever that may be) is logically the same as the software owner? This is seldom true, particularly when the software is acquired from external or commercial sources. We can, in fact, see some sense of process ownership actually devolving to the designers and developers of the software. In fact, the more the software dictates the business practices, the greater control (and sense of ownership) the developers and acquirers have over the system. Conversely, the more the software is configurable by the operators, the greater control and sense of ownership the operators will have over the system.

Ownership of Software

Software is ephemeral; in one sense, at least, it has no existence outside of its operation in the context of a system. Indeed, it is not obvious what we should consider to be software: is it the source code, the compiled executable, or both? Is it the design or the architecture? The algorithms? All these and more have been conjectured to be elements of software that can be owned.

Software thus offers a dilemma for such definitions of ownership as that made by Oliver Wendell Holmes (". . . what are the rights of ownership? They are substantially the same as

those incident to possession."), since possession of software is not always a well-defined thing. For instance, many software companies sell executables to their clients, who, it would seem, thereby possess them. But those clients rarely own the source code that produced the executable. And it may not even be clear who owns the executable. Many legal rulings, for instance, have been made about "software" that do not distinguish between source code and binary executable code. Thus, in a ruling about taxation of corporate property, the state of Louisiana ruled that

> Company A's software . . . is comprised [sic] of prewritten modules, or programs, that are combined and installed to properly integrate the communications system. . . . Ownership of [Company A's] software does not pass to Company B. Instead, the company is granted a perpetual software license agreement, which allows Company B to use, but not own, the software [Louisiana 04].

As can be seen, the ruling states that ownership of Company A's software (which presumably meant the binary executables) does not pass to Company B despite the fact that Company B purchased it and owns the overall system in which the modules operate (and which did include the executables).

Software Ownership and Software Rights

With ownership of software, we move a good distance away from the notion of ownership in the sense of simple possession. Software lacks most attributes of tangible property (though not all, as we shall see below), and the major consideration for software is that its intellectual property is the critical aspect of its ownership. Deciding who owns software involves some rearrangement of all the ownership aspects we have discussed: possession, authority, role, and rights. Hence, understanding the ownership of a software component means first understanding who holds its **intellectual** property, then defining in what sense the various purchasers can be said to possess the software, and finally determining what authority and rights are granted to the purchasers. (Indeed, we have already noted how software ownership has become an issue for the judiciary; that is, the issue of taxation of parts of a software system). In the case of a single software component, there may be many types of license agreements available, and hence many possible modes of ownership. In the case of a large system with numerous software components, the variability of ownership modes will be quite complex.

Rosen describes the intellectual property aspect of software as follows:

> Software is a product of human intellect, and therefore it is a kind of intellectual property. Intellectual property is a valuable property interest, and the law allows its owner to possess and control it. The programmer who writes software—or the company that hires that person to write software—is deemed to be the first owner of intellectual property embodied in that

> software. That owner may exercise dominion over that intellectual property. He can give it away, sell it, or license others to use it. That owner has the prerogative to create copies of the intellectual property, and he or she may prevent others from making, using, or selling those copies. Because of these partly tangible and partly intangible aspects of computer software, it is possible to have different owners own (1) a tangible copy of software purchased at a computer store or downloaded from a website, and (2) the intellectual property embodied in that software [Rosen 04].

In most cases, the owner with the tangible copy in the Rosen's example owns only the right to use that software and seldom owns the software itself. And unlike most tangible property, where possession generally implies ownership by one party at a time (e.g., in the case of real estate), software may be replicated at little or no cost. Multiple instances of a software component are essentially equal yet owned differently by many persons simultaneously.

The issues of software ownership and rights have not been fully resolved and are in need of careful consideration on all sides. And although we have only touched the surface of the question of software ownership, even this brief examination suggests, whatever else may be true, there is a broad spectrum of different ways that software can be owned. Software can be viewed as property—that is, intellectual property—and thus be owned by its creators or vendors. Software can be viewed as something that people (i.e., purchasers, integrators, or users) have the right to use. When many different software elements jointly participate in large-scale systems of systems, these different nuances of ownership will unquestionably affect the manner of their interoperation.

Before leaving the matter of software ownership, it may be useful to posit an analogy. We consider that software may be analogous to music. Both are ephemeral in that they really exist only when they are "performed." Both are notated in representations that must be brought into actuality through some sort of execution. Thus, we have software source code which may be likened to the score for a given piece of music. The operation of the software may be likened to a given performance of the piece.

The value of this analogy is that for music as for software, the question of ownership is complex and currently undergoing considerable legal debate. Just as we can speculate on what it might mean to own software, we can also speculate on what it means to own a piece of music. Most clearly, we consider the owner to be the copyright holder. And yet, by selling the musical score commercially, the copyright holder grants certain rights to others. Musicians can purchase the score, which gives them the right to perform the piece in public; the **performer** retains certain rights to that specific performance (e.g., on a commercial recording) that are distinct from the rights of the composer. The analogy with software is not far-fetched: the owner of a piece of software (e.g., a COTS vendor) may sell or lease the right to execute ("perform") that software to others. And, indeed, those with the right to execute

the software have some sense of ownership over it, but it is a limited ownership and will likely be constrained by agreements with the vendor.

Whatever else may be true, ownership of software cannot be treated in the same way as ownership of physical things, but rather, as a matter of rights: We typically own the right only to use the software (analogous to buying and thus owning a musical score with the right to perform it in public). We may own the right to execute the software but not to duplicate it (analogous to owning a recording of a performance but not the right to share it through Napster or some other file-sharing medium). We may own the physical medium embodying the software, but not the software itself (analogous to owning a copy of a score, but not the copyright for the music written on the score).

References

URLs are valid as of the publication date of this document.

[Alberts 99] Alberts, D.; Gartska, J.; & Stein, F. *Network Centric Warfare.* Washington, D.C.: Center for Advanced Concepts and Technology, 1999.

[Alberts 03] Alberts, D. & Hayes, R. *Power to the Edge.* Washington, D.C.: Center for Advanced Concepts and Technology, 2003.

[Bergstrom 00] Bergstrom, L. *The Concept of Ownership.* http://www.philosophy.su.se/texter/ownership.htm (2000).

[Branscomb 91] Branscomb, L., et al. *Intellectual Property Issues in Software.* Washington D.C.: National Academy Press, 1991.

[Brownsword 04] Brownsword, L., et al. *Current Perspectives on Interoperability* (CMU/SEI-2004-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tr009.html

[Carney 05] Carney, D.; Fisher, D.; & Place, P. *Topics in Interoperability: System-of-Systems Evolution* (CMU/SEI-2005-TN-002). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tn002.html

[Collopy 98] Collopy, P. *Joint Strike Fighter: Optimal Design through Contract Incentives.* http://www.dfmconsulting.com/jsf.htm (1998).

[Cowan 04] Cowan, C.; Mescher, A.; Miller, J.; Pettway, K.; & Pink, B. *A Framework for Evaluating Water System Ownership and Management Alternatives.* Donald Bren School of Environmental Science, Project Proposal. Santa Barbara, CA: University of California at Santa Barbara, 2004.

[Hefter 05] Hefter, L. & Litowitz, R. *What Is Intellectual Property?* http://usinfo.state.gov/products/pubs/intelprp/homepage.htm (2005).

[HM 82] Houghton Mifflin Company. *The American Heritage Dictionary of the English Language*, 2nd edition, Boston, MA: 1982.

[Holmes 00] Holmes, O.W. *The Common Law.* http://biotech.law.lsu.edu/Books/Holmes/claw_c.htm (2000). Presented as The Lowell Lectures, Harvard University, Cambridge, MA, 1881.

[Lewin 78] Lewin, R. *Ultra Goes to War: The Secret Story.* New York: McGraw-Hill, 1978.

[Locke 02] Locke, John. *The Second Treatise of Civil Government.* http://www.constitution.org/jl/2ndtreat.htm (2002)

[Louisiana 04] Loftus, C. *Private Letter Ruling 04-006.* New Orleans, LA: Louisiana Dept. of Revenue, 2004.

[Peltz 03] Peltz, C. "Web Services Orchestration and Choreography." *Computer, 36,* 10 (October 2003), 46–52.

[Québec 99] Government of Québec. *Sharing Jurisdiction over Transportation.* http://www1.mtq.gouv.qc.ca/en/services/documentation/lois/partage.asp (1999).

[Rosen 04] Rosen, L. *Open Source Licensing: Software Freedom and Intellectual Property Law.* Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[Schlenker 03] Schlenker, S. "IDMA Focuses on Data Ownership." *The Data Administration Newsletter.* http://www.tdan.com/i025ht04.htm (2003).

[Syque 00] *The Improvement Encyclopedia.* http://syque.com/improvement/Process%20Ownership.htm (2000).

[NCTAUS 04] National Commission on Terrorist Attacks Upon the United States *The 9/11 Commission Report: Final Report on the National Commission on Terrorist Attacks Upon the United States.* Official Government Edition, New York: Norton, 2004.

[Winterbotham 74] Winterbotham, F. *The Ultra Secret.* London: Weidenfeld and Nicolson, 1974.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	November 2005	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Topics in Interoperability: Concepts of Ownership and Their Significance in Systems of Systems	FA8721-05-C-0003

6. AUTHOR(S)

David Carney, William Anderson, Patrick Place

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2005-TN-046

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note is a brief examination of the concept of ownership and the ways in which it might apply to systems of systems. It first analyzes ownership itself from a number of perspectives and then describes how ownership is generally understood in the context of computer systems. Next, the note outlines some implications that different notions of ownership will pose for large-scale, complex systems of systems, particularly such systems as those envisioned in network-centric warfare. The note describes several real-world examples of ownership issues that exist in existing systems of systems and posits some areas in which research is necessary.

14. SUBJECT TERMS	15. NUMBER OF PAGES
interoperability, systems of systems, ownership, system of systems	42

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL