A Campus Situational Awareness and Emergency Response Management System Architecture

by

Amine Chigani

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Osman Balci, Chair

James D. Arthur
Reza Barkhi
Steven D. Sheetz
Eli Tilevich

April 6, 2010
Blacksburg, VA

Keywords: Architecting, Capabilities Specification, Campus Safety, Emergency Response Management, SOA Security

A Campus Situational Awareness and Emergency Response Management System Architecture

by

Amine Chigani

ABSTRACT

The history of university, college, and high school campuses is eventful with man-made tragedies ensuing a tremendous loss of life. Virginia Tech's April 16 shooting ignited the discussion about balancing openness and safety in open campus environments. Existing campus safety solutions are characterized by addressing bits and pieces of the problem. The perfect example is the recent influx in demand for Electronic Notification Systems (ENS) by many educational institutions following the tragedies at Virginia Tech and Northern Illinois University. Installing such systems is important, as it is an essential part of an overall solution. However, without a comprehensive, innovative understanding of the requirements for an institution-wide solution that enables effective security control and efficient emergency response, the proposed solutions will always fall short.

This dissertation describes an architecture for SINERGY (campu**S** s**I**tuational aware**N**ess and **E**mergency **R**esponse mana**G**ement s**Y**stem) – a Service-Oriented Architecture (SOA)-based network-centric system of systems that provides a comprehensive, institution-wide, software-based solution for balancing safety and openness on any campus environment. SINERGY architecture addresses three main capabilities: Situational awareness (SA), security control (SC), and emergency response management (ERM). A safe and open campus environment can be realized through the development of a network-centric system that enables the creation of a COP of the campus environment shared by all campus entities. Having a COP of what goes on campus at any point in time is key to enabling effective SC measures to be put in place. Finally, common SA and effective SC lay the foundation for an efficient and successful ERM in the case of a man-made tragedy.

Because this research employs service orientation principles to architect SINERGY, this dissertation also addresses a critical area of research with regards to SOA; that area is SOA security. Security has become

a critical concern when it comes to SOA-based network-centric systems of systems due the nature of business practices today, which emphasize dynamic sharing of information and services among independent partners. As a result, the line between internal and external organization networks and services has been blurred making it difficult to assess the security quality of SOA environments. In order to do this evaluation effectively, a hierarchy of security indicators is developed. The proposed hierarchy is incorporated in a well-established evaluation methodology to provide a structured approach for assessing the security of an SOA-based network-centric system of systems.

Another area of focus in this dissertation is the architecting process. With the advent of potent network technology, software/system engineering has evolved from a traditional platform-centric focus into a network-centric paradigm where the "system of systems" perspective has been the norm. Under this paradigm, architecting has become a critical process in the life cycle of software/system engineering. The need for a structured description of the architecting process is undeniable. This dissertation fulfills that need and provides a structured description of the process of architecting a software-based network-centric system of systems. The architecting process is described using a set of goals that are specific to architecting, and the associated specific practices that enable the realization of these goals. The architecting process description presented herein is intended to guide the software/system architects.

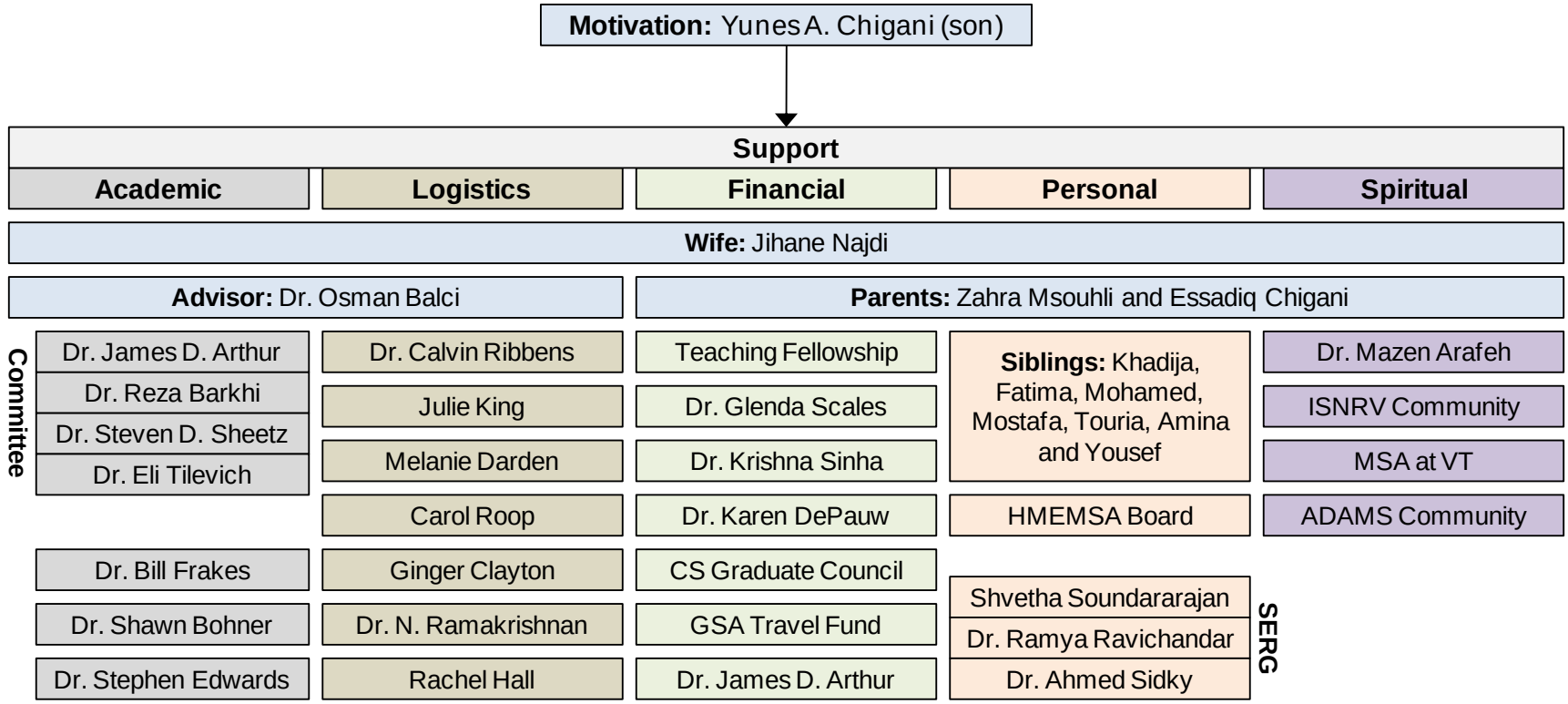
DEDICATION

To my dear mother, Mrs. Zahra Msouhli

إلى أمي العزيزة، زهرة مسوحلي

ACKNOWLEDGMENTS

The only way to do justice to this section of my dissertation is to write a book about it, which I hope I will do one day. Instead, I would like to admit that all the years and hard work I contributed to reach this milestone in my life does not compare to the support I received on the way here. Therefore, for the fear of using words that are unfair to or not enough for all those who must be acknowledged here, I use a layered architecture to represent the support I received during my tenure at Virginia Tech. As for the details of design and code, I will share them in person (and possibly in a book one day).



Architectural representation of my acknowledgment of all who supported my tenure at Virginia Tech

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: PROBLEM DEFINITION AND OVERVIEW

1.1. PROBLEM DEFINITION

In the wake of the April 16 tragedy, several issues were identified relating to the ability of educational institutions operating in open campus environments to protect their constituents while maintaining a balance between safety and openness. From a technology perspective, the available solutions (e.g., electronic notification systems (ENS)) provide only bits and pieces of the solution. While many legal and social aspects of this issue need to be taken into consideration, the problem must be addressed primarily by involving all campus constituents including students, staff, administrators, law enforcement personnel, and first responders, along with existing software-based systems used to facilitate situational awareness (SA), security control (SC), and emergency response management (ERM) on campuses.

When one navigates the literature related to campus safety and security, the obvious conclusion is the diversity of expertise that enables addressing this issue from various perspectives. For instance, the field of SA is rich with theoretical and practical solutions to achieving a common operating picture (COP) in both closed and open environments [Endsley 2000]. Moreover, current surveillance technologies can provide ways for real-time situation monitoring in various contexts. In addition, there is concrete practical and theoretical work in the area of ERM in the case of disasters [Perry and Wolf 2003] that provides guidance for managing teams and resources during an emergency. More importantly, from a software engineering viewpoint, service orientation has given rise to the connectedness of more remote and heterogeneous communities of interest than ever before [Erl 2008; Rosen *et al.* 2008; Friedman 2005]. All this exists while there is a lack of a comprehensive approach that leverages these capabilities to provide a network-centric solution for SA, SC, and ERM in an open campus environment. To address this problem, we view it from three different prisms and propose a solution that addresses all facets of the problem in a comprehensive manner.

1.2. RESEARCH OBJECTIVES

The goal of this dissertation is to contribute to three areas of research. The main area of contribution is in investigating best practices and technologies related to Web Services-based Service-Oriented Architecture (SOA), and developing a network-centric architecture based on these practices and technologies for SINERGY (campu**S** s**I**tuational aware**N**ess and **E**mergency **R**esponse mana**G**ement s**Y**stem)—a system that provides SA, SC, and ERM capabilities on any open campus environment. Second, we contribute a hierarchy of indicators to assess security in SOA-based network-centric systems of systems, and use this

hierarchy in the evaluation of the security quality characteristic of our proposed architecture. Finally, we contribute a definition and description of the process of architecting for software/system engineering.

Prior research experience in network-centric systems, software architectures, and quality assessment served as a foundation for conducting this research work [Chigani 2007; Chigani and Arthur 2007; Chigani *et al.* 2007]. In addition, access to Virginia Tech's resources (students, staff, and systems) served as a context for a viewpoint-oriented capabilities specification. Most importantly, experience with large-scale architecture development using the Department of Defense Architecture Framework (DoDAF) [DoDAF 2009a,b,c] was used to develop and document the proposed architecture. Finally, the Military System Architecture Assessment Methodology (MSAAM) [Balci and Ormsby 2008] is employed in validating the usefulness of our proposed architecture.

1.3. RESEARCH IMPORTANCE

The impact of this research is threefold. First, the hierarchy of security indicators proposed and its integration into the Evaluation Environment (EE) methodology [Balci 2001; Orca 2002] will bring structure to the assessment of security quality for any SOA-based network-centric system of systems. Second, the Capability Maturity Model Integration (CMMI) for Development defines process areas such as product integration, project planning, requirements development, risk management, and others [SEI 2010a]. Yet, it does not define an architecting process, which has become a key aspect of project management and product quality for software development efforts. The definition of an architecting process based on the CMMI template for process definition will help organizations assess their architecture competency and set goals to improve their architecture practices. Finally, a successful demonstration of the effectiveness of SINERGY architecture will lead to SINERGY implementations that enable institutions to be prepared to respond to man-made tragedies on their campuses. Therefore, the impact can reach millions of students, faculty, and staff in thousands of colleges, universities, and other open campus environments.

1.4. DISSERTATION ORGANIZATION

Table 1-1 shows how this dissertation is organized.

Table 1-1. A summary of how the dissertation is organized

Chapter	Description
1	Problem Definition and Overview
2	Background
3	The Process of Architecting for Software/System Engineering
4	SINERGY Capabilities Specification
5	SINERGY Architecture Specification
6	SINERGY Architecture Evaluation
7	A Hierarchy of Indicators for SOA Security
8	Concluding Remarks and Future Work

CHAPTER 2: BACKGROUND

In this chapter, we present all related areas that influence our research. Since the problem we are addressing is multifaceted, our literature review spans different fields. We propose a network-centric architecture for SA, SC, and ERM in an open campus environment. Based on this, we can easily identify multiple areas that need to be studied in order to tackle this problem. These areas include network centricity, software and system architecture, architecting process, SA, SC, and ERM. Other fields also need to be investigated which relate to the ones stated above including SOA, Web Services, and security.

To properly summarize the related work in these fields, we divide this chapter into corresponding sub-sections for better readability and understandability. Figure 2-1 shows the components motivating this work, while the central contribution is the SOA-based network-centric architecture.

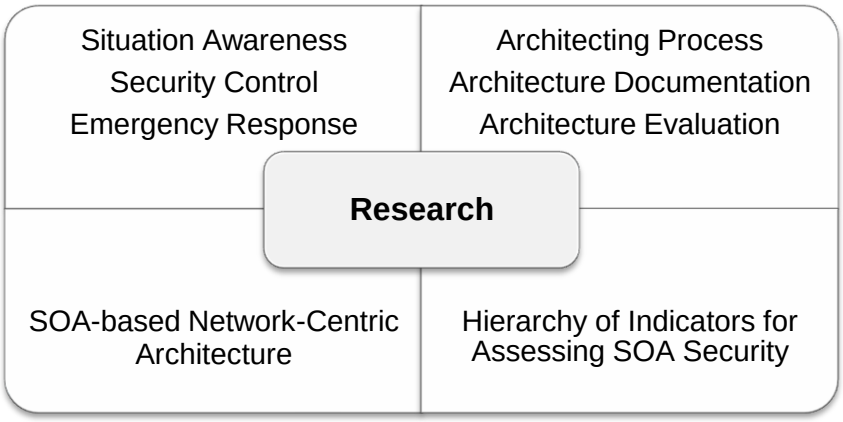


Figure 2-1. Areas of research

2.1. NETWORK CENTRICITY

Recent years have revealed a clear transformation from a platform-centric to a network-centric software development model [Waldo 1999; Xiaochun *et al.* 2002; Alberts 2002; Chigani 2007; Chigani and Arthur 2007; Chigani *et al.* 2007; Balci and Ormsby 2006, 2007, 2008]. Conventional software use focuses on applications installed and updated manually on local devices such as personal computers and mainframes. Increasingly, the need to use various smart devices to access remote data and computational resources has become inevitable. Furthermore, software acquisitions gave rise to challenges of integrating existing applications with the acquired ones. Additionally, technological breakthroughs in hardware and network communications have opened the door for the software engineering community to address larger and

more complex problems that were, until more recently, unsolvable. As a result, distributed, grid, cloud, utility, on-demand, service-oriented computing, and other similar terms have emerged as a product of the shift towards network-centric computing.

2.1.1. Origin

"Network centric" is a term used loosely in many sub-disciplines of software engineering including software architecture. Understanding the origin and background of this term enables us to use it more accurately and to describe what it means in the context of software/system development.

The term "network centric" has gained a widespread use after the introduction of the Department of Defense (DoD) network-centric warfare (NCW) and network-centric operations (NCO). The history of the term NCO can be traced as far back as 1996 when Admiral Williams Owens introduced the concept of "system of systems" in one of the reports published by the Institute for National Security Studies [Owens 1996]. However, Admiral Arthur K. Cebrowski and John Gastska formally introduced the term NCW in 1998 and later the term was given more depth in the book, *Network Centric Warfare Developing and Leveraging Information Superiority* [Alberts *et al.* 2000]. The book derives this new theory of warfare from a series of case studies on how businesses are using information and communication technologies to improve situation analysis, to accurately control inventory and production, as well as to monitor customer relations.

Much of the literary work related to network-centric computing was introduced within the context of military operations. However, several academic researchers have substantial contributions to evolving the field to where it is now. As early as 1998, Shaw [1998] recognized the emergence of a new approach to software development enabled by the prevalent use of the Internet. The premise of this approach is computing through dynamically-formed, task-specific coalitions of distributed, autonomous resources. These coalitions are characterized by a lack of direct control over the independently developed and managed components that constitute them. In addition, many other researchers took the concept of coalitions of resources a step further. In analyzing the challenges facing the emerging field of software architecture, Garlan [2000] identifies several architecting challenges related specifically to this new model of software development: Network-centric computing. Balci [2011] has been teaching the concept of network-centric computing since early 2000 at both the graduate and undergraduate level software engineering courses. He also evolved his evaluation methodology to be used for the assessment of network-centric system of systems [Balci and Ormsby 2008].

Even though the term "network centric" appeared formally in a pure military context, the shift towards network-centric computing was recognized in many non-military domains including industry and academic research. Since its public inception in 1998, many organizations have embarked on journeys to achieve the ideals of network centricity [Microsoft 2002; Boeing 2006; Oracle 2004a; NCOIC 2008; Raytheon 2008]. Many have different definitions of what network centricity is. However, these definitions revolve around a common understanding that network centricity:

- refers to a system the components of which work with each other over a network,
- is the extent to which a software-based system has the network as its main axis, and
- is the key to addressing today's complex computing problems where integration among geographically-dispersed systems is needed.

In the broader context, network-centric computing has the ideal of facilitating the sharing of information and computing resources between locations that have it and those that need it in soft and hard real-time environments. The value of this idea is that it can bring unprecedented success in achieving higher return on investments by orders of magnitude. The approach is to make the data, applications, and sometime human resources available on the network. This can all be summarized in the expression: "Connecting everything with everything else."

2.1.2. Reconciling the Terms: Network Centric and Software Systems

Network centricity is not just connectivity across systems or nodes, but between people, information, and cognitive domains. It stresses sharing information to achieve SA, which leads to increased speed of decision making within various operational environments. Network centricity is technical, operational, and behavioral. It has been the driving force behind the ongoing transformation of government and commercial business operations.

Network-centric systems focus substantially on their communication element. To accomplish the ideals of network centricity, these systems must achieve effective application and data integration. This integration is achieved by taking various systems on different platforms (i.e., operating systems), built with different implementation frameworks, expressed using different programming languages [Fay 2003], accessing different remote and local data repositories, and integrating them into robust systems for supporting critical business processes, scientific research programs, and military mission threads.

A key reason for becoming network centric is to be able to assemble software systems by integrating a mix of existing and new applications, and ensuring that the end product is capable of integrating with other net-ready applications. In general, this is the distinguishing characteristic between network-centric and distributed systems. The latter is engineered mostly for performance improvement. The former is engineered to reuse existing resources and integrate them to form larger and more complex systems over the network. For network-centric software systems, having their elements geographically distributed is a reality rather than a choice of the architect.

2.2. SOFTWARE, SYSTEM, AND ENTERPRISE ARCHITECTURES

When the term architecture comes to mind, three other qualifying terms come with it: Software, System, and Enterprise (Figure 2-2). It is crucial that we understand each type of architecture so that our discussion and proposed solution can be within the right scope. The concerns and the stakeholders in each type of architecture are different. There may be some overlapping but the scope remains different for each level.

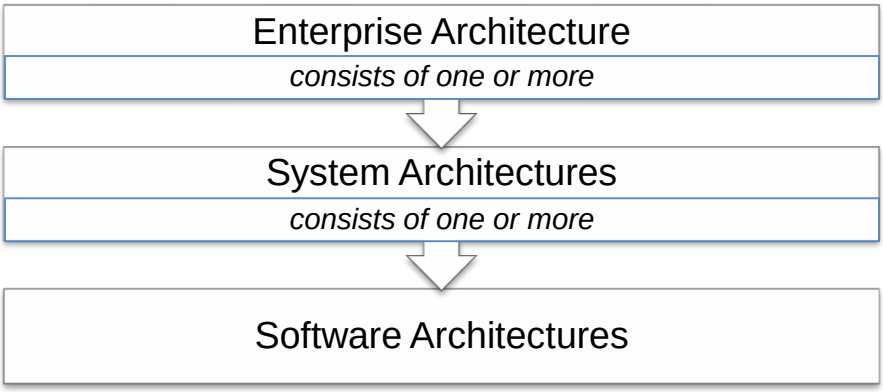


Figure 2-2. Architecture at all scales

2.2.1. Software Architecture

The software architecture of a system is "its fundamental organization […] embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution" [IEEE 2007]. The scope at this level of architecting focuses on the software components, the relationships that exist among these components, and the underlying operating environment, which include the hardware platform and communication network [Allen 2005; Allen and Garlan 1994]. Software architecture provides the blueprint for downstream development activities such as design, implementation, integration, and maintenance [Perry and Wolf 1992].

This research employs software architecture principles for developing, documenting, evaluating, and maintaining architectures of network-centric systems. The use of Web Services standards and technologies focuses the architecting process primarily on the software components that our proposed system has in order to provide SA, SC, and ERM in an open campus environment.

2.2.2. System Architecture

System architecture has a similar definition as that of software architecture. From a systems view, system architectures are also concerned with the system components, the relationships between these components, and the guidelines and constraints for their interactions. However, the types of components involved in system architecture include hardware and people as well. System architecture describes the solution from a system view while software architecture describes the solution from a software view. Generally, in complex systems or systems of systems, a system architecture may consist of one or more software architectures, representing the various software components that are involved in the system components.

However, in recent years the amount of software in almost all kinds of systems has grown resulting in an overlap in the activities performed by software and system architects. Today more than ever before, we see more communication and collaboration between these two types of architects to deliver quality software-intensive solutions. System architecture provides guidance, standards, constraints, and oversight for the software architecture. At the same time, there should be a feedback loop so that software architects can provide their expertise in developing the system architecture.

2.2.3. Enterprise Architecture

Enterprise Architecture (EA) is a new way of conceptualizing the functions of a large enterprise along with its products and services. EA relates organizational mission, goals, and objectives to work processes and to the technical or IT infrastructure required to execute them. Evidently, we can see how the scope of EA is larger than both system and software architectures. EA may consist of one or many system or software architectures. It serves as a medium for enterprise decision makers to reason about possibilities to align their goals with available assets and to respond effectively to changing requirements by reusing existing resources to form novel configurations of their assets to respond to change. At this level of architecture, the architect's focus is on utilizing architecture to introduce efficiency into the enterprise processes to shorten time to market and increase compliance with enterprise and cross-enterprise standards.

Several EA frameworks have emerged to capture the gist of enterprise architectures. These frameworks can be categorized into two. The first category of frameworks focuses mainly on the work products and services that an enterprise provides. The following is a list of prominent EA frameworks that are used in practice. More discussion about these frameworks can be found in the references for each.

- DoD Architectural Framework (DoDAF) [DoDAF 2009a, b, c]
- The Open Group Architectural Framework (TOGAF) [TOG 2009]
- Treasury Enterprise Architecture Framework (TEAF) [DoT 2000]
- Zachman Framework for Enterprise Architecture [Zachman 2009]
- Gartner Architecture Framework [James *et al.* 2005]
- Federal Enterprise Architecture (FEA) [FEA 1999]
- Ministry of Defense Architecture Framework (MoDAF) [MoDAF 2008]
- NATO Architecture Framework (NAF) [NAF 2007]

Enterprises typically have a set of business goals or missions that they are set out to accomplish. Therefore, another way practitioners categorize enterprise architectures is by looking at the business goal they are trying to achieve. This kind of categorization deals with all kinds of business-driven architectures. In practice, one might find categorizations such as data-centric architectures [Dineley 2005], process-driven architectures [Margulius 2005c], security architectures [Erlanger 2005], lightweight and open architectures [McAllister 2005], pervasive architectures [Margulius 2005b], service-oriented architectures [Udell 2005], and needs-based architectures [Margulius 2005a]. This labeling scheme refers to enterprise architectures that are focused on a specific goal or mission.

In our research, we propose an architecture for a network-centric system. We carry out this development from software and system level perspectives considering the software, hardware, and people components. The focus is on the software components but some system engineering issues are considered as well. We only get into the enterprise level when we look at the stakeholders of the proposed systems.

2.3. THE ARCHITECTING PROCESS

At odds with the state of the practice, software engineering textbooks (e.g. [Pressman 2010; Sommerville 2007]) do not necessarily place proper emphasis on architecture and the architecting process. In their most recent versions, these textbooks include chapters about architectural design – a term that, even though it is widely used in the literature, is confusing to say the least. In software engineering, design and architecture mean very specific things. In many other engineering disciplines, the distinction does not need to be

made. However, design in software engineering describes a life cycle phase that is concerned with how the system components will be built. On the other hand, architecture or architecting is a separate phase that is concerned mainly with the structure of the system in terms of its components, the relationships among these components, and the constraints that are placed on these relationships. The goal of the two phases is different and so is the work products produced by each. Even the skills required in each phase are different. Therefore, when the term "architectural design" is used in textbooks and other literarily work even when it denotes architecture or architecting, it can be misleading.

In the network-centric era, there is no such a thing as architectural design. There is architecture or architecting and there is design. The best analogy that can be made is as follows: Design is to architecture is what an object is to a class. For instance, when we create an architecture for a system based on SOA, we can use either the Java Enterprise Edition framework or the .NET framework to develop a design for the system. Therefore, an architecture is an abstraction of the components of the system and the way these components relate to one another. Designing those components based on either framework is a concern of the design phase not the architecting phase.

A simple Google search of the term architecting will yield almost 1.5 million hits. Thus, the term itself is not specific to software engineering. Many practitioners in small and big organizations have already adopted the term architecting. In addition, pioneers in architecture research in University of Southern California, Carnegie Mellon University, DoD contractors, Microsoft, IBM, and Siemens among others also have used both architecting and architectural design in their literature. There is a certain amount of maturity in the use of the term. There is also a wide adoption of architecture practices in industry. However, this has not caught on in the way we train next generations of architects and software engineers. One reason is the lack of a systemic emphasis on architecture and the architecting process in textbooks and software engineering programs. My experience in attending the Software Architecture Educator's Annual Workshops since 2006, one of the few workshops that deals with how to teach architecture in software engineering programs, shows that many educators and researchers complain about the lack of a systemic inclusion of architecture principles in software engineering curricula.

There is a great need to leverage the literature available for practitioners and in some other research publications to define an architecting process as a CMMI process area of its own right. In this research, the architecting process is defined similar to how CMMI defines its process areas such as Requirements Development, Verification, and Validation. More importantly, we capture the activities and the skills

needed to execute the architecting process in a structured manner to enable teaching these concepts both in software engineering programs and in training seminars within government and industry organizations.

2.4. SERVICE-ORIENTED ARCHITECTURE

When one navigates the literature related to SOA, it is easy to see how the principles of service orientation existed as early as the introduction of component object architecture, especially in Microsoft's Distributed Component Object Model (DCOM) [Microsoft 1996] and the Object Management Group (OMG)'s Common Object Request Broker Architecture (CORBA) [OMG 2010]. However, it is undeniably clear that SOA and Web Services are the reasons why service orientation has become the *de facto* architectural paradigm used to achieve network centricity. Erl [2005] describes SOA as an architectural model that improves the efficiency, agility, and productivity of an enterprise. The basic elements of any SOA solution are the services it provides. It positions these services as reusable building blocks for other solutions. Therefore, the challenge with SOA is not with building services. The challenge lies in creating services that can be reusable—services that are agnostic to the business processes in which they participate.

The basic representational model for SOA is shown in Figure 2-3. The premise of SOA is that a service provider publishes its service description on a registry for others to find. Service consumers can then look up registries for offered services and request those services on a need basis. Many approaches exist today to implement SOA including the Representation State Transfer (REST) [Fielding 2000]. The people who developed SOA-like solutions as early as 1996 [Rosen *et al.* 2008] used CORBA objects as reusable entities. However, the full impact of SOA was achieved through the development of Web Services standards and technologies namely eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI). In this research, when we talk about SOA, we mean SOA-based on Web Services

The reason why SOA is the de facto approach for achieving network centricity is that SOA is focused on the development of systems with an enterprise or cross-enterprise scope [Rosen *et al.* 2008]. Often times, enterprise resources are heterogeneous and geographically dispersed requiring a network-centric solution to connect these resources with one another. Furthermore, inter-organization interactions are increasingly needed to connect organizations with needs (expressed in the form of business processes) and others with corresponding capabilities (expressed in the form of services).

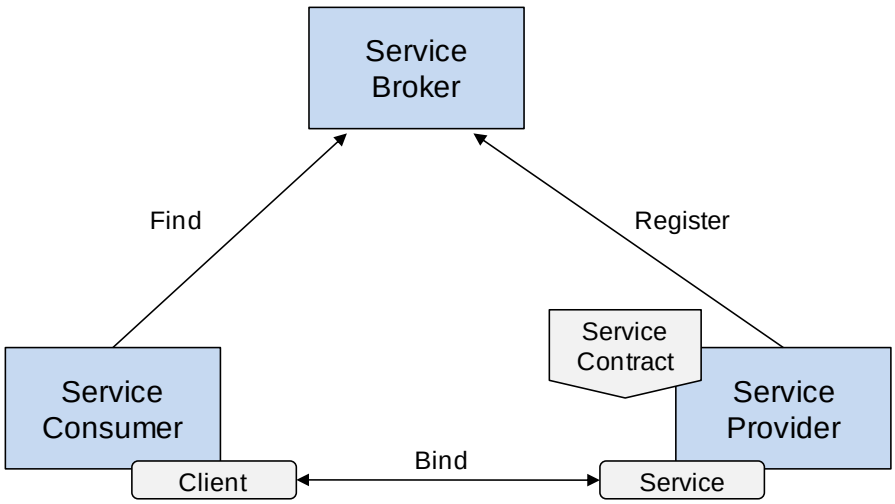


Figure 2-3. SOA basic architectural paradigm

Figure 2-3 only describes SOA from a high-level perspective. However, when we consider SOA based on Web Services, creating a full solution based on SOA requires more understanding than that provided by the basic architectural paradigm. In order to understand SOA fully, we consider the layered description provided in Figure 2-4 [Glass 2008; Balci and Ormsby 2008].

Figure 2-4 corresponds with our understating of the basic elements of the SOA. Four basic elements are needed in every SOA solution: Web Services Basic Profile, Enterprise Service Bus (ESB), Orchestration, and Services Management as depicted in Figure 2-4. These elements are described in the following sections.

2.4.1. Web Services

Every SOA solution should have services that provide useful functionality and that can be accessed by service consumers. The World Wide Web Consortium (W3C) [W3C 2004] defines a web service as "… a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HyperText Transfer Protocol (HTTP)." The Web Services layer represents services that can be called in a standard way so anyone can use them without knowing their internals. The basic profile (or best practices) for Web Services includes XML, HTTP, WSDL, and SOAP. The basic profile describes the standards, and their version, that can be used to develop and deploy Web Services. The following are descriptions of some of the basic profile standards mentioned above.

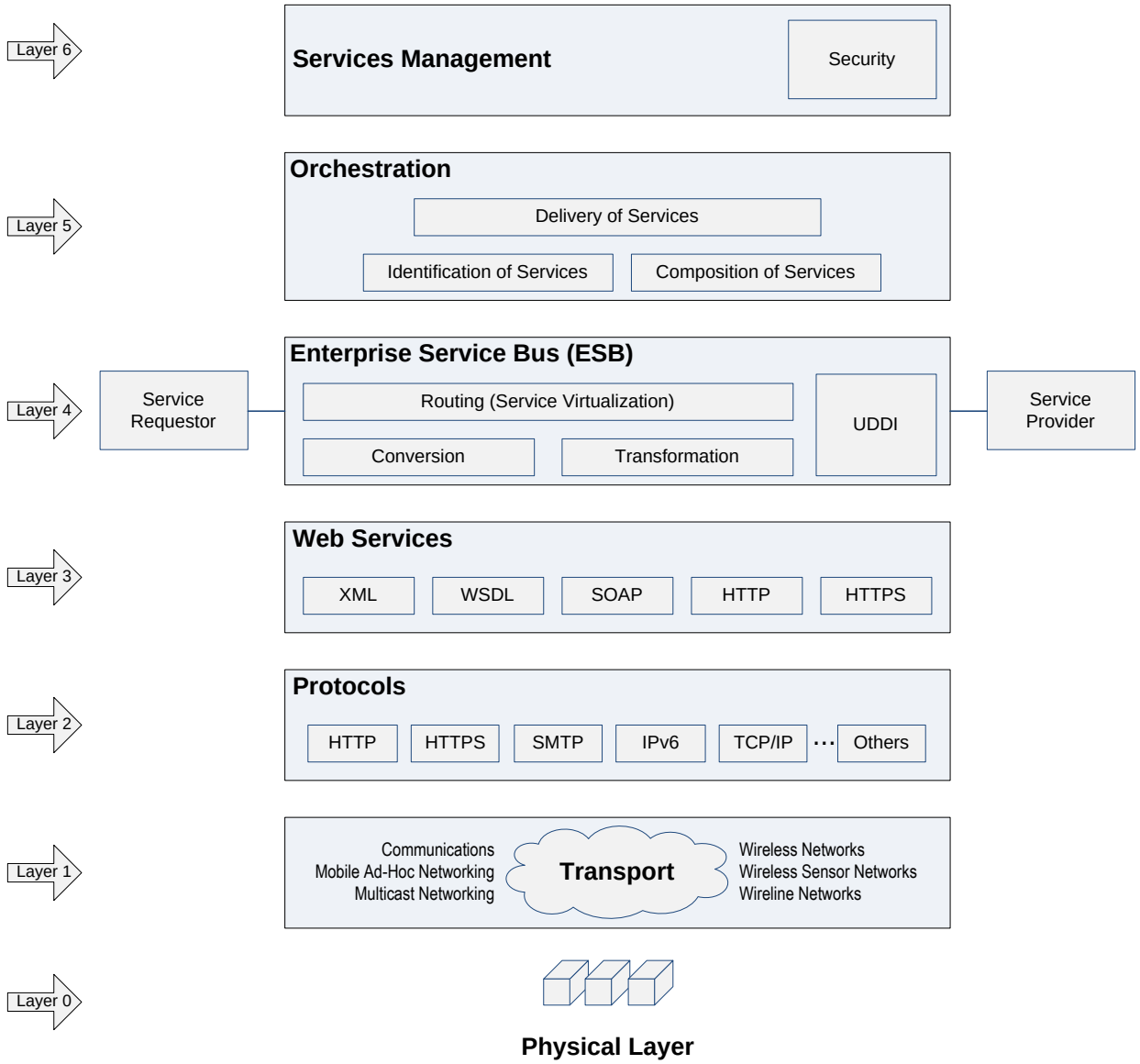


Figure 2-4. Conceptual layers of Web Services-based SOA (Used under the fair use guidelines, 2011)

2.4.1.1. XML

XML stands for the eXtensible Markup Language. It is a markup language designed to describe structured content in documents. Unlike HTML tags that are used to format data, XML tags are not predefined [W3 Schools 2008]. Therefore, the key benefit of XML is that it is extensible; meaning that new tags can be defined allowing applications to describe in a detailed and meaningful manner their data. XML is the language used in Web Services messages, and is the basis for Web Services standards. An example of an XML document describing a note is shown in Table 2-1.

Table 2-1. An XML document describing a note

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
        <to>
                CS6704 Class
        </to>
        <from>
                Amine Chigani
        </from>
        <heading>
                Reminder
        </heading>
        <body>
                See you in class today!
        </body>
</note>
```

2.4.1.2. WSDL

WSDL stands for Web Service Description Language. It is an XML-based language for describing Web Services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol (e.g., HTTP) and message format to define an endpoint [W3C 2001b]. WSDL structure consists of six elements described in Table 2-2.

Table 2-2. Structure of a WSDL document

<definitions>	Root WSDL element
<types>	Data types used by web service
<messages>	Messages used by web service
<portType>	Operations supported by web service
<binding>	How will message be transmitted?
<service>	Location of the web service

WSDL is a machine-understandable language used by service providers to describe the capabilities and the requirements of their services. WSDL documents are documents that contain these descriptions and that are published in the UDDI registry. The following is excerpt from a WSDL document (Table 2-3).

Table 2-3. An excerpt from a WSDL document gas price service

```
<definitions name="GasPrice" >
    …
    <message name="GetGasPriceRequest">
        <part name="ZipCode" type="xs:string"/>
    </message>
    <message name="GetGasPriceResponse">
        <part name="Price" type="xs:integer"/>
    </message>
    <portType name="InGas">
        <operation name="GetGasPrice">
            <input message="GetGasPriceRequest"/>
            <output message="GetGasPriceResponse"/>
        </operation>
    </portType>
    …
    <service name = "GasPriceService">
        …
    </service>
</definitions>
```

2.4.1.3. SOAP

SOAP originally stood for Simple Object Access Protocol. Later, the full name is dropped and only the acronym is used now. It is a protocol specification used for the exchange of XML-based messages among Web Services over a network. SOAP codifies the existing practice of using XML and HTTP as a method invocation mechanism [W3C 2003]. Table 2-4 shows an example of a SOAP request/response message exchange between a service consumer and a service that provides gas prices based on zip codes.

Table 2-4. Example SOAP request/response messages

SOAP Request

```
POST /Gas   HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/gas">
    <m:GetGasPrice>
        <m:ZipCode>
            24060
        </m:ZipCode>
    </m:GetGasPrice>
</soap:Body>
</soap:Envelope>
```

SOAP Response

```
HTTP/1.1   200   OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap=http://www.w3.org/2001/12/soap-envelope
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/gas">
    <m:GetGasPriceResponse>
        <m:Price>
            3.48
        </m:Price>
    </m:GetGasPriceResponse>

</soap:Body>
</soap:Envelope>
```

2.4.2. Enterprise Service Bus

The second layer of any SOA needs is to provide an intermediary that facilitates the interactions between service consumers and service providers. The ESB is a combination of software and hardware middleware that offers a standard way for services to interact and communicate with one another. The ESB must be able to, at least, provide the following capabilities: Connect, Transform, and Route. These capabilities are described below.

- **Transformation:** An ESB should provide the capability to connect to Web Services through either WSDL or adapters in case of legacy systems.
- **Conversion:** An ESB should perform data transformation to and from data types that are required by the Web Services to which it connects.
- **Routing:** An ESB should route messages from one service to the others using appropriate communication protocols for each service.
- **UDDI** is a platform-independent, XML-based registry for service providers to list their services on the Internet. UDDI is an open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), enabling businesses to publish service listings, discover each other, and define how the services or software applications interact over the Internet [OASIS 2004a].

These basic capabilities ensure the loose coupling between service consumers and service providers. Implementation details, data types, and communication mechanisms are made transparent through the

ESB enabling an agile environment that can adapt to change, which has been a key driver in the wide adoption of SOA.

2.4.3. Orchestration

One of the most important principles of service orientation is orchestration. Bernstein [2005], a music composer, said that "the right music played by the right instruments at the right time in the right combination: That's good orchestration." In SOA, the orchestration layer is concerned with the issue of assembling services together dynamically to provide some desired capabilities. The challenge is that it has to be done at runtime, posing the question of how we can know all potential service configurations at the architecting phase. We achieve orchestration of services by creating workflows that map to business processes, which may extend across one or more organizations. Service orchestration follows these workflows at runtime.

Among other options, the standard language today for implementing orchestration in SOA solutions is the Business Process Execution Language (BPEL). BPEL is based on XML Schema, SOAP, and WSDL. It offers a standard language for defining how to [OASIS 2007c]:

- Send XML messages to remote services.
- Manipulate XML data structures.
- Receive XML messages asynchronously from remote services.
- Manage events and exceptions.
- Define parallel sequences of execution.
- Undo parts of processes when exceptions occur.

Fundamentally, BPEL is an XML-based programming language similar to Java or C#. It enables a developer to create logic that defines what happens when requests require the collaboration of more than one service to provide the desirable capability. For instance, a full vacation package request will require the collaboration of many services such as hotel reservation service, flight reservation service, limousine reservation service, ticket booking service, credit card processing service, and many more. For this kind of requests, a workflow is developed using BPEL and is executed at runtime when such a request is received. Another example is in exception handling where a secondary service provider is used if a primary service is not available.

2.4.4. Services Management

Services Management is the highest level in the layered SOA representation described by Glass [Glass 2008]. Service orientation principles all revolve around the concept of a service as a reusable building block for SOA environments. Due the complexity of the network-centric systems we build today, any non-trivial SOA-based network-centric system will be composed of hundreds, if not thousands, of services. At this scale, management of these services as enterprise assets becomes critical. Proper management of these services will yield higher return on investment as more and more services are reused in different configurations to satisfy more and more business needs. Bad management will result in redundancy, configuration problems, versioning issues, and security vulnerabilities.

Several Web Services standards and specification have been created to address services management at all levels of complexity. This component of SOA is concerned with ways to manage and secure services using standards such as WS-Security [OASIS 2004b], WS-Federation [OASIS 2007a], WS-Policy [W3C 2007a,b] and Identity Management. At this level, a major research question deals with addressing security at this layer and trickling its effect to implementation. This way, the developer is relieved from the responsibility to address the security requirements at the code level and the security expert and the architect are enabled to specify security policy suitable for the whole system at a high level.

Security is an important aspect of services management as it can either make or break many interoperability mechanisms between services. For instance, when two collaborating organizations use different security standards to ensure access control to their services, interoperability is negatively affected. When service A requires Security Assertion Markup Language (SAML) assertions to be specified in the request SOAP message about what a service consumer is authorized to do, the service consumer must be implemented in such a way that it can provide such an assertion, otherwise it cannot interact with the service provider even if everything else is interoperable. Addressing security at the management layer becomes important. More research is needed to define how security can be specified at this level.

2.5. SOA SECURITY

In the platform-centric computing era, security was a big yet a manageable challenge. In many standalone applications and software products, it was enough to protect them inside the enterprise firewalls. In the new network-centric era, security challenges rose to a level that traditional security measures are not enough. In this era, organizations want to dynamically do business and share information across organizational boundaries. This led to blurring the line between what is internal and what is external

[Rosen *et al.* 2008]. Everything is a service and is available to be accessed by anyone, anywhere, anytime. Pervasiveness is a common term used to describe the way business and other activities are conducted in the new era of computing.

SOA provides an architectural paradigm that enables network-centric computing. In particular, implementations of SOA environments based on Web Services benefit a great deal from standards and specifications that make the daunting task of securing one's SOA solutions achievable, if done properly. To deal with security within an SOA-based network-centric system, a plethora of standards has been created to address various aspects of SOA security. In this section, we list the most prominent and widely adopted standards and specifications related to SOA security. A brief description of the standard and a reference to its authoring organization or group is also provided.

2.5.1. WS-Security

WS-Security is a widely accepted OASIS Standard developed for the purpose of providing end-to-end secure SOAP messaging. It was first released in 2002, and WS-Security version 1.1 was released in 2006. WS-Security specification expands the SOAP messaging protocol to provide integrity, non-repudiation, confidentiality, and token passing. One of the main strengths of WS-Security is that it leverages other standards to provide secure messaging including XML Signature and XML Encryption. In addition, WS-Security supports the use of multiple token formats for authorization of subjects, and supports multiple trust models used for sharing security contexts [Rosen *et al.* 2008]. Examples of token formats supports are SAML Token Profile, X.509 Certificate Token Profile, Kerberos Token Profile, and Username Token Profile.

WS-Security standard is supported by most SOA product vendors and is independent from any particular infrastructure. For these reasons, it is the wisest choice to provide secure messaging among service providers and service consumers. More details about the standard can be found in the OASIS website at: http://www.oasis-open.org/committees/wss.

2.5.2. XML Signature

XML Signature is a W3C recommendation with a latest edition released in June 2008 [W3C 2006b]. It is also referred to as XML-SIG or XML-DSIG. XML Signature is a type of digital signature used for XML-based transactions. This type of signature can be used to digitally sign an XML document or an element of an XML document. The purpose of XML Signature is to provide for message integrity (it has not been tampered with in transit) and non-repudiation (sender cannot deny sending it). XML Signature defines an

XML schema (or syntax) for capturing the results of digital signature operations applied to XML data [Naedele 2003].

One of the useful features of this security recommendation is that it allows different elements of a message to be signed by different entities. This feature proves to be extremely useful in network-centric environment where components are geographically dispersed and are possible owned or managed by different entities. In addition, due to its flexibility, XML Signature is used in WS-Security SOAP messaging where information in the SOAP header, such as subject's identity and message timestamp, is cryptographically tied together with the SOAP body to prevent attackers who may get access to the security information in the SOAP header to reuse it with other SOAP messages [Rosen *et al.* 2008]. More details about this recommendation can be found in the W3C website at: http://www.w3.org/TR/xmldsig-core.

In addition, W3C released a related draft document called XML Signature Best Practices available at: http://www.w3.org/TR/2009/WD-xmldsig-bestpractices-20090226/.

2.5.3. XML Encryption

XML Encryption is also a W3C recommendation that was released in 2002. This recommendation is formally called XML Encryption Syntax and Processing, and it details a process by which data can be encrypted and represented in XML format [W3C 2002]. There are no constraints on the data that can be encrypted. Any type of data including XML documents, XML elements, or XML element content can be encrypted and the product of that encryption is represented as an XML Encryption element. This element encloses or points to the cipher data.

Just like XML Signatures, XML Encryption is also used in WS-Security SOAP messaging to provide confidentiality of the data transmitted at the XML document level or at the XML element level. This level of granularity affords the security needed in multi-hop message transmission where sensitive data is encryption and can only be decrypted and seen by the intended parties and not by any intermediaries [Vega and Epstein 2006]. More details about this recommendation can be found in the W3C website at: http://www.w3.org/TR/xmlenc-core/.

2.5.4. WS-Policy

WS-Policy is a W3C recommendation that was released in 2007. It is an assertion framework and general-purpose model with its corresponding syntax for expressing the requirements and capabilities of

Web Services. WSDL lacks the expressiveness needed to describe requirements for performance, quality of service, and security among others [W3C 2007a]. For this reason, WS-Policy provides an important extension to other Web Services standards to describe how service consumers and providers can specify their requirements and capabilities in the form of policies. These policies are specified in XML, which make them machine readable. This fits the dynamic environments of SOA deployments. Currently, service providers and service consumers share information about such policies in the form of human readable documentation or through personal correspondence [Microsoft 2002].

More details about this recommendation can be found in the W3C website at:
http://www.w3.org/Submission/WS-Policy/.

2.5.5. WS-SecurityPolicy

WS-SecurityPolicy is a subset of the WS-Policy. IBM led the development of this standard along with other industry participants, and it then became an OASIS standard in 2007. WS-SecurityPolicy provides the syntax by which security assertions can be defined and used by other specifications such as WS-Security SOAP Messaging, WS-Trust, and WS-SecureConversation (which are discussed later in this section). This standard enables service developers to describe their security requirements explicitly for service consumers that want to communicate their services. This capability is not afforded by using WSDL because it is not expressive enough.

The latest version of this standard can be found at the following OASIS link:
http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html.

2.5.6. WS-SecureConversation

WS-SecureConversation is an OASIS standard released in 2007 [OASIS 2007b]. It builds on the capabilities of WS-Security to provide long-lived, multi-message secure communications between services [Erl 2008]. Typical SOA implementations rely on the classical one message request/reply model. However, with the increase of the dynamic natures of SOA environments, services may require multiple messages to be exchanged in order to complete a transaction. From a security standpoint, each message sent must have appropriate security measures such as the ones we discussed so far: Encryption, confidentiality, and integrity. Performance wise, this approach is bad for the efficiency by which business processes are expected to be carried out. As a solution, WS-SecureConversation standard defines messaging semantics that can be used for multiple message exchanges.

The way it works is as follows. The WS-Security standard is focused on security at the message level. It defines mechanisms by which a message can be securely exchanged between services. On the other hand, the primary goal of WS-SecureConversation standard is to define security contexts between services. A security context is a new WS-Security token type that can be created by a Security Token Service (STS) in WS-Trust model. More details about this standard can be found in the OASIS website at: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html.

2.5.7. Security Assertion Markup Language

SAML is an OASIS standard that was first released in 2002. Prior to this standard, there was no approach for service providers to exchange security information about their services with partners or service consumers, especially using XML-based approach. SAML was developed for this purpose: Enable the exchange of security information such authorization and authentication information based on an XML-based framework.

SAML is an XML-based framework for communicating user identity and attribute information using assertions [OASIS 2002]. An assertion is simply a declaration about a subject. For instance, a SAML assertion can be a declaration about a user's authentication, a list of user's attributes (permissions), or a statement granting or denying a user access to a requested resource [Rosen *et al.* 2008; Epstein *et al.* 2006]. This language is extensible and can provide the flexibility needed in the dynamic SOA environment. For the latest information about SAML 2.0, visit the following link: http://saml.xml.org/.

All the abovementioned security standards and specifications are available for the SOA architect to use. They provide solutions to specific security needs just as design patterns provide solutions to specific design problems. As explained earlier, many of these standards are complementary to each other. The architect has the responsibility to make a decision on which mix of standards to use, and how they should all fit together. This decision in not easy and guidance as to how to make such a decision is needed. At the architecture level where security must be addressed, there needs to be a structured approach to adopting WS-* standards to provide a comprehensive level of security.

2.6. KEY DRIVERS FOR THE PROPOSED ARCHITECTURE

The main goal of this research is to address the issue of balancing safety and openness on campuses through a network-centric solution. Three main drivers guide the development of the proposed architecture: SA, SC, and ERM. These key drivers are kept in mind at all times during the architecting and evaluation processes. The following sections discuss these drivers in detail and explain our

understanding of them. Only pertinent information to our problem domain is discussed in this section. We understand each driver is an area of research in and of itself.

2.6.1. Situational Awareness

SA is "all the knowledge that is accessible and can be integrated into a coherent picture, when required, to assess and cope with a situation" [Sarter and Woods 1991]. The definition emphasizes a couple of very important points. First, access to all knowledge (in the form of data) is paramount to having SA. Second, the definition stresses on the ability to integrate this knowledge in useful ways to make appropriate decisions during a situation. In the context of software-based solutions, SA is afforded when the system enables its users to have to access to important information to make appropriate decisions in various situations. This definition is in concert with the goal of network-centric computing: Connecting everything with everything else to provide access to all available information (and functionality) in order for that information to be used in various kinds of situations.

In particular, providing SA is a critical objective of any network-centric system developed for the purposes of SC and ERM, particularly in open environments. SA, in fact, is supposed to be an essential prerequisite to the safe operation and management of any complex environment [Endsley 2000]. This simply means that the system must enable the creation of a COP at any point in time so that it allows its users to effectively deal with situations as they occur. To do so, SC personnel and administrators need to be able access information that enables them to make the right decisions about an incident based on a common knowledge of how the campus looks like during that incident.

In our problem domain, SA can be achieved through monitoring devices that can be deployed all over campus. Authorized personnel can connect to these devices through a central command system. When we surveyed the available technologies that can facilitate this monitoring, few technologies came to bear including wireless sensor networks, IP-based cameras, alert systems, and various sensing technologies [Jason 2003; Lorinez *et al.* 2004; Zang and Wang 2006]. These technologies can be deployed all around a campus environment to automatically monitor, intelligently assess, and report situations and incidents to a central location where decisions can be made.

Wireless Sensor Networks

A wireless sensor network consists of a collection of small and simple sensor nodes distributed over a geographic area. They are used in applications such as surveillance, security, fire detection, industrial process control, environmental, and health monitoring [VaCAS 2008]. Each node has one or more

sensing devices (e.g., microphone, thermometer, or camera), a wireless communication device, simple processing capability, and a limited energy supply. Though each node is an independent hardware device, they need to coordinate their sensing, communication, and computation in order to acquire relevant information about their environment to accomplish some high-level task [Lewis 2004].

In our context, these networks can be used to sense and report information to a central campus location to be analyzed. A constant flow of such real-time information will enhance SA among security personnel and administrators and reflect positively on the decision-making process during an incident.

To show the usefulness and the applicability of wireless sensor networks, we list few types of sensors that exist today.

- Chemical sensors
- Magnetic sensors
- Position sensors
- Pressure sensors
- Temperature sensors
- Motion sensors

The application of these types of sensors can span all aspects of our lives from finding our keys at home to monitoring the movement of an enemy in a war zone. For instance, enhancing regular monitoring cameras with sensing capabilities can enhance the role of these cameras to only monitor and record pertinent information. A sensor-based camera trained in object recognition can differentiate between a person and an animal, and between a car and a track. Such capability will reduce the amount of information that needs to be transmitted over the network to the central node thereby enhancing performance and relevance of the information.

This area is an emerging area of research that has borne a lot fruit to date. Many institutions and research labs including Virginia Tech have active research labs and groups in this area. Our work led to collaboration with researchers in this field to identify best technologies that can be used for SA and security on a campus environment.

2.6.2. Security Control

The second driver of our architecture is SC. The need to be able to have a system that enables security providers (i.e. police and others) on campuses to coordinate their efforts of deterring crime and responding to incidents in the most efficient manner is of a paramount importance [George *et al.* 2001]. In providing a network-centric environment, these security providers are able to connect all their resources including their systems in a fashion that makes control and management of security on campus more effective.

SC requires the deployment of monitoring and surveillance equipment across campus in order to facilitate SA and provide a COP for all campus constituents [Professional Underwriters 2006]. However, serious considerations need to be given to the fact that students, faculty, and staff have privacy and personal space rights that need to be preserved. For instance, locker rooms and lounge areas remain places where placing cameras may raise several issues. Table 2-5 contains a list of privacy and security considerations derived from the literature and offered by a subject matter expert (SME) in the area of security.

The use of specific types of cameras can be helpful in addressing some of the concerns over SC. For instance, there are certain types of cameras that are trained to capture certain objects such as license plates. These cameras should not be of a concern and can be placed anywhere needed. Another type of cameras is equipped with motion or sound sensors. These can be placed in placed where perpetrators may hide such as closets and equipment rooms. Other types of controlled cameras can be used when monitoring is needed but privacy is of a concern. They can be used only on a need basis.

In our architecture specification, we address recommendations of how to deal with these issues by providing an architecture that is tailorable to accommodate various types of campuses. For instance, private schools may be able to enforce the monitoring of areas such as lounges, offices, cafeterias, and others. In many open campuses, however, these areas may not be monitored. Our solution addresses various types of campus environments, especially those with various levels of openness.

Table 2-5. Privacy and security considerations for an open campus environment

Area Type	Monitoring	Rationale
Locker Rooms	No	Inherently private areas and shouldn't be monitored
Restrooms	No	Inherently private areas and shouldn't be monitored
Lounge Areas	No	Monitor areas that provide access to and from lounges but not the lounges themselves. The challenge is that if advertised they will be targets to planned attacks.
Hallway Areas	Yes	Safety: Track fleeing perpetrator; traffic studies for stats and modeling students to be applied to class scheduling
Classrooms	Yes	Safety: This is for safety and real-time SA of events going on in classroom when incidents occur. Academics: Data can be used for increased levels of raw recorded content that can support other university initiatives
Walkways	Yes	Provide SA and SC of common walkways between campus buildings
Parking Lots	Yes	Provide SA and SC of campus parking lot to manage proper emergency evacuation.

2.6.3. Emergency Response Management

The third driver for our architecture is ERM that is effective and appropriate. There is an increasing concern among campus administrators regarding ERM and preparedness. In the wake of the April 16 tragedy, Virginia's Governor sent an e-mail to all state employees about a new initiative called: The Adjunct Emergency Workforce [DHRM 2007]. This initiative aims at engaging state employees in efforts to stand at the forefront of a response to any disaster or emergency. Within this effort, employees will be assigned to emergency response teams during periods of declared emergencies. These teams will have to coordinate among one another to provide best possible care for people affected by these emergencies.

Here at Virginia Tech, the administration also made a similar move by updating its guidelines about how to respond to emergencies. In an e-mail on March 8, 2008, the Associate Vice President for University Relations laid out the primary and secondary channels of communications that would be used in the case of an emergency.

From these examples and many others, it is evident that officials also see the importance of connecting all available resources (people, entities, departments, systems, information, etc.) to come to bear in the face of a tragedy. ERM has become a full-fledged responsibility of administrators of all educational institutions. Therefore, our research lies at the heart of software engineering research—providing innovative solutions to real problems. Accordingly, we see a need for a comprehensive software-based solution to facilitate efforts and initiatives such as the one proposed by Virginia's Governor. Our architecture serves as a foundation for the development for such solutions.

CHAPTER 3: THE PROCESS OF ARCHITECTING FOR SOFTWARE/SYSTEM ENGINEERING

With the advent of potent network technology, software/system engineering has evolved from a traditional platform-centric focus into a network-centric paradigm where the "system of systems" perspective has been the norm. Under this paradigm, architecting has become a critical process in the life cycle of software/system engineering. The need for a structured description of the architecting process is undeniable. This chapter fulfills that need and provides a structured description of the process of architecting a software-based network-centric system of systems. The architecting process is described using a set of goals that are specific to architecting, and the associated specific practices that enable the realization of these goals. The architecting process description presented herein is intended to guide the software/system architects.

3.1. INTRODUCTION

The ubiquity of the network and the ability to deploy software over a network has changed the underpinnings of software-based solutions. The "system of systems" perspective [Maier 1998] dominates much of the engineering now as new systems are composed of multiple interconnected systems to support emerging missions. One reason behind this shift is the need to reach beyond tightly-coupled environments to access data and functionality that reside on remote systems deployed on different platforms, and which are possibly owned and managed by different entities. Another reason is the dynamic and complex structures of today's organizations, where the organization's computing resources can span multiple national and international locations.

These changes gave rise to the Software-as-a-Service (SaaS) paradigm, where software-based systems no longer reside on users' own computing devices. Instead, software-based solutions are provided as services over a network, and users access these services through a plethora of network-capable devices that range from mainframe computers to smart phones.

Under the SaaS paradigm, architecting has become a distinct and essential life cycle process for software engineering as well as system engineering. Today, architecting is well recognized in practice as evidenced by job titles such as Software Architect, Solution Architect, Enterprise Architect, Application Architect, Integration Architect, and Information Architect. In addition, several programs exist to confer professional certification and certificates to architecture practitioners. Most of such programs are

organizationally based promoting individuals along the organization ladder, and are typically focused on in-house technologies and products. Examples of such programs include:

- IBM Professional Certification Program [IBM 2010].
- Microsoft Certified Architect Program [Microsoft 2010a].
- Raytheon Certified Architect Program [Raytheon 2010].

Other programs exist to grant professional architecture certification to practitioners who are interested in going through the certification process. Examples of such initiatives include:

- The International Association of Software Architects (IASA): Certified IT Architect Program [IASA 2010].
- The Software Engineering Institute (SEI): Software Architecture Professional Certificate Program [SEI 2010b].
- The Open Group: IT Architect Certification Program (ITAC) [TOG 2010].

On the one hand, the architecting discipline has moved from mere qualitative observations of the structure of working systems into a rich repertoire of concepts, methods, standards, frameworks, and tools to create, document, analyze, and assess architecture specifications. Architecting has become an "indispensible" process in the software/system engineering life cycle [Shaw and Clements 2006]. This indispensability makes it vital for software-based solution providers to execute best architecting practices in order to deliver quality solutions.

On the other hand, the adoption of the current architecting knowledge is dependent on the existence of a defined architecting process that integrates with system/software life cycle processes. Processes are important to streamlining the work required to develop a software-based solution from inception through operation and to retirement. Solution providers typically focus on three dimensions of their development approaches: People, methods, and tools [SEI 2010a]. To link these dimensions together, organizations follow processes of the software/system engineering life cycle to ensure the production of systems with the desired functionality and quality, on time, and within budget. However, published life cycle models leave out architecting from their processes.

The CMMI for Development is a widely-used process improvement and appraisal approach, which provides a reference point to organizations to assess their competencies in process areas related to software/system development [SEI 2010a]. A process area is a set of specific practices that achieve a set

of specific goals for making improvement in that area. A specific goal of a process area is a distinctive objective that must be met in order to satisfy that area. A specific practice is an activity that must be carried out to achieve part or all of a corresponding specific goal.

CMMI's basic philosophy is that the quality of a product or service is positively correlated with the processes used to develop such product or service. However, CMMI also leaves out architecting as one of its 22 defined process areas. Although CMMI helps organizations identify process areas crucial to developing quality products and provides a framework to improve such processes, architecture is briefly discussed as part of a Specific Practice (*SP 2.1 Design the Product or Product Component*) of a Specific Goal (*Develop the Design*) of the Technical Solution Process Area.

Evidently, architecting requires more prominence in the life cycle than what it currently has in the CMMI models [Valerdi *et al.* 2008]. We advocate that architecting should have its own process area. The purpose of this paper is to provide a structured description of the process of architecting software-based network-centric system of systems. We propose that such a description of the architecting process be added as a CMMI process area at Level 3. We describe the architecting process area following the CMMI for Development structure. However, for the purposes of this paper we only emphasize the specific goals and specific practices of the architecting process area.

The remainder of this chapter is organized as follows. Section 2 presents an overview of the architecting process area. The specific goals and practices of the architecting process are described in Section 3. Finally, concluding remarks are provided in Section 4.

3.2. THE ARCHITECTING PROCESS AREA

We define the architecting process area and propose it as an Engineering Process Area of the CMMI for Development at Maturity Level 3. Engineering process areas are processes that span the entire development life cycle of a system from its inception to retirement. Maturity Level 3 process areas are established processes within the organization and are executed in a proactive manner.

The purpose, activities, and performers of the architecting process, as well as the uses of architecture specification are described below.

3.2.1. The Purpose of the Architecting Process

The purpose of architecting is to develop an architecture specification for a software-based system (or a system of systems). The process of architecting takes the *Problem Specification* and *Requirements Specification* as input and produces an *Architecture Specification* as an output work product.

The development of an architecture specification focuses on several objectives. The primary objective is to ensure that system quality attributes (i.e., non-functional requirements) such as interoperability and security are architected for early on in the development life cycle. Another objective is to ensure that the architecture specification is a usable asset for decision makers to evaluate investment alternatives of resources such as systems, system components, and middleware technologies that will be used in the development of the system. Moreover, the development of an architecture specification focuses on ensuring that the produced architecture is represented from different perspectives to facilitate communication among various stakeholders.

3.2.2. Activities of the Architecting Process

The architecting process area encompasses all activities performed to produce artifacts constituting the architecture specification. An *architecture specification* describes the fundamental organization of components, the relationships among these components, the mapping of these components to their environment, and the principles and guidelines governing the system design and evolution [IEEE 2007].

The activities of the architecting process focus on:

- Identifying system components and establishing the relationships among them based on the system requirements.
- Creating an architecture that satisfies the requirements by either choosing an existing architecture, making up a composite architecture, or creating an architecture from scratch.
- Describing the architecture using a multi-view description approach and evaluating it to ensure that it meets the system requirements.

The architecting process typically follows a systems perspective [Kossiakoff and Sweet 2003]. It looks at the system as a whole. Externally, the architecting process takes into account interfacing systems, the operating environment, and the users of the system. Internally, it focuses on the architectural arrangements that represent the system. Therefore, the architecture specification must include software, hardware, and human (when appropriate) components.

3.2.3. Performers of the Architecting Process

The architecting process is carried out under different roles. The roles and job titles of architecting practitioners vary by organization and domain. However, these roles can be categorized as follows:

- A single architect who is solely responsible for the entire architecting process.
- A team of architects (also referred to as the architecture team) who collaborate throughout the entire architecting process. In this case, there is often a lead architect who manages the team.
- A single architect who leads a team of engineers (for requirements, hardware, network, database, usability, security, etc.) throughout the entire architecting process.

We use the term "architect" to refer to any of the roles listed above.

3.2.4. Uses of Architecture Specification

An architecture specification is used by several stakeholders for various reasons including [DoDAF 2009a,b,c]:

- Making acquisition decisions about whole systems, system components, and technologies that can be used in the development.
- Discovering the true requirements of the system and refining existing ones.
- Creating an overall strategy for aligning independent systems into a common use or purpose (i.e., system of systems).
- Evolving the system by deploying new technologies, adding services, and replacing/retiring old capabilities.

Equally important, architecture specifications are used as means of communication between the architect and the stakeholders, and among stakeholders. Therefore, architecture specification is produced from multiple perspectives (viewpoints) to enable such communication since stakeholders often have disparate stakes in the architecture.

CMMI for Development defines a process area by identifying the specific goals of that area and the associated specific practices that enable the realization of those goals. We follow the CMMI for Development structure and describe the specific goals and practices of the architecting process area below.

3.3. SPECIFIC GOALS AND PRACTICES OF THE ARCHITECTING PROCESS

We define the process of architecting for software/system engineering in accordance with the CMMI for Development structure by using a set of specific goals (SGs) and their associated specific practices (SPs) as shown in Table 3-1. Each SG and SP is described below.

Table 3-1. Specific goals and practices of the architecting process

SG 1: Identify System Components
SP 1.1: Identify Architecturally-Relevant Requirements
SP 1.2: Decompose the System into Components
SP 1.3: Assess the Component Decomposition
SG 2: Establish Relationships among the Identified Components
SP 2.1: Establish Structural Relationships
SP 2.2: Establish Behavioral Relationships
SG 3: Create the Architecture
SP 3.1: Select an Existing Architecture
SP 3.2: Make Up a Composite Architecture
SP 3.3: Create an Architecture from Scratch
SG 4: Describe the Architecture
SP 4.1: Describe the Architecture from an All Perspective
SP 4.2: Describe the Architecture from a Capability Perspective
SP 4.3: Describe the Architecture from a Data and Information Perspective
SP 4.4: Describe the Architecture from an Operational Perspective
SP 4.5: Describe the Architecture from a Project Perspective
SP 4.6: Describe the Architecture from a Services Perspective
SP 4.7: Describe the Architecture from a Standards Perspective
SP 4.8: Describe the Architecture from a Systems Perspective
SG 5: Evaluate the Architecture
SP 5.1: Evaluate the Architecture Based on the Four Ps
SP 5.2: Evaluate the Architecture Following a Risk-Driven Approach
SP 5.3: Evaluate the Architecture Based on a Set of Scenarios

SG 1. Identify System Components

Goal Statement: *System components are identified to satisfy the requirements.*

The architecting process starts with identifying the software, hardware, and human (e.g., users, operators) components that will constitute the system under development. Human components may or may not be relevant parts of the architecture depending on the context of the system. These components should be

identified so as to satisfy the requirements and provide the mandated capabilities. To achieve this goal, the following specific practices are employed.

SP 1.1. Identify Architecturally-Relevant Requirements

Practice Statement: *Requirements that are germane to the architecture are identified to guide the architecting process.*

Before identifying the components that will make up the system architecture, the requirements that are germane to the architecture should be identified. Architecturally-relevant requirements are those that have far-reaching implications on the architecture and span more than one component of the system.

Architecturally-relevant requirements include the functional requirements that describe major capabilities of the system – capabilities without which the system cannot satisfy the purposes intended for its use. Such requirements are important to identify system components and the relationships among them.

Architecturally-relevant requirements also include non-functional requirements (also referred to as quality characteristics or attributes) such as dependability, interoperability, and performance. Such quality characteristics are germane to the architecting process because they affect the quality of the entire system.

Quality attributes often tradeoff. For instance, enhancing security by adding extra layers of authentication and data encryption introduces a communication overhead and therefore affects the performance negatively. Thus, it is essential to identify such tradeoffs among quality characteristics to ensure that proper architecture decisions are made when creating the architecture.

The outcome of this specific practice should consist of a list of architecturally-relevant requirements – functional and non-functional. In addition, this specific practice should produce a set of potential tradeoffs that may exist among the identified quality characteristics.

SP 1.2. Decompose the System into Components

Practice Statement: *The system is decomposed into components that satisfy the identified architecturally-relevant requirements.*

The system should be decomposed into software, hardware, and human components based on the requirements that the architecture must satisfy. A component is a conceptual representation of an identifiable part of a system [Kossiakoff and Sweet 2003]. A component can be a software or hardware

module that encapsulates a set of related functions, data, or responsibilities associated with a user or operator.

A component is identified by mapping a function (or a set of related functions) that the system must perform into a conceptual representation that will carry out that function (or those related functions). Each component should be given a name that describes its functions. Techniques such as modularization, separation of concerns, and information hiding should be applied in component identification.

The quality of a component is determined by examining two characteristics: Cohesion and coupling. In principle, a component should be put together to have the highest possible cohesion and the lowest possible coupling. *Cohesion* is the degree to which the component's elements are related to each other. It depicts the "togetherness" among the elements comprising the component. A component with high cohesion directs all its elements towards achieving a single objective. *Coupling*, on the other hand, is the degree to which a component is built based on the logic of another component. Low coupling is desired, implying that there is minimal or no logical dependence among the components. When components are created with the highest possible cohesion and the lowest possible coupling, several architecture quality characteristics are enhanced, namely maintainability and modifiability.

Decomposition or modularization of a system into components continues until the component at the leaf node (i.e., the one that is no further decomposed) possesses the highest possible cohesion and the lowest possible coupling. Two approaches are commonly used to carry out this decomposition: Vertical and horizontal.

Vertical decomposition slices the system vertically by decomposing it in a top-down fashion. It starts by decomposing the system into components at level 1. A level 1 component is selected for decomposition into components at level 2. Then, a level 2 component is selected for decomposition into components at level 3. This decomposition continues until reaching the leaf component that has the highest possible cohesion and the lowest possible coupling.

Horizontal decomposition modularizes the system level by level. It starts by decomposing the system into all of the components at level 1. Each level 1 component is further decomposed into components at level 2. Then, each level 2 component is further decomposed into components at level 3. All of the components at a level must be identified before proceeding to the next level. When a component is identified as

possessing the highest possible cohesion and the lowest possible coupling, it is designated as a leaf component and is not decomposed further.

Both vertical and horizontal decomposition approaches can be intertwined for the same architecting project.

SP 1.3. Assess the Component Decomposition

Practice Statement: *Decomposition of the system into components is assessed against all architecturally-relevant requirements.*

Component decomposition should be assessed based on coverage and depth. The identified components should cover all of the requirements identified in SP 1.1. A matrix should be created to map each architecturally-relevant requirement to a component (or a set of components) that addresses that requirement. Gaps identified in this mapping should be addressed by either mapping a requirement to an already-identified component or by identifying a new component. Identifying a new component to address a particular requirement should follow the guidelines outlined in SP 1.2.

Component decomposition should also be assessed based on depth. Decomposition depth is evaluated by examining abstraction level and cohesion and coupling of each component. If a component is found to have an undesirable amount of cohesion and/or coupling, the principle of *separation of concerns* [Dijkstra 1982] should be applied to determine whether it should be decomposed further or combined with another component.

This specific practice ensures that each architecturally-relevant requirement can be addressed by the identified components and that each component is properly created.

SG 2. Establish Relationships among the Identified Components

Goal Statement: *Relationships among identified components are established.*

Relationships among the identified components should be established. A relationship defines how two components relate to each other in the context of the system being architected. Many perspectives exist from which a relationship can be established; for example, based on structure or behavior.

The Specific Practices that enable the accomplishment of the goal stated above are described below.

SP 2.1. Establish Structural Relationships

Practice Statement: *Structural relationships among identified components are established.*

Structural relationships represent how the components can be structurally organized to form a unified blueprint of the system. This perspective focuses on the static layout of these components, which can be either a logical or a physical layout.

The logical layout of components is concerned with the conceptual grouping of these components. The following are example types of logical relationships:

- Dependency: Component A *depends-on* Component B.
- Use: Component A *uses* Component B.
- Hierarchy: Component A *is-part-of* Component B or Component A *is-a-child-of* Component B.

The physical layout of components is concerned with relationships among software, hardware, infrastructure, and deployment components. The following are example types of physical relationships:

- Interfacing: Relationships that highlight the interfaces between two hardware components, between a software component and a hardware component, or between a software/hardware component and a human operator/user.
- Deployment: Relationships that highlight, among other aspects, the distribution of components over the network, resource sharing, and communication channels.

SP 2.2. Establish Behavioral Relationships

Practice Statement: *Behavioral relationships among identified components are established.*

Behavioral relationships among components represent interactions. This perspective focuses on the dynamic behavior of the system during execution. Behavior can be represented in many ways including processes, events, and services.

- Process: Component A starts/pauses/stops a process X.
- Service: Component A sends a request to Component B and Component B processes the request and may send a reply back.

- Event: Component A publishes a notification and one or more components subscribe to receive that notification.

Structural (static) and behavioral (dynamic) relationships help in the identification of a suitable architecture. For instance, the decomposition of the system into components may reveal that some components are responsible for producing data or performing tasks for other components. In such a case, the client-server architecture may seem plausible as a candidate architecture for the system under development. Therefore, the set of relationships identified in this specific practice provides the foundation to begin the creation of the system architecture.

SG 3. Create the Architecture

Goal Statement: *System architecture is created based on the identified components and the relationships among them.*

The identified components and the relationships that exist among them form the basis of the overall architecture. Architecture creation, therefore, consists of making decisions about which architecture(s) best satisfies the requirements of the system.

Three approaches to making architecture decisions exist: (1) select an existing architecture, (2) make up a composite architecture from existing ones, or (3) create an architecture from scratch.

SP 3.1. Select an Existing Architecture

Practice Statement: *An existing architecture is selected as the overarching architecture of the system.*

The major architectures that have been in use for architecting a software-based network-centric system of systems include: Client-Server Architecture, Distributed Objects Architecture, Peer-to-Peer Architecture, and Service-Oriented Architecture. For a given architecting task at hand, one of these architectures can be selected if it satisfies the requirements of the system under development.

- **Client-Server Architecture**

The client-server architecture (CSA) consists of the following five conceptual layers designated with circled numbers in Figure 3-1:

Layer 1. *Data Source Layer* (e.g., a relational database management system)

Layer 2. *Data Mapping Layer* (e.g., Entity Enterprise Java Beans, ActiveX Data Objects)

Layer 3. *Business / Application Logic Layer*

Layer 4. *Web Container Layer* (consisting of controller/mediator and server-side presentation preparation components)

Layer 5. *Client Presentation Layer* (e.g., Asynchronous JavaScript and XML (AJAX), Cascading Style Sheets (CSS), Document Object Model (DOM), Extensible HTML (XHTML), Extensible Markup Language (XML), Extensible Stylesheet Language Transformation (XSLT), HyperText Markup Language (HTML))

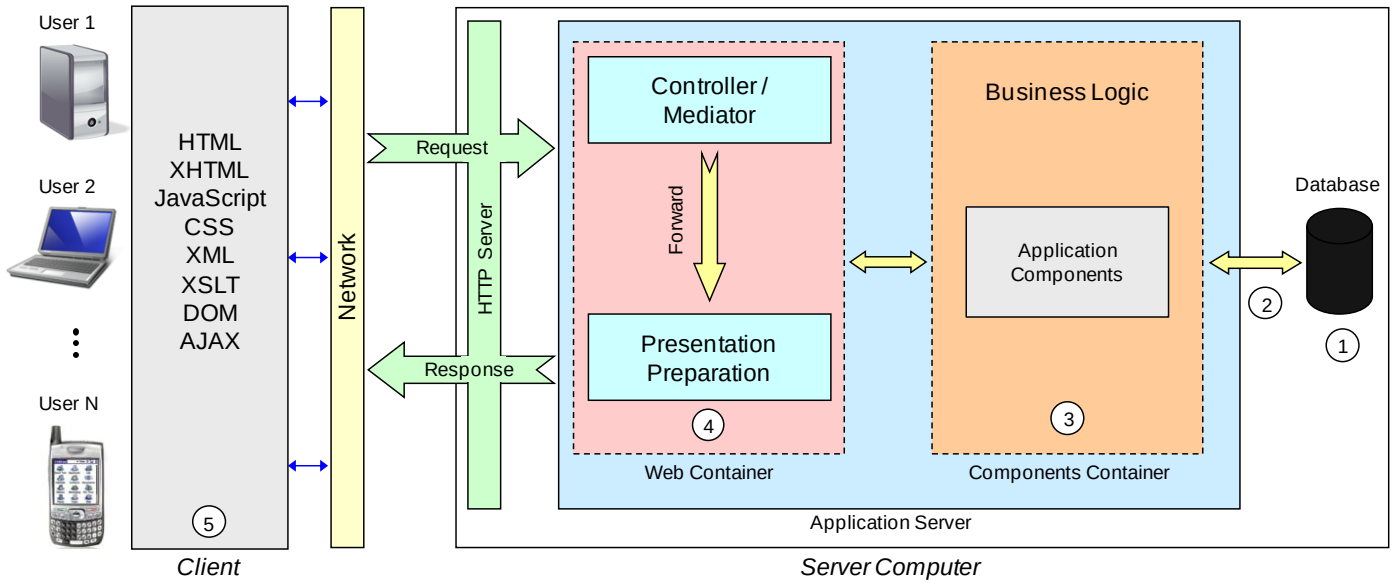


Figure 3-1. Conceptual layers of the client-server architecture

Two major industry standards can be employed to build a system under CSA: Java Platform, Enterprise Edition (Java EE) [Oracle 2010] and Microsoft .NET Framework [Microsoft 2010b].

- **Distributed Objects Architecture**

Figure 3-2 shows a conceptual representation of the distributed objects architecture (DOA).

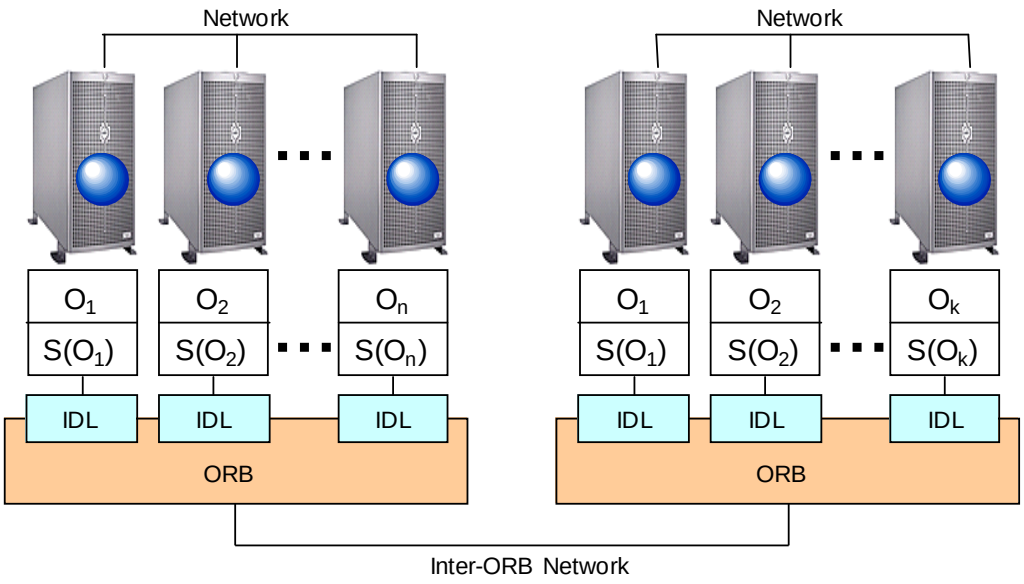


Figure 3-2. Conceptual representation of the distributed objects architecture

DOA consists of objects, which are software applications that run on distributed computers with different hardware and operating systems. An object O_j can act as a server and provide services $S(O_j)$ or act as a client and consume services as depicted in Figure 3-2. The communication among objects is mediated by the object request broker (ORB). Objects can be implemented in different programming languages and can be integrated with ORB using an interface definition language (IDL). A system created under DOA can be connected to another DOA-based system using the Inter-ORB communications over a network to form a system of systems.

Two major industry standards can be employed to build a system under DOA: Common Object Request Broker Architecture (CORBA) [OMG 2010] and Distributed Component Object Model (DCOM) [Microsoft 1996].

- **The Peer-to-Peer Architecture**

The peer-to-peer (P2P) architecture consists of a set of networked computers, where each computer makes some of its resources (e.g., computational power, storage, music files, and technical documents) directly available to other computers on the network as depicted in Figure 3-3. A P2P computer (node) acts as a supplier or consumer of resources.

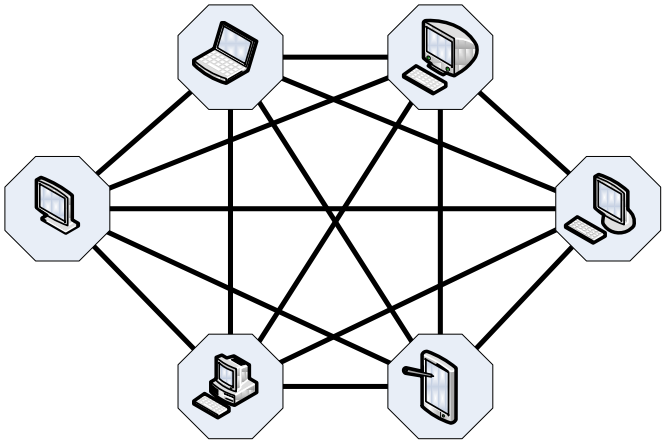


Figure 3-3. Conceptual representation of the P2P architecture

- **Service-Oriented Architecture**

SOA consists of the conceptual layers as depicted in Figure 3-4 [Glass 2008; Balci and Ormsby 2008]. Layers 3 to 6 in Figure 3-4 make up the major components of SOA built on top of the network structure consisting of layers 0 to 2.

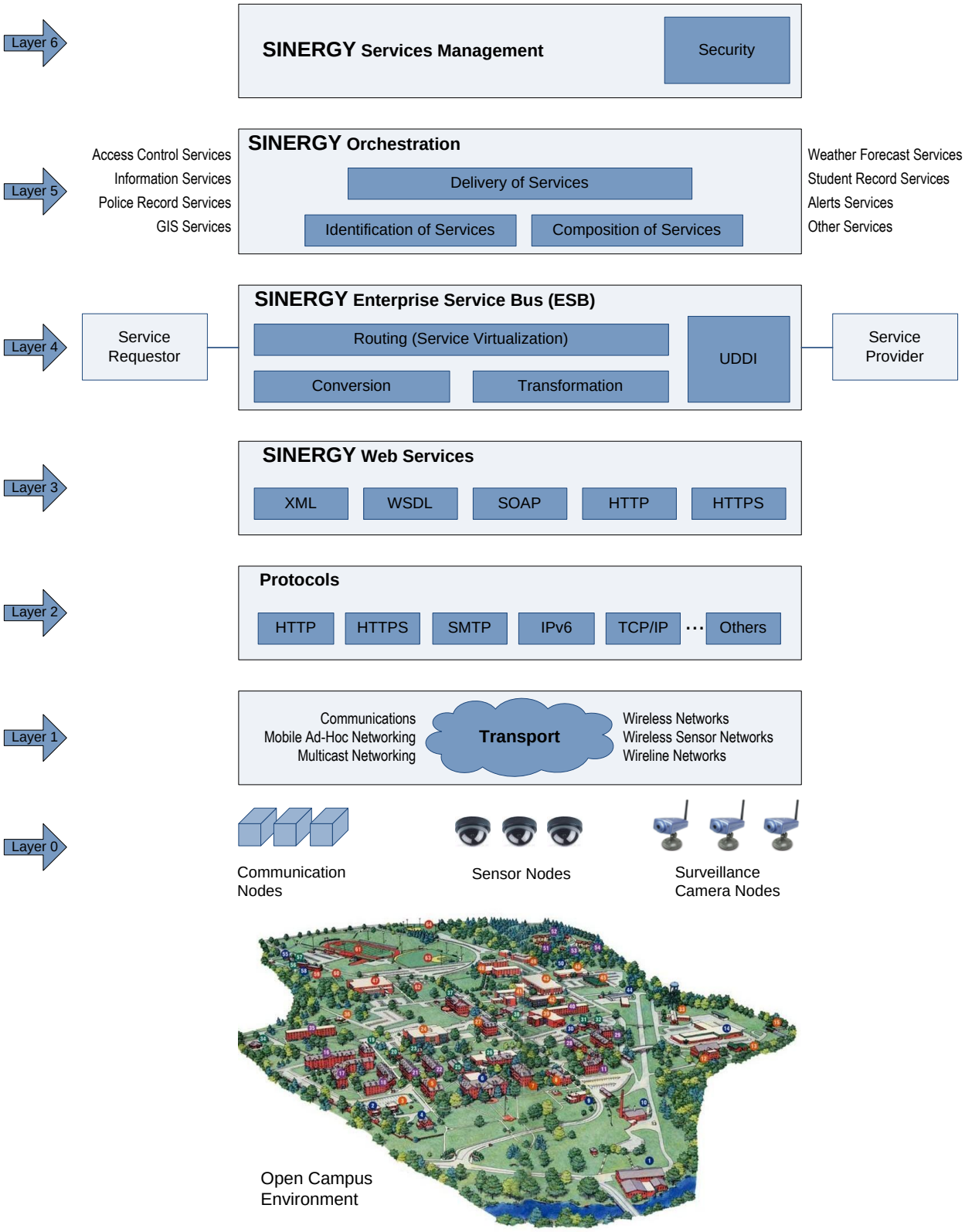


Figure 3-4. Conceptual layers of SOA

SP 3.2. Make Up a Composite Architecture

Practice Statement: *Two or more known architectures are selected to make up a composite architecture of the system.*

The second approach to architecture creation is to combine a number of known architectures to create a composite architecture for the system under development. Known architectures include the ones described in SP 3.1. In this context, known architectures are used to describe various parts of the system. The result is an overall architecture that is made up of several known architectures.

Figure 3-5 shows an example composite architecture of a system consisting of CSA and DOA. This is the architecture of a simulation development environment intended for use by geographically-dispersed users over the network. The overall architecture is CSA. However, the CSA's business/application logic layer is developed using the CORBA Component Model (CM) to satisfy the requirement for a heterogeneous execution environment. Thus, a composite architecture is created by combining CSA with DOA.

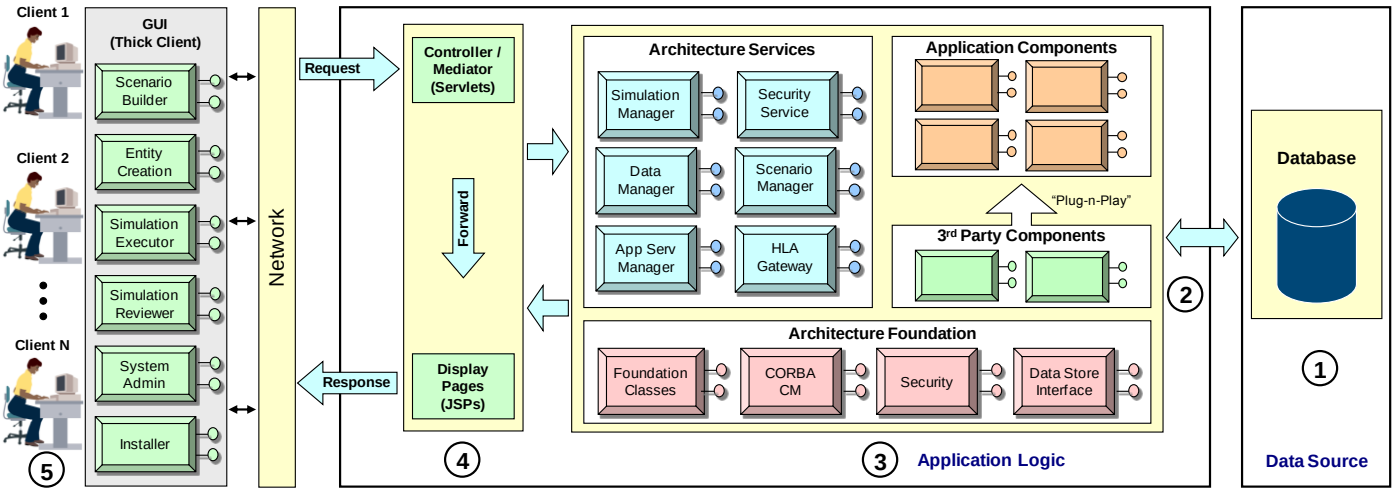


Figure 3-5. A composite architecture consisting of CSA and DOA

SP 3.3. Create an Architecture from Scratch

Practice Statement: *A new architecture is created from scratch based on the identified components and relationships.*

The third approach to architecture development is to create an architecture from scratch. In such a case, the architect uses the identified components and their relationships as the building blocks of the new

architecture. Therefore, the main purpose in this case is to come up with a novel organization of these components.

This approach is suitable in cases where known architectures do not provide the appropriate structure needed to satisfy the requirements. Such architectures are sometimes called custom-made or domain-specific architectures [Agrawala *et al.* 1992].

Custom-made architectures should follow basic guidelines such as the following.

- The new architecture should propose a new structure that is not represented by a known architecture.
- The new structure should specify the types of components, the types of relationships among them, and should provide guidelines specific to how the architecture can be instantiated into a particular design.
- The new architecture should be applicable to create architectures for similar systems. This is because architectures provide reusable abstractions for a particular class of problems. A new architecture should provide a new architecture solution to a class of systems.

SG 4. Describe the Architecture

Goal Statement: *Architecture description consists of a set of models representing multiple viewpoints of the system.*

A network-centric system of systems architecture is a complex entity that cannot be described using only one representation. We slice this complex entity in many different ways and come up with a representation for each slice. DoDAF [2009a,b,c] provides 52 representations (models) under the following eight viewpoints (perspectives) to describe an architecture.

1. All Viewpoint (AV)
2. Capability Viewpoint (CV)
3. Data and Information Viewpoint (DIV)
4. Operational Viewpoint (OV)
5. Project Viewpoint (PV)
6. Services Viewpoint (SvcV)
7. Standards Viewpoint (StdV)
8. Systems Viewpoint (SV)

No one viewpoint can properly represent the entire architecture of a system. The best practice is to describe the architecture under multiple viewpoints using a number of representations (e.g., DoDAF-described models) under each viewpoint. Which viewpoints and which representations in each viewpoint to use depend on the requirements specification given for a particular system [IEEE 2007; DoDAF 2009a,b,c; Clements *et al.* 2003; Kruchten 1995].

The Specific Practices that enable the accomplishment of the goal stated above are described below.

SP 4.1. Describe the Architecture from All Viewpoint

Practice Statement: *Create models that describe aspects of the architecture that are pertinent to all viewpoints.*

The all viewpoint (AV) is concerned with information pertinent to the entire architecture specification.

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- AV-1 Overview and Summary Information
- AV-2 Integrated Dictionary

DoDAF AV models describe the architecture effort with its scope, context, and findings, and defines a common terminology (definitions of terms) used throughout the entire architecture specification.

SP 4.2. Describe the Architecture from Capability Viewpoint

Practice Statement: *Create models that describe the architecture from capability viewpoint.*

The capability viewpoint (CV) is concerned with the description of the capabilities provided by the system. A *capability* is "the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks" [DoDAF 2009b, p. 80].

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- CV-1 Vision
- CV-2 Capability Taxonomy
- CV-3 Capability Phasing
- CV-4 Capability Dependencies

- CV-5 Capability to Organizational Development Mapping
- CV-6 Capability to Operational Activities Mapping
- CV-7 Capability to Services Mapping

DoDAF CV models are used to represent the architecture by describing:

- a strategic context for the capabilities,
- a hierarchy (taxonomy) of capabilities,
- planned achievement of capabilities at different points in time,
- dependencies between planned capabilities,
- logical groupings of capabilities,
- fulfillment of capability requirements,
- a mapping between the capabilities required and the operational activities that those capabilities support, and
- a mapping between the capabilities and the services that these capabilities enable.

SP 4.3. Describe the Architecture from Data and Information Viewpoint

Practice Statement: *Create models that describe the architecture from data and information viewpoint.*

The data and information viewpoint (DIV) is concerned with conceptual, logical, and physical levels of abstraction in representing the operational and business data and information.

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- DIV-1 Conceptual Data Model
- DIV-2 Logical Data Model
- DIV-3 Physical Data Model

DoDAF DIV models describe high-level data concepts and their relationships, data requirements and structural business process rules, and physical implementation format.

SP 4.4. Describe the Architecture from Operational Viewpoint

Practice Statement: *Create models that describe the architecture from operational viewpoint.*

The operational viewpoint (OV) is concerned with the overall context within which the system operates and the external systems with which the system interacts. Describing the architecture from this viewpoint requires capturing all system elements in one or more models at a high level of abstraction.

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- OV-1 High-Level Operational Concept Graphic
- OV-2 Operational Resource Flow Description
- OV-3 Operational Resource Flow Matrix
- OV-4 Organizational Relationships Chart
- OV-5a Operational Activity Decomposition Tree
- OV-5b Operational Activity Model
- OV-6a Operational Rules Model
- OV-6b State Transition Description
- OV-6c Event-Trace Description

DoDAF OV models are used to represent the architecture by describing:

- a high-level graphical operational concept,
- resource flows exchanged between operational activities,
- resources exchanged and the relevant attributes of the exchanges,
- organizational context, role or other relationships among organizations,
- capabilities and operational activities organized in a hierarchal structure,
- context of capabilities and operational activities and their relationships among activities, inputs, and outputs,
- business rules that constrain operations,
- business process (activity) responses to events, and
- traces of actions in a scenario or sequence of events.

OV models serve as a means of providing the big picture of what the architecture is and what the system is supposed to do. In addition, OV models form the architecture basis for communication among non-technical stakeholders such as managers, customers, and decision makers.

SP 4.5. Describe the Architecture from Project Viewpoint

Practice Statement: *Create models that describe the architecture from project viewpoint.*

The project viewpoint (PV) is concerned primarily with architecture models that describe project management activities. This viewpoint addresses assignments of development efforts and project planning.

DoDAF [2009a, b, c] provides the following models to represent the architecture from this viewpoint:

- PV-1 Project Portfolio Relationships
- PV-2 Project Timelines
- PV-3 Project to Capability Mapping

DoDAF PV models focus on the following aspects of the architecture:

- *Development:* Models in this category describe how architecture components should be assigned to development teams. They also highlight the interdependencies among the components to ensure that proper planning of milestones is achieved and that development bottlenecks are avoided.
- *Planning:* Models in this category describe aspects of the system that relate to project management such as cost estimation, commercial-off-the-shelf (COTS) components, and release planning.

SP 4.6. Describe the Architecture from Services Viewpoint

Practice Statement: *Create models that describe the architecture from services viewpoint.*

The services viewpoint (SvcV) is concerned with describing the services that provide access to system capabilities. In general, a service is a means to access a system capability [DoDAF 2009b], and has a predefined description of how it can be invoked [Erl 2008].

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- SvcV-1 Services Context Description
- SvcV-2 Services Resource Flow Description
- SvcV-3a Systems-Services Matrix
- SvcV-3b Services-Services Matrix
- SvcV-4 Services Functionality Description
- SvcV-5 Operational Activity to Services Traceability Matrix
- SvcV-6 Services Resource Flow Matrix
- SvcV-7 Services Measures Matrix
- SvcV-8 Services Evolution Description
- SvcV-9 Services Technology and Skills Forecast
- SvcV-10a Services Rules Model
- SvcV-10b Services State Transition Description
- SvcV-10c Services Event-Trace Description

DoDAF SvcV models are used to represent the architecture by describing:

- services, service items, and their interconnections,
- resource flows exchanged between services,
- relationships among or between systems and services,
- relationships among services,
- functions performed by services and the service data flows among service functions (activities),
- mapping of services (activities) back to operational activities (activities),
- service resource flow elements being exchanged between services and the attributes of that exchange,
- measures (metrics) of services,
- evolution of services over time, and
- rules, transition, and trace of services.

SP 4.7. Describe the Architecture from Standards Viewpoint

Practice Statement: *Create models that describe the architecture from standards viewpoint.*

The standards viewpoint (StdV) is concerned with the documentation of the required standards used and that must be adhered to during the development and operation of the system.

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- StdV-1 Standards Profile
- StdV-2 Standards Forecast

DoDAF StdV models describe operational, business, technical, and industry policies, standards, guidance, constraints, and forecasts that are applicable to capability and operational requirements, system engineering processes, systems, and services.

SP 4.8. Describe the Architecture from Systems Viewpoint

Practice Statement: *Create models that describe the architecture from systems viewpoint.*

The systems viewpoint (SV) is concerned with describing component systems that make up the overall system. This viewpoint is relevant for the architecture specification of systems of systems, network-centric systems, and families of systems.

DoDAF [2009a,b,c] provides the following models to represent the architecture from this viewpoint:

- SV-1 Systems Interface Description
- SV-2 Systems Resource Flow Description
- SV-3 Systems-Systems Matrix
- SV-4 Systems Functionality Description
- SV-5a Operational Activity to Systems Function Traceability Matrix
- SV-5b Operational Activity to Systems Traceability Matrix
- SV-6 Systems Resource Flow Matrix
- SV-7 Systems Measures Matrix
- SV-8 Systems Evolution Description
- SV-9 Systems Technology and Skills Forecast

- SV-10a Systems Rules Model
- SV-10b Systems State Transition Description
- SV-10c Systems Event-Trace Description

DoDAF SV models are used to represent the architecture by describing:

- systems, system items, and their interconnections,
- resource flows exchanged between systems,
- relationships among systems,
- the functions (activities) performed by systems and the system data flows among system functions (activities),
- a mapping of system functions (activities) back to operational activities (activities),
- a mapping of systems back to capabilities or operational activities (activities),
- system resource flow elements being exchanged between systems and the attributes of that exchange,
- measures (metrics) of systems, and
- evolution, rules, transition, and trace of systems.

SG 5. Evaluate the Architecture

Goal Statement: *The architecture is evaluated to determine how well it enables the system to exhibit the required quality characteristics.*

System architecture influences many system quality indicators (attributes or characteristics) such as adaptability, dependability, deployability, extensibility, interoperability, maintainability, modifiability, openness, performance, scalability, survivability, and testability [Balci and Ormsby 2008]. The goal of architecture evaluation is to find out how well an architecture enables a system to possess a desired set of quality characteristics under a given set of intended uses of the system.

Architecture evaluation should be conducted by using a methodology such as the ones listed below:

- MSAAM: Military System Architecture Assessment Methodology [Balci and Ormsby 2008]
- ATAM: Architecture Tradeoff Analysis Method [Bass *et al.* 2003]
- SAAM: Software Architecture Analysis Method [Kazman *et al.* 1994]
- CBAM: Cost-Benefit Analysis Method [Nord *et al.* 2003]
- ALMA: Architecture Level Modifiability Analysis [Lassing 2002]

- FAAM: Family-Architecture Analysis Method [Dolan 2002]

The Specific Practices that enable the accomplishment of the goal stated above are described below.

SP 5.1. Evaluate the Architecture Based on the Four Ps

Practice Statement: *The architecture is evaluated from four perspectives: product, process, project, and people.*

Architecture evaluation is considered to be a confidence building activity. The more comprehensive and detailed the evaluation is, the more confidence we can gain in the evaluation. Four major perspectives or four Ps influence the architecture quality: Product, process, people, and project. Architecture evaluation can be approached from any one of the four Ps, but a combination of all four provides the best balance and results in a much higher level of confidence in the evaluation. Therefore, an architecture should be assessed from the four perspectives by way of assessing: [Balci and Ormsby 2008]

1. the architecture itself as a product,
2. the process used in creating the architecture,
3. the quality of the people employed in creating the architecture, and
4. architecture development project characteristics (e.g., capability maturity, documentation, planning, risk management).

SP 5.2. Evaluate the Architecture Following a Risk-Driven Approach

Practice Statement: *A risk-driven evaluation approach is employed to ensure early identification and mitigation of risks.*

A risk-driven approach is advocated for architecture evaluation. The evaluation should consider all potential risks including the following: [Balci and Ormsby 2008]

- *Acceptance Risk* is the probability that the system architecture will not pass the acceptance test with respect to the prescribed acceptance criteria.
- *Integration Risk* is the probability that the system architecture components will not be successfully integrated.
- *Interface Risk* is the probability that the system architecture will not successfully interface with the required external systems.
- *Performance Risk* is the probability that the performance of the system to be built based on the proposed architecture will not be acceptable.

- *Reliability Risk* is the probability that the reliability of the system to be built based on the proposed architecture will not be acceptable.
- *Reproducibility Risk* is the probability that a component of the proposed system architecture cannot be reproduced. For example, if the component relies on a piece of specialized hardware and the specialized hardware is no longer in production, then the system may not be reproduced at a reasonable cost.
- *Supportability Risk* is the probability that the system to be built based on the proposed architecture cannot be properly maintained after its delivery. For example, there may be no technical support, no support for correcting errors, and no support for making improvements and upgrades.
- *Technological Risk* is the probability that one or more components of the proposed system architecture depend on undeveloped technologies that cannot be developed in an acceptable time frame.
- *Utility Risk* is the probability that the system to be built based on the proposed architecture will be less useful than required by the system stakeholders.

3.4. CONCLUDING REMARKS

Architecting should be designated as a process between the Requirements Engineering and Design processes in the life cycle models for software engineering as well as for systems engineering. The architecting process plays a critically important role in engineering a software-based network-centric system of systems.

Architecting a network-centric enterprise system, a system of systems, or a family of systems poses significant complexity. The complexity can be overcome by way of modularization. The DoD architecture framework decomposes the architecture representation into eight viewpoints (perspectives) and defines a total of 52 models to represent the architecture under these viewpoints. Which models to build for a particular project depends on the requirements specification document. Certainly, developing all 52 models is an onerous task requiring significant resources and time.

CHAPTER 4: SINERGY CAPABILITIES SPECIFICATION

The history of university, college, and high school campuses is eventful with man-made tragedies ensuing a tremendous loss of life and property. Virginia Tech's April 16 tragedy ignited the discussion about balancing openness and safety in open campus environments. Decision makers in educational institutions face the challenge of ensuring that their campuses provide an open learning environment and at the same time are safe. Most deployed technology solutions are characterized by addressing bits and pieces of the problem (e.g. ENS). Without a comprehensive understanding of the requirements for an institution-wide solution that facilitates augmented SA and efficient emergency response, the proposed solutions fall short of the desired outcome. This chapter provides a capabilities specification for such a solution. The capabilities specification serves as a case study and a framework that guide the architecting, design, and implementation of decision support systems that facilitate campus ERM.

4.1. INTRODUCTION

Academic institutions operating in open campuses are facing an increasing challenge to protect their constituents while maintaining the openness nature of their campuses. A major aspect of this challenge is enabling administrators and first responders anticipate as well as respond effectively to incidents. On April 16, 2007, Virginia Tech was a scene for the most tragic shootings in American school history leaving 32 students and faculty dead and 17 others wounded [VT 2007]. In the wake of this tragedy, several issues were identified regarding the ability of schools to maintain a balance between safety and openness. To study the events leading to this incident, a panel called the Virginia Tech Review Panel was convened, the purpose of which was to investigate and to assess the effectiveness of the response by decision makers and public safety entities during and after the incident.

---

* The logo is created by graphic designer, Elias Layachi (Used with permission, 2011)

Notably, a key finding of the Virginia Tech Review Panel was that "the university did not respond effectively" in spite of the fact that various individuals and entities within and outside the university knew pieces of relevant information. The panel cites, however, that the reason for this failure is that "no one knew all the information and no one connected all the dots" [VT Review Panel 2007]. In other words, the study blames the failure on a lack of a decision support mechanism that would have enabled decision makers to *connect the dots*. The unfortunate April 16 tragedy presents a clear case for the need for an integrated decision support system that enables groups of decision makers to "connect the dots" through interoperable communication and information sharing facilities of a Group Decision Support System (GDSS). This finding is supported by other studies that dealt with ERM to man-made disasters [FEMA 2010b].

A GDSS plays a critical role in the success of an organization and provides essential support to the decision-making process at all levels, including planning, operations, and management [Turban *et al.* 1997]. Nevertheless, the acquisition influx of ENS by educational institutions in the aftermath of Virginia Tech's shootings reflects a typical reaction to solve a multi-faceted problem by using off-the-shelf, one-size-fits-all technology. Unfortunately, available technologies, such as ENS, provide only bits and pieces of the desired solution. Addressing this issue needs to consider many technical, legal, and social aspects. In this chapter, we address this need by examining the role of software-based technology in establishing a safe campus environment through the integration of disparate information systems, databases, departments, and agencies around campus.

This study is the basis for our larger research project, which proposes to develop an SOA-based architecture for a GDSS called SINERGY: campu**S** s**I**tuational aware**N**ess and **E**mergency **R**esponse mana**G**ement s**Y**stem. SINERGY is a software-based, network-centric system of systems intended to provide SA, SC, and ERM for open campus environments. To accomplish this, the project is structured into four phases: Problem formulation, capabilities specification, architecture specification, and architecture evaluation. In this chapter, we report on the first two phases of the research, which consisted of a study we conducted within the Virginia Tech campus to perform a domain analysis about a GDSS that supports ERM for open campus environments. We define the problem and propose a capabilities specification of the GDSS based on interviews that we conducted with the stakeholders so that the system capabilities reflect the user requirements.

The results of the study contribute to the design science of GDSS [DeSanctis and Gallupe 1987]. Design science is concerned with creating various life cycle artifacts of a system under development [Simon

1981]. Artifacts of design science are of four categories: Constructs, models, methods, and implementations [March and Smith 1995]. This paper presents a system capabilities specification – a contribution that falls under the category of constructs. Constructs form a set of concepts that enables developers to analyze, characterize, and communicate about a system context. The capabilities specification presented herein provides a framework for a GDSS that supports campus ERM and serves as an input to the architecting and design of such systems. This framework describes *key capabilities* and *quality attributes* that should be present in any GDSS for campus ERM. Hence, the framework guides the future development of various types of GDSS for campus ERM.

The remainder of this chapter is organized as follows. Section 4.2 presents the research approach. Section 4.3 describes the types of stakeholders of SINERGY. Section 4.4 describes stakeholders' functionality needs along with the SINERGY capabilities that satisfy those needs. Section 4.5 describes SINERGY quality attributes and Section 4.6 lists the intended uses (IUs) of SINERGY. Finally, we present our concluding remarks in Section 4.7.

4.2. RESEARCH APPROACH

Figure 4-1 shows the four phases of our research and the first two phases that are the focus of this chapter. The ultimate goal of this work is to develop an architecture for a GDSS that will facilitate coordinated decision making and address the issue of balancing safety and openness for campuses. To do so, however, we need to understand what it is that we plan to architect. For this purpose, we need to formulate the problem concisely taking into consideration social and legal constraints, and elicit the functional and quality needs that will guide the architecting process. Therefore, our approach consists primarily of:

1. involving campus constituents including students, staff, administrators, law enforcement, and first responders in specifying the desired solution;
2. studying the characteristics of open campuses and the legal guidelines that govern them;
3. surveying the technologies that best support the achievement of the desired solution; and
4. deriving and specifying functional and quality requirements that guide the architecting, design, and development of such a system.

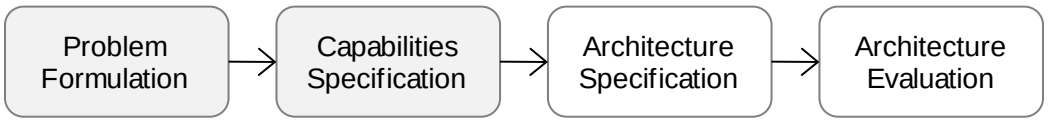


Figure 4-1. Phases of our research

4.2.1. Problem Formulation

What it means to be safe in an open campus differs from one institution to another, from one administrator to another, and from one law enforcement agency to another. However, commonalities exist across these differences. Figure 4-2 illustrates how we formulate the problem of campus safety based on our comprehensive study of the Virginia Tech campus environment. We view this problem from three complimentary prisms: SA, SC, and ERM. In other words, a safe and open campus environment can be realized through a GDSS that enables the creation of a COP of the campus environment shared by all campus entities (i.e., SA). Having a COP of what goes on campus at any point in time is key for law enforcement personnel to put in place effective SC measures in real-time decision-making context [Burstein *et al.* 2011]. Finally, common SA and deliberate SC lay the foundation for an efficient and effective emergency response in the case of an emergency.

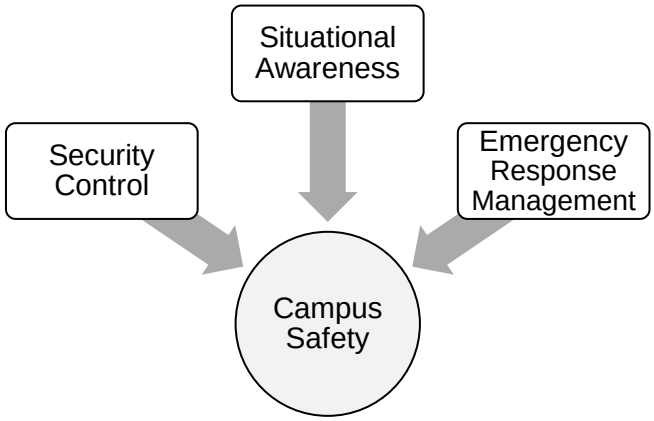


Figure 4-2. Elements of safety in open campuses

4.2.2. Capabilities Specification

We elicit SINERGY capabilities by interviewing stakeholders who have expertise regarding campus safety and find initiatives to campus safety as critical. Virginia Tech's resources (students, staff, faculty, campus structure, and existing systems) provided the context for these interviews. The context of the study helped us focus on understanding the problem domain and deriving *key capabilities* and *quality attributes*. Figure 4-3 illustrates the process through which the capabilities and quality attributes were identified. In addition, this phase was guided by the opinions of SMEs in related technologies and research areas. Finally, our study was complimented by participating in the Incident Command System (ICS) training program offered by the Federal Emergency Management Agency (FEMA) Emergency Management Institute [ICS 2010]. This program provided us with the understanding of best practices in

emergency management, and the legal constraints that guide these practices. Our objective was to conceptualize practice to guide GDSS design for campus safety, so that the systems facilitate coordination and response in a real-time decision-making context that characterize emergency response situations.

Figure 4-3. Elicitation process of capabilities and quality attributes

Table 4-1 shows a list of SINERGY stakeholder categories. We interviewed a number of representatives from each of the categories listed as indicated by the number next to the stakeholder category. The interviews focused on the capabilities that SINERGY should provide as a potential solution to current security and emergency management challenges from their perspective.

Table 4-1. List of stakeholder categories

Stakeholder Category	Examples
Administrators (4)	University president, vice president, dean, department chair
Campus Constituents (34)	Faculty, staff, students, campus visitors
Security Providers (6)	University police, town police, residence hall directors/assistants
First Responders (7)	Firefighters, hospital staff, university emergency staff, health center staff
SMEs (11)	Wireless sensor network researchers, autonomous systems researchers, information management researchers, social media researchers, sociologists

4.2.3. Assumptions and Constraints

Although the capabilities and quality attributes described in this paper can be generalized to any GDSS that supports ERM, the research is conducted under certain assumptions and constraints that represent state-of-art technologies and clear standards. First, SINERGY shall be based on SOA using Web Services technology. Second, SINERGY architecture shall be represented by using DoDAF Version 2.0. Third, SINERGY architecture shall represent a “to-be” viewpoint of the system, i.e., a desired system to be built. Finally, the campus police department shall establish a dedicated SINERGY command center to manage its resources and serve as the command center for emergency response efforts. This link from

decision to action is critical given the GDSS real-time decision context and the importance of real-time action as a proper response to each incident.

4.2.4. Types of Emergencies

Based on the interviews with 62 stakeholders, we identified the scope of incidents for which SINERGY should be designed. In order to focus our study on the types of emergencies that may effect campus environments, we consider those emergencies that may result in potential threats to the life of those on campus. Figure 4-4 shows a toxonomy of the types of incidents for which SINERGY may be employed.

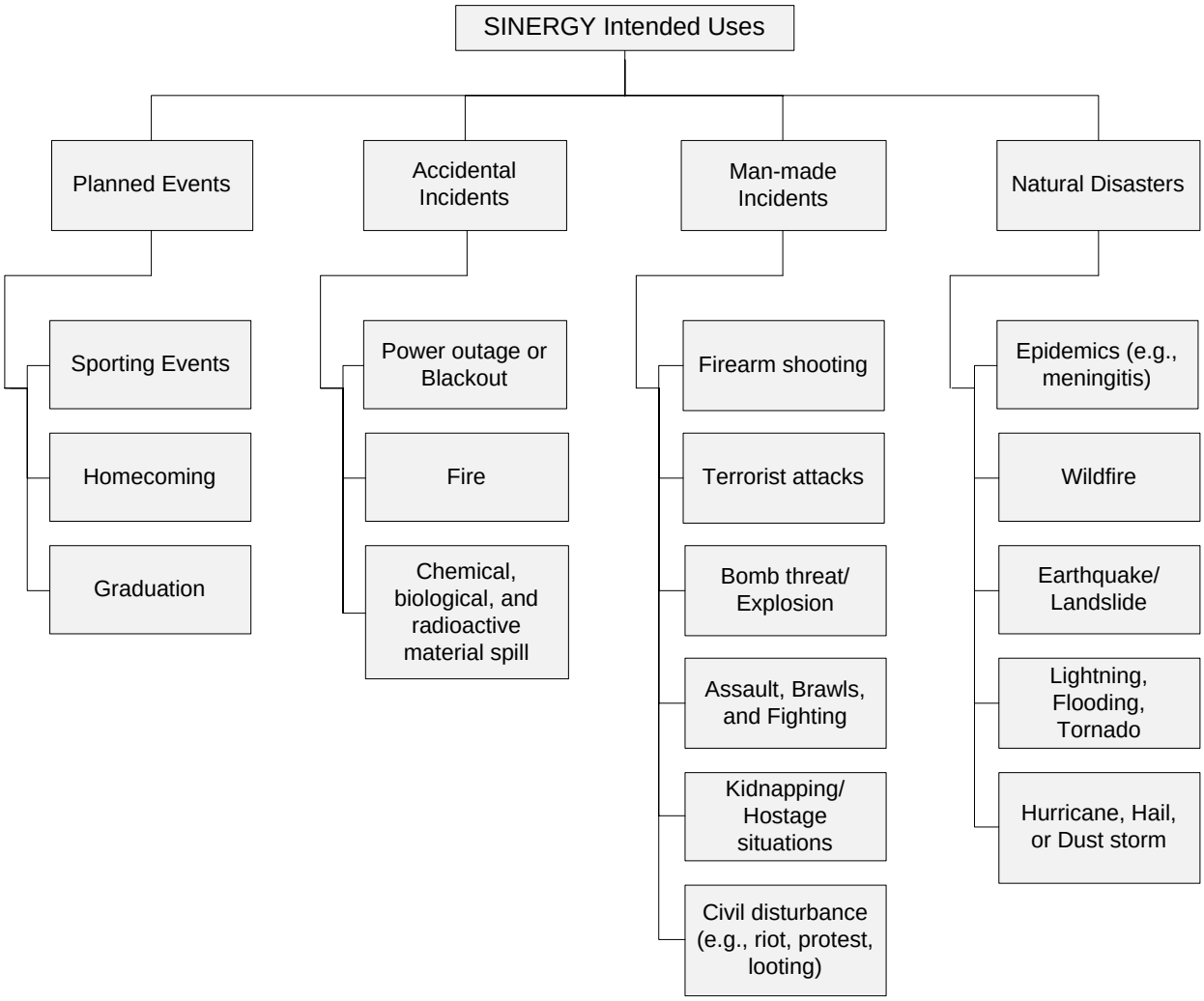


Figure 4-4. Types of incidents for which SINERGY can be used

4.3. STAKEHOLDER TYPES

Stakeholder analysis is essential to ensuring the relevance of any proposed solution. In this study, we identified several stakeholders who are likely to impact and be impacted by SINERGY. We classified these stakeholders into *passive* and *active* stakeholders.

4.3.1. Passive Stakeholders

Passive stakeholders are those who will not interact or use the system directly or actively once the system is operational. However, these stakeholders have a lot of input and interest (stake) in the system and they influence what quality characteristics the system should possess. This category of stakeholders includes sponsors and SMEs.

For SINERGY, Table 4-2 lists the passive stakeholders from which we received input.

Table 4-2. List of SINERGY passive stakeholders

Researchers in the Digital Libraries for Crisis, Tragedy, and Recovery Research Network, Department of Computer Science, Virginia Tech
Researchers in the Cultural Differences in Emergency Response, Department of Sociology, Virginia Tech
Researchers in the Use of Phones and Social Networks during Emergencies, Department of Accounting and Information Systems, Virginia Tech
Office director of Converged Technologies for Security, Safety, and Resilience, Virginia Tech
Medical doctors and administrator of the Schiffert Health Center, Virginia Tech
Researchers and administrator of the Emergency Planning in Biological and Chemical Research, Via College of Osteopathic Medicine (VCOM), Virginia Tech
Autonomous systems researcher, Institute of Critical Thinking & Applied Science, Virginia Tech
System architect from Communications Network Services, Alerts Systems, Virginia Tech
Experts in campus security, Q&A webinars organized by Campus Technology [Campus Technology 2010]

4.3.2. Active Stakeholders

Active stakeholders, on the other hand, are those who will actively and directly interact and use the GDSS once it becomes operational. Examples of these stakeholders include faculty, students, campus administrators, campus security, and law enforcement personnel.

For SINERGY, Table 4-3 lists the active stakeholders who participated in the interviews.

Table 4-3. List of SINERGY active stakeholders

Director of the Office of Emergency Management, Virginia Tech
Director and staff of the Office of Residential Life, Virginia Tech
Officer from the Virginia Tech Police Department, Crime Prevention Unit and Outreach
Vice president of University Relations, President's Office, Virginia Tech
Officers from the Blacksburg Police Department, Blacksburg, VA
Students from Virginia Tech campus: Full-time, part-time, undergraduate, graduate, on-campus, off-campus, Blacksburg campus, and extended campus

The list of stakeholders shows the diversity of individuals, groups, and entities that hold a stake in SINERGY. Given that a GDSS is designed for a multi-participant, multi-objective, and multi-criteria decision problem, adhering to these inputs is critical. Given the importance of these inputs, we elaborate on the functionality and quality needs of each stakeholder type that we derived from the interviews.

4.4. FUNCTIONALITY NEEDS: CAPABILITIES

Educational institutions, as any large enterprise, must comply with state and federal regulations in the preparation for emergencies and establishment of contingency plans for responding to threats that could harm personnel, faculty, students, and/or property. They are required to establish an Emergency Management office that works in tandem with other university offices to create response plans to emergencies not only for the short term but also for the long term. The director of this office serves as the campus emergency response manager in case of an incident involving the university.

Educational institutions should also comply with the ICS, which is FEMA's recommended emergency management framework that guides response plans to emergencies [ICS 2010]. Within this system, the director of the Emergency Management office serves as the Incident Commander (IC) and is responsible for assembling and managing all the agencies that need to collaborate in response to an incident.

Based on the interviews conducted with SINERGY stakeholders, a list of functionality needs have been identified. These functionality needs are specified in terms of capabilities. A capability is a broader concept than a requirement, and refers to what a system provides but will not elaborate on the details of

how it is provided. Formally, a *capability* is defined as "the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks" [DoDAF 2009b, p. 80].

Table 4-4 lists stakeholders' functionality needs. This list covers the overarching needs expressed by more than one stakeholder. The list represents a starting point for the identification of SINERGY capabilities and the SOA architecture facilitates expending functionalities to build on the initial design. Each need represents a high-level specification of a capability or a set of capabilities that the stakeholders would like SINERGY to provide. These needs are grouped by the three components of SINERGY's overall goal: SA, SC, and ERM.

Table 4-4. List of stakeholders' functionality needs

Security Control
Diverse incident detection capabilities
Diverse incident reporting capabilities
On-demand surveillance capabilities
Situational Awareness
Common operational picture capabilities
Geographical Information System and campus visualization capabilities
Audible situational awareness capabilities
Visual situational awareness capabilities
Electronic text situational awareness capabilities
Emergency Response Management
Direct communication with emergency response personnel capabilities
Response coordination capabilities
Documentation capabilities for decisions, expenses, and damages
Training capabilities for personnel and public

4.4.1. Diverse Incident Detection Capabilities

One of the major needs, as stated by several law enforcement personnel, from a system such as SINERGY is to "*help them anticipate incidents before they happen and detect them as they unfold.*" Diverse detection capabilities are essential to enabling SC personnel do their jobs effectively.

Capability Statement: SINERGY shall provide the capability of managing a video-based surveillance network consisting of multiple cameras deployed across campus, operated by the campus police department.

Capability Statement: SINERGY shall provide the capability of managing a network of sensors deployed around critical infrastructure facilities on campus, operated by managers of those facilities.

Capability Statement: SINERGY shall provide analysis capabilities of recorded and sensed data to detect security warnings and anticipate incidents before they occur. Security experts within campus law enforcement shall manage these capabilities.

4.4.2. Diverse Incident Reporting Capabilities

SINERGY is expected to enable the reporting of incidents to security personnel. Diverse reporting capabilities are needed to enable both security providers and campus constituents "*share the responsibilities of making the campus safe*" by reporting incidents in a variety of ways.

Capability Statement: SINERGY shall provide students, faculty, and staff the capability to report incidents to an established SINERGY command center through the following means:

- Fax: a dedicated incident report fax number
- 911 Dispatch: cellular phone, landline phone, and VoIP (Voice-over-IP) applications
- E-mail: a dedicated incident reporting e-mail address that accepts e-mails with documents, photos, and videos as attachments
- Messaging: a dedicated number that accepts SMS (Short Message Service) and MMS (Multimedia Messaging Service) messages
- Online, location-based, and secure incident reporting portal
- Mobile, location-based, and secure incident reporting software application for mobile devices

Capability Statement: SINERGY shall provide law enforcement personnel the capability to report incidents to the command center through the means mentioned above in addition to:

- Police in-car computers
- Two-way radios (walkie-talkies)

4.4.3. On-demand Surveillance Capabilities

On-demand surveillance is important due to privacy concerns in campus environments. Locker rooms, classrooms, dining rooms, residential halls, and athletic facilities "*are personal areas and cannot be monitored all the time.*" However, during an emergency, it is critical that these areas are monitored to manage the response to an incident. Need-based surveillance is a solution to balancing safety and privacy.

> **Capability Statement:** The network of surveillance cameras shall be sensor-based. Sensor-based cameras shall record data only when it detects motion of certain types of objects (e.g., human or animal).
>
> **Capability Statement:** The network of surveillance cameras shall be IP-addressable. SINERGY shall provide the capability to remotely-manage (switch on and off) these cameras to record only on a need-basis (e.g., during an emergency or at nights).
>
> **Capability Statement:** SINERGY shall provide the capability to seamlessly integrate a set of newly deployed cameras at the scene of an incident. Law enforcement personnel shall be able to deploy portable surveillance cameras at the scene of an incident and register them into SINERGY to be part of the network of surveillance cameras.

4.4.4. Common Operational Picture Capabilities

A common operational picture (COP) is a consistent and accurate representation of the overall information to all constituents on campus. The role of Emergency Management director is "*to ensure that the most accurate picture possible of the campus is available to support the decisions of its constituents.*" To do so, an on-demand, interactive, graphical representation of the COP is needed for different SINERGY users personalized to their security roles, responsibilities, and required action responses. This will help each user to receive personalized information corresponding to their decision and action domains.

> **Capability Statement:** SINERGY shall provide a COP for a user's personalized decision-making domain.
>
> **Capability Statement:** The COP shall be accessible to users based on their access privileges. Capabilities required to generate the COP shall depend on the type of users and their access levels.

Capability Statement: The COP shall be based on the integration of information retrieved from various databases and data services on and off campus including: Student records information, police records, health records, maps and geographic information system (GIS) services, building schematics, and digitized campus maps.

Capability Statement: The COP shall have default instantiations, which shall be recomposable by the user based on user's specified parameters and preferences. For instance, the police shall be able to identify all cameras deployed on campus and activate them to locate a perpetrator or monitor a situation. Administrators shall be able to monitor campus walkways to direct traffic. Health officials shall be able to locate patients' residence and classrooms to reason about the contagiousness patterns of a particular virus.

4.4.5. GIS and Campus Visualization Capabilities

GIS is a critical enabler of SA. Interviews with a building maintenance manager expressed a need for such capability to help the sharing of consistent and up-to-date building information among different offices on campus.

Capability Statement: SINERGY shall use GIS technology to provide a centralized campus basemap allowing users to download, query, and edit content with proper access privileges.

For instance, the police can have access to the basemap including data about crime statistics, potential hideouts, and risk areas, to anticipate the next move of a loose shooter on campus. The campus health center can place current cases of flu-infected students on the basemap and draw conclusions on possible modes of transmission of this virus on campus. This capability will facilitate more timely responses to incidents.

Capability Statement: SINERGY shall provide visualization capabilities of the COP through the use of large displays, multi-displays, and touch-displays.

4.4.6. Audible Situational Awareness Capabilities

Audible SA allows campus constituents to know about ongoing incidents through audio technologies. SINERGY shall incorporate audio mechanisms to keep students, staff, and faculty informed about ongoing incidents.

Capability Statement: SINERGY shall provide audible notifications about campus events to all constituents through one or more of the following mechanisms:

- Loudspeakers
- Sirens
- Voicemail
- Fire alarm speaker system
- Text-to-speech (TTS) tools

4.4.7. Visual Situational Awareness Capabilities

Visual SA allows campus constituents to know about ongoing incidents through visual technologies. SINERGY shall incorporate visual mechanisms to keep students, staff, and faculty informed about ongoing incidents.

Capability Statement: SINERGY shall provide visual notifications about campus events to all constituents through one or more of the following mechanisms:

- Digital Signage
- In-class digital boards
- Siren lights
- Fire alarm system lights

4.4.8. Electronic Text Situational Awareness Capabilities

Electronic text (E-text) SA allows campus constituents to know about ongoing incidents through electronic text technologies. SINERGY shall incorporate electronic text mechanisms to keep students, staff, and faculty informed about what is going on campus.

Capability Statement: SINERGY shall provide electronic text reports about campus events to all constituents through one or more of the following mechanisms:

- Webpage updates
- Digital signage
- Desktop alerts software application
- Alerts software applications for mobile devices
- Text messages
- Instant messages
- Twitter updates

- Facebook updates
- Speech-to-Text tools

4.4.9. Direct Communication with Emergency Response Personnel Capabilities

Communication between SINERGY command center and emergency response personnel is critical. It should be conducted in a direct manner to ensure efficient response efforts. In addition, communication through diverse media ensures that there is always a link between the command center and personnel on the ground.

Capability Statement: SINERGY shall provide direct communication capabilities between incident command and emergency response personnel through one or more of the following means:

- Cellular phones: video and voice
- Satellite-based phones
- VoIP applications: video and voice
- Fax
- Closed instant messages (IM) sessions
- Messages: SMS and MMS
- E-mail
- Two-way radios (walkie-talkies or handheld transceivers)
- Twitter updates

4.4.10. Response Coordination Capabilities

Coordination is critical to effective and efficient ERM. First responders and decision makers need the capability to make, change, and revise decisions based on new information in an efficient manner. Voice and videoconference tools are important for providing this capability.

Capability Statement: SINERGY shall provide the capability to create, manage, and moderate voice conferencing sessions where users participate using cell phones, landline phones, or VoIP applications.

Capability Statement: SINERGY shall provide the capability to create, manage, and moderate video conferencing sessions where users participate using camera-equipped devices connected to the Internet.

4.4.11. Documentation Capabilities for Decisions, Expenses, and Damages

Emergencies and other unanticipated incidents always bring new insights into how the response to them can be more effective. Therefore, learning from such insights and proper updates of documentation of emergency response is critical.

In addition, emergency response managers must keep track of the cost incurred during an incident for accountability, assessment, and insurance claims. For instance, as emergency personnel inspect damages to people, property, and equipment, SINERGY should provide capabilities to store recorded notes, written notes, photos, and videos of the damages.

> **Capability Statement:** SINERGY shall provide the capability to document all operations and transactions that it automatically executes, as well as those operations and transactions the users execute.

> **Capability Statement:** SINERGY shall provide the capability of managing mobile services that integrate with portable mobile devices (e.g., phones, cameras, netbooks, tablets) to record video, audio, and text information about actions, decisions, and expenses related to emergency response.

4.4.12. Training Capabilities for Personnel and Public

SC and ERM involve everyone on campus. The ability to train personnel and engage the public is critical. The system should provide access to information about emergency plans, SA resources, security help, and online tutorials. In addition, the system should provide tools to train new personnel about the capabilities and deployed services, and run interactive simulations based on past documented cases for training purposes.

Law enforcement, health care providers, building managers, and others face the challenge of ensuring that each member of their personnel works in tandem with others. This becomes a challenge in the case of new personnel who need to be trained on how to use the capabilities of the system while interoperating with others. The system should facilitate new personnel by going through interactive simulations where they receive proper feedback on their decisions and actions. In addition, administrators expressed the need to be able to inform their staff and the students about the SC measures put in place and the security management plan. Having an online help and training capability that is accessible to them will enable self-paced learning.

Capability Statement: SINERGY shall provide training resources as an online capability to train new personal and to educate students, faculty, and staff about emergency management and SC on campus.

4.5. QUALITY NEEDS: QUALITY ATTRIBUTES

Quality attributes of a system are those characteristics that affect the quality of the entire system. They are also referred to as quality characteristics and non-functional requirements. Examples include security, interoperability, and performance.

Table 4-5 lists the quality needs that SINERGY should satisfy, as expressed by stakeholders, in order to support their business goals. These needs are expressed in the language of the problem domain. From this list, we extract key quality attributes for GDSS that support ERM in open campus environments.

Table 4-5. List of stakeholders' quality needs

Quality Needs Specification
Balance between information control and response flexibility
Interoperability among personnel and among system components
Real-time availability of information
Infrastructure availability during incidents
Fault tolerance and graceful degradation of system performance during incidents
Simplicity of solution to cater to users with limited technical skills

4.5.1. Balance between Information Control and Response Flexibility

Control over the dissemination of information is critical during emergencies. Decision makers have to walk a fine line between keeping the public informed and avoiding misinformation, secondary incidents, and misuse of information by perpetrators. While lack of information causes problems, too much information can also result in information overload where important messages can be ignored. Meaningful judgments of the tradeoff are crucial. Law enforcement and public relations stakeholders need the capability to control the flow of official information on all SINERGY outlets. Given that new and unanticipated incidents may occur, the flexibility of an emergency response plan is critical. The ability to keep campus constituents informed is important to emergency managers. Providing multiple modes of

communication helps in this regard but creates a challenge for controlling which and when information is shared. The GDSS can provide a multi-participant ranking mechanism so that depending on the importance and relevance of the information, it can be shared or not, and on one or multiple communication modes with proper levels of redundancy, speed, and reach.

Security: SINERGY shall allow authorized users to control and manage the dissemination of official information to campus constituents through university official channels (e.g., website, alerts, digital signs, e-mails, twitter updates, and Facebook updates.)

Response Flexibility: SINERGY shall provide the capability to disseminate official information to all campus constituents through one or all communication outlets based on the situation to achieve maximum reach and flexibility of the response.

4.5.2. Interoperability among Personnel and among System Components

SC and emergency preparedness and response involve many entities within an open campus. The terminology/jargon used by each of these entities is often different. Bringing all these groups to work together raises a communication challenge among individuals from different disciplines such as police officers, medical doctors, administrators, and firefighters. Unifying the communication terminology among these entities becomes a critical aspect of the success of any collaborative emergency response effort. For example, the conciseness of symbolic language and the lack of ambiguity relative to natural language can reduce ambiguity and equivocality in communication. For example, a fire siren is a clear and unambiguous message resulting in an unequivocal action of exiting the building. Similarly, the "check out" button on online stores such as Amazon.com is unambiguous symbolic message from the user to the system indicating that the user wants to act unequivocally by checking out and paying for the purchased items. While symbolic languages enjoy efficiency in communication due to avoiding ambiguity and equivocality, the potential range of incidents should also allow a flexible communication structure. A mix of symbolic language and natural language may be a reasonable tradeoff. In addition, the need to interface with other systems requires adherence to standard architectures to facilitate interoperability.

Interoperability: SINERGY shall be interoperable on three levels.

1. *Terminology:* SINERGY shall be compliant with the ICS, which advocates a common terminology defining organizational functions, facilities, resource description, and position titles.
2. *Interfaces:* SINERGY shall use Web Services standards to provide interoperable interfaces among system components, and wrappers for legacy systems.

3. *Data:* SINERGY architecture shall be based on SOA, which prescribes an ESB layer that provides seamless transformation of data shared among services.

Compliance with Standards: SINERGY shall comply with the required technology, legal, and institutional standards.

4.5.3. Real-time Availability of Information

The need to access information in real-time during an incident is important, especially for campus constituents [Kim *et al.* 2007]. Sharing a COP with the public has advantages and disadvantages. Emergency managers believe that "*while sharing of information as it becomes available is desirable, verifying it may take time.*" Misinformation can lead to panic or secondary incidents—those resulted as a reaction to a primary incident. In addition, too much information can result in information overload where individuals can potentially disregard critical information when they reach their information processing capacity.

Information Availability: SINERGY shall provide real-time access to information in a pervasive manner through two complimentary mechanisms: 1) push mechanisms such as brief SMS alerts pushed to campus constituents wherever they are, and 2) pull mechanisms such as website updates where the information is updated regularly but constituents need to go to the website and read (pull) it.

4.5.4. Infrastructure Availability during Incidents

One of the stakeholders stated that "*even the best infrastructure in the world is useless if it is not available when needed.*" An emergency could be that the infrastructure itself failed. Therefore, many stakeholders expressed a need that the emergency infrastructure should be independent from normal operation infrastructure. For instance, an alert system should use servers that are dedicated for this service and does not share resources with other university operations such as e-mails and course management systems. There should also be enough redundancy and decentralized and networked structure so that the breakdown of one node will not bring down the system.

Infrastructure Availability: SINERGY infrastructure shall be available 24 hours a day, 7 days a week. This quality attribute shall be achieved in the following manner:

- SINERGY shall use dedicated physical resources hosted on-site.
- SINERGY shall use redundant physical resources hosted in national and international off-site locations to ensure the availability of capabilities when campus infrastructure is compromised.

- SINERGY command center critical facilities shall be duplicated on a mobile vehicle that can be moved closer or away from an emergency scene as needed.

4.5.5. Fault Tolerance and Graceful Degradation of Performance during Incidents

Many stakeholders expressed their concerns about the dependencies among deployed technology solutions. When one part of the solution is affected, the entire system goes down. Therefore, the separation of concerns is a key driver for identifying SINERGY capabilities. If e-mail communication is affected due to hardware failure, texting and phone communication should not be affected.

Fault Tolerance: SINERGY shall be fault tolerant and resilient to failure through the adoption of the architecture and design concepts that promote minimal dependencies among system hardware components and among system software components, graceful degradation of performance during failures, and robust response to errors, attacks, failures, and exceptions.

4.5.6. Simplicity of Solution to Cater to Users with Limited Technical Skills

Complex technology solutions do not warrant optimal results. As an example, one director in the office of campus life suggested that "*the use of a single hallway phone in each dorm floor instead of room phones seems to be a better solution*". The reason is that this solution increases the chance of at least one student will hear the hallway phone and inform everyone in the hall in the case of an emergency. He added that "*a 40-year old model of communication can be more effective in this case*." Hence, technology should be a vehicle that facilitates effective human response to incidents so that nobody is left out due to technological complexity or various disabilities. Following this principle, SINERGY shall use simple yet useful solutions that take into consideration the variety of campus constituents including those with disabilities and limited technical skills.

Usability: SINERGY shall use usability and human factors best practices of simplicity to achieve the needs of the stakeholders by adopting popular technologies among students as well as traditional technologies used by faculty, staff, and emergency personnel. In addition, user-facing system interfaces shall make use of simple and efficient design choices to reduce the effect on system performance and scalability especially during emergencies.

4.5.7. Expected Quality Attributes

Our study focused on the quality attributes that are important to stakeholders. These constitute the attributes that should drive the architecting process. However, as a network-centric system of systems,

SINERGY should exhibit a more cohesive set of quality attributes to understand the tradeoffs among them. While implementation of SINERGY will vary based on its operational context, the list below provides key common quality attributes that complement the specific quality attributes presented in the previous section.

Reliability is the extent to which SINERGY provides its capabilities without failure under normal conditions and during emergencies within the specified performance parameters.

Maintainability is the extent to which SINERGY facilitates changes to its components. Four types of maintenance exist [Balci and Ormsby 2008]. Adaptive maintainability is concerned with adaptations required as SINERGY external environment evolves. Corrective maintainability is concerned with fixing bugs and making corrections to SINERGY components. Perfective maintainability is concerned with enhancements requested because of changing stakeholder requirements. Preventive maintainability is concerned with preventing potential problems for reengineering.

Performance is the extent to which SINERGY executes its tasks in an efficient manner to support the decision-making process of the user.

Scalability is the extent to which SINERGY maintains its functional correctness as its workload is increased within some pre-defined limits.

Survivability is the extent to which SINERGY satisfies and continues to satisfy specified critical requirements (e.g. security, reliability, real-time responsiveness, and accuracy) under adverse conditions.

4.6. DISCUSSION OF LESSONS LEARNED

The lessons learned from this study go beyond the elicitation of functional and quality needs of SINERGY stakeholders. They provide both practical and theoretical insights into the next generation of GDSS that provide real-time decision support for scenarios requiring context-awareness. In this section, we present these insights in a way that triggers future research questions.

4.6.1. SINERGY Intended Uses

The main practical contribution of this capabilities specification is intended to provide a framework that guides the architecting, design, and implementation of a real-time context-aware GDSS for campus ERM. The list of capabilities provided constitutes a high-level domain analysis from various stakeholder perspectives. Based on this analysis, we classified the IUs for SINERGY capabilities specification into

two categories: 1) IUs related to the development of SINERGY architecture specification, and 2) IUs related to the evaluation of SINERGY implementations when operational. This categorization is important because it bounds how these results can be used. Table 4-6 lists SINERGY intended uses.

Table 4-6. IUs of SINERGY

IUs related to the development of SINERGY architecture specification
Establish interoperability and integration guidelines for SINERGY component systems
Multiple channels of communication between SINERGY stakeholders
Shared understanding of responsibilities related to campus SA, SC, and ERM
Support decision makers devise an acquisition strategy of SINERGY capabilities
Contractual reference between decision makers and providers of SINERGY component systems
IUs related to the evaluation of SINERGY implementations when operational
Evaluation of the architecture specification against its required quality attributes
Evaluation of the soundness of architecture decisions and their impact on development
Evaluation of the adherence of SINERGY operational models to established campus emergency, privacy, and legal guidelines
Gap analysis between existing and planned SINERGY capabilities

4.6.2. Emergency and Decision Making

During emergencies, the outcome of decision-making processes may have a direct impact on life. Any support that can be provided to decision makers to make the most optimal decisions is considered desirable. We conducted interviews to propose a GDSS architecture that can enhance campus safety without diminishing the value of open campus environment. However, key challenges remain to be addressed to ensure continuous improvements to ensure the right information to the right people at the right time for optimized coordination and decision making.

Technology: Typically, organizational structures and business processes influence the way GDSS is designed and used. However, given the ubiquity of the network and the pervasiveness of information-producing and information-consuming devices, the influence should be managed properly to the positive [Zhang and Goddard 2007]. Information technology has a tremendous influence on organizational

structures and business processes now, thereby influencing traditional decision-making processes. Novel approaches to design GDSS for providing a COP and helping decision makers make better decisions and auditing how decisions are converted into action in response to incidents is critical. SINERGY architecture is derived from the interviews that describe the behavior of the users. Given that SINEGY architecture will influence behavior, users' behavior is likely to change requiring new structural changes to SINERGY to address new emerging behaviors. The duality between structure and behavior, and how they influence each other, raises a fundamental question about how decision processes are designed within a useful architecture [Bose and Chen 2009]. More specifically, how to design decision processes into a technology architecture such as a GDSS to facilitate coordinated decision making and assuring safety while preserving an open campus environment with its ever changing student behavior? The ease of updating the GDSS structures and decision making support mechanisms is essential to facilitate accommodating the new behavior on campus. How can the GDSS support distributed decision making using the new technologies that enable us to distribute software over a network providing both autonomy and control to nodes in the network? Next generation GDSS may be viewed as a loosely coupled system of nodes acting as semi-autonomous agents in a coordinated fashion that can learn using a case-based approach. As new technologies become mainstream, their influence on decision-making processes become undeniable warranting their incorporation into new decision-making processes. SINERGY used by campus constituents can influence new decision-making paradigm aiding timely decisions by campus administrators and emergency responders to act and make decisions for the right context.

Augmented Decision Making: Organizations follow established decision-making processes to ensure optimal outcomes. More than any time before, decision-making processes rely heavily on information systems to augment their impact [Gundersen 1995]. Augmented decision making is critical, especially during emergencies where information is continuously changing, and may be incomplete, incorrect, or inaccessible. One of the key findings of this study is the importance of distribution of assets to ensure availability and redundant coverage. Availability of assets across various networks allows decision makers to make use of the most current information to form their decisions.

Collaborative Decision Making: Decision making involves consultation with others to ensure consistency of decisions and overall alignment with established processes. In cases of emergencies, collocation of decision makers is not always possible or desirable. Therefore, support for distributed decision making is crucial. Video and audio conferencing technologies provide necessary tools to connect geographically dispersed decision makers to allow collaborative efforts to tackle complex emergency problems and make the best decisions in real-time for the right context.

Case-based Learning: The challenging fact about emergency response is that emergencies evolve from different sets of conditions. It is almost impossible to prepare for a particular instant of an emergency given that there may not be a priori information about how to respond to every incident. Therefore, emergency preparedness focuses on procedures that address types of emergencies rather than instances that may have happened in the past. Case-based learning is a concept that came up in several discussions with stakeholders. It is concerned with the issue of using cases from past incidents to develop self-awareness (i.e., artificial intelligence) in GDSS. Hence, GDSS can help implement case-based reasoning to propose a base solution that may be modified and tailored by human decision makers for new incidents.

When we talk about a system as complex as SINERGY, several challenges surface to the discussion. The study that we conducted allowed us to learn about several issues related to decision-making processes, emergency management, collaboration, and coordination. These lessons, we believe, provide rich topics for further research. We provided a framework for the design of a GDSS for emergency management response for open campus environments and this basic framework should help answer the initial questions in this inquiry.

4.7. CONCLUDING REMARKS

The capabilities and quality attributes identified herein are intended to provide a framework to guide the architecting, design, and implementation of SINERGY. They are a starting point from which other SINERGY capabilities and quality attributes are identified based on the operational context of a particular implementation of this GDSS.

The outcome of this study shows a clear contrast between what is currently practiced and deployed, and what is actually needed to fully prepare open campuses to man-made disasters. It provides a supporting case to our argument that current technology solutions address the problem from one or few perspectives, when it should be addressed comprehensively. The nature of the problem is multi-faceted and so should be its solution. The next step is to build on these results to develop an architecture that takes into consideration the operational context of a solution implementation.

Lastly, this study also provides an illustration for the use of capabilities as an alternative approach to determining functionality requirements for guiding the design of complex and useable GDSS. This approach balances between baselining key requirements in the form of capabilities early on in the development life cycle, and the accommodation of requirements volatility inherent in GDSS developments. Therefore, this balanced approach enables GDSS developers to manage system complexity

by ensuring that tradeoffs are reasoned about early on before details are addressed as the development moves on to design and implementation.

CHAPTER 5: SINERGY ARCHITECTURE SPECIFICATION

SINERGY architecture specification consists of 31 DoDAF-described models. These models are included in this chapter for the cohesiveness of the dissertation. However, to better navigate through the architecture models, we have created a dedicated website that makes it easy to view these models. The website URL is http://manta.cs.vt.edu/sinergy.

5.1. AV-1: OVERVIEW AND SUMMARY INFORMATION

This model describes the AV-1 DoDAF-described model for SINERGY architecture. The AV-1 model serves as a point of reference for the entire architecture specification. It provides a high-level summary of architecture information pertinent to all viewpoints.

This model consists of the following main sections:

- Architecture Metadata: Lists the name of the architecture, project information, developers of the architecture and the tools they used, and the DoDAF-described models that comprise the architecture specification.
- Purpose of the architecture: Describes the main purpose of creating the architecture. It also describes other objectives for the use of the architecture.
- Context of the architecture: Describes the concept of operation, the scenarios, and business needs the architecture addresses.

The AV-1 should be used as a guide to understand the high-level objectives of SINERGY architecture. The list of DoDAF-described models listed herein provides detailed documentation of the architecture.

5.1.1. Architecture Overview

This architecture is designed to support the SINERGY key goal of providing SA, SC, and ERM capabilities in open campus environments.

5.1.1.1. Architecture Identification

Name : SINERGY Architecture

Version : version 1.0

Completion status : under development

Completion date : 04/01/2010

5.1.1.2. Project Identification

Project start date : 09/01/2009

Project end date : 04/01/2011

Estimated effort : 1.5 man-years

5.1.1.3. Points of Contact

Lead Architect : Amine Chigani, achigani@vt.edu

Supervisor : Osman Balci, balci@vt.edu

5.1.1.4. Tools and File Formats

Tools used : Microsoft Office 2010, IBM Rational System Architect v.11.3
Microsoft Visio 2010, Micromedia Dreamweaver 8, Adobe Professional 9

File formats used : docx, vsd, mdb, html, png, jpeg, pdf, emf

5.1.1.5. Architecture Models

SINERGY architecture specification consists of 31 DoDAF-described models namely:

1. AV-1 Architecture Overview and Summary Information
2. AV-2 Integrated Dictionary
3. CV-1 Vision for the Architecture Capabilities
4. CV-2 Capability Taxonomy
5. CV-4 Capability Dependencies
6. CV-6 Capability to Operational Activities Mapping
7. CV-7 Capability to Services Mapping
8. OV-1 ERM Operational Model
9. OV-1 SA and SC Operational Model
10. OV-2 ERM Operational Resource Flow Description
11. OV-2 SA and SC Operational Resource Flow Description
12. OV-4a Organizational Relationships Chart
13. OV-4b Role-based Organizational Relationships Chart
14. OV-5a ERM Operational Activity Decomposition Tree
15. OV-5a SA and SC Operational Activity Decomposition Tree

16. OV-5b ERM Operational Activity Model
17. OV-5b SA and SC Operational Activity Model
18. StdV-1 Standards Profile
19. SV-1a System Context Model
20. SV-1b Systems Interface Description
21. SV-2 Systems Resource Flow Description
22. SV-3 Systems-Systems Matrix
23. SV-4 Systems Functionality Taxonomy
24. SV-5b Operational Activity to Systems Traceability Matrix
25. SvcV-1a SOA Conceptual Layers
26. SvcV-1b SINERGY SOA-based Architecture
27. SvcV-1c Services Context Description
28. SvcV-2 Services Resource Flow Description
29. SvcV-3a Systems-Services Matrix
30. SvcV-4 Services Categorization
31. SvcV-5 Operational Activity to Services Traceability Matrix

5.1.2. Purpose

SINERGY architecture specification was developed primarily to guide decision makers on university campuses make acquisition decisions and develop strategies regarding the provision and maintenance of safety within their campuses. The architecture provides a university-wide solution to balancing openness and safety within campus environments through capabilities that enable SA, SC, and ERM. It takes into consideration state and federal standards and is compliant with the ICS.

In addition, SINERGY architecture specification was developed to understand the true requirements of the issues relating to campus safety and emergency management. It provides a blueprint to guide the design, development, and deployment of systems that can be used across open campuses to provide SA, SC, and ERM capabilities.

Finally, a successful demonstration of the effectiveness of SINERGY architecture through architecture evaluation leads to implementations of software-based, network-centric systems of systems enabling institutions to be prepared to respond to man-made tragedies on their campuses. The impact can reach millions of students, faculty, and staff in thousands of colleges, universities, and other open campus environments.

5.1.3. Intended Uses of the Architecture Specification

Table 5-1 lists the IUs of SINERGY architecture specification models. We categorize these IUs by stakeholder categories.

Table 5-1. IUs of SINERGY architecture specification models

Developers	
1.	Establishing interoperability and integration guidelines for SINERGY component systems
2.	Evaluation of how the architecture specification enables SINERGY to exhibit its required quality characteristics
3.	Evaluation of the soundness of architecture decisions and their impact on SINERGY development
4.	Input to the Design Process of SINERGY life cycle
5.	Guide to the deployment, maintenance, and evolution of SINERGY
6.	Reusable abstraction for developing systems that exhibit similar functional and quality characteristics to those of SINERGY
Managers and Decision Makers	
7.	Means of communication among all SINERGY stakeholders
8.	Evaluation of the adherence of SINERGY operational models to established campus emergency, privacy, and legal guidelines
9.	Training personnel with responsibilities related to campus SA, SC, and ERM
10.	Gap analysis between existing and planned SINERGY capabilities
11.	Supporting decision makers devise an acquisition strategy of SINERGY capabilities
12.	Identification of performers, organizational units, and locations related SINERGY operations
13.	Contractual reference between decision makers and potential providers of SINERGY component systems

5.1.4. Context

The history of university, college, and high school campuses is eventful with man-made tragedies ensuing a tremendous loss of life and property. Virginia Tech's April 16 shooting [VT 2007] ignited the discussion about balancing openness and safety in open campus environments. Decision makers face the challenge to ensure that their campuses provide an open learning environment and at the same time safe. Existing technology solutions are characterized by addressing bits and pieces of the problem (e.g. ENS). Without a comprehensive, innovative understanding of the requirements for an institution-wide solution

that enables effective SC and efficient emergency response, the proposed solutions fall short from the desired outcomes.

5.1.4.1. The Campus Environment

The first step toward an effective solution is to understand the operating environment. The scope of SINERGY architecture is confined to an open campus environment, where access to campus facilities is generally open to faculty, staff, students, and campus visitors. Shared campus facilities include the gym, athletic facilities, library, hallways, cafeterias, administrative office buildings, and residential halls. Figure 5-1 depicts an overview of the facilities in an open campus environment.

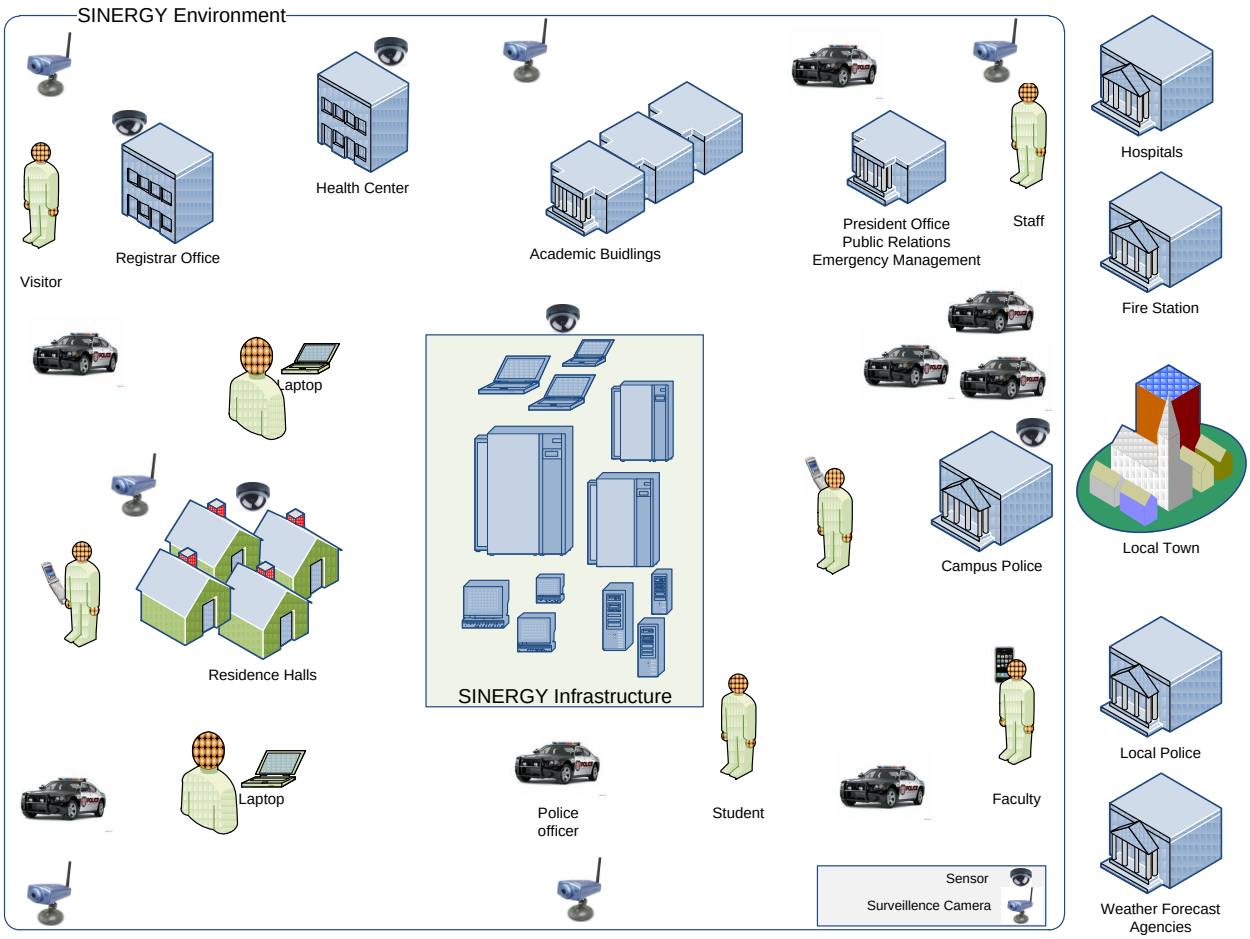


Figure 5-1. SINERGY operating context

5.1.4.2. Problem Components

SINERGY architecture provides a holistic view that addresses the problem by decomposing it into three major areas of interests: 1) SA, 2) SC, and 3) ERM.

In other words, a safe and open campus environment can be realized through a system that enables the creation of a COP of the campus environment shared by all campus entities (i.e., SA). Having a common picture of what goes on campus at any point in time is key to enabling effective SC measures to be put in place (i.e., SC). Finally, common SA and effective SC lay the foundation for an efficient and effective ERM in the case of an emergency.

5.1.4.3. Guidance References

The SINERGY architecture describes a solution that will exist within a larger set of business rules that guide the day-to-day operation of a university campus. Educational institutions are required to comply with state and federal regulations in the preparation for emergencies and establishment of contingency plans for responding to threats that could harm personnel, faculty, students, or property.

The development of SINERGY architecture is guided by the following references:

- ICS: A mandatory emergency management framework that must be used to response to all emergencies [FEMA 2010a].
- SOA: Service-Oriented Architecture
- Web Services standards and specifications
- MSAAM: Military System Architecture Assessment Methodology [Balci and Ormsby 2008]
- Process of Architecting for Software/System Engineering
- Department of Defense Architecture Framework Version 2.0

5.1.5. Releaseability

SINERGY is developed as part of a research conducted in the Computer Science Department at Virginia Tech by Amine Chigani under the supervision of Osman Balci. The developers of this architecture permit the unlimited reuse and distribution of part or all of the architecture for non-commercial purposes, provided that clear attribution to the original authors is specified.

5.2. AV-2: INTEGRATED DICTIONARY

This model presents the AV-2 DoDAF-described model for SINERGY architecture specification. It describes the integrated dictionary of the SINERGY architecture, which includes definitions of acronyms used across the architecture specification artifacts.

ALMA	Architecture Level Modifiability Analysis
AOI	Area of Interest
ATAM	Architecture Tradeoff Analysis Method
AV	All Viewpoint
BPEL	Business Process Execution Language
CBAM	Cost-Benefit Analysis Method
CFO	Chief Finance Officer
CMMI	Capability Maturity Model Integration
COP	Common Operating Picture
CORBA	Common Object Request Broker Architecture
CSA	Client-Server Architecture
CSD	Capabilities Specification Document
CV	Capability Viewpoint
DCOM	Distributed Component Object Model
DIV	Data and Information Viewpoint
DOA	Distributed Objects Architecture
DoD	United States Department of Defense
DoDAF	Department of Defense Architecture Framework
E911	Enhanced 911
ENS	Electronic Notification System
ERM	Emergency Response Management
ESB	Enterprise Service Bus
FAAM	Family-Architecture Analysis Method
FCC	Federal Communications Commission
FEMA	Federal Emergency Management Agency
FERPA	Family Educational Rights and Privacy Act
GIS	Geographic Information System
GPS	Global Positioning System
HR	Human Resources
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IASA	International Association of Software Architects
ICS	Incident Command System
IDL	Interface Definition Language
IM	Instant Messaging

IP	Internet Protocol
IPv6	Internet Protocol version 6
IT	Information Technology
ITAC	IT Architect Certification Program
JMS	Java Message Service
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging System
MoDAF	Ministry of Defense Architecture Framework
MOM	Message-Oriented Middleware
MSAAM	Military System Architecture Assessment Methodology
NAF	NATO Architecture Framework
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
ORB	Object Request Broker
OSI	Open Source Initiative
OV	Operational Viewpoint
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PSAP	Public Safety Answering/Access Point
PV	Project Viewpoint
REST	Representation State Transfer
RSS	Really Simple Syndication/Rich Site Summary/RDF Site Summary
SA	Situational Awareness
SAAM	Software Architecture Analysis Method
SAML	Security Assertion Markup Language
SC	Security Control
SEI	Software Engineering Institute
SG	Specific Goal
SINERGY	campuS sItuational awareNess and Emergency Response manaGement sYstem
SMS	Short Messaging System
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture

SOAP	Simple Object Access Protocol
SP	Specific Practice
SSL	Secure Sockets Layer
StdV	Standard Viewpoint
SV	Systems Viewpoint
SvcV	Services Viewpoint
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
TOGAF	The Open Group Architecture Framework
TTS	Text to Speech
UAS	Unmanned Areal System
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
UGS	Unmanned Ground System
VoIP	Voice over IP
VT	Virginia Tech
WS	Web Service
WSDL	Web Service Description Language
XML	eXtensible Markup Language

5.3. STDV-1: STANDARDS PROFILE

This model describes the StdV-1 DoDAF-described model for the SINERGY architecture specification. It describes the list of standards that apply to SINERGY.

5.3.1. Compliance Standards

E-911: Enhanced 911 is a telecommunication system similar to the basic 911 system required by the Federal Communications Commission (FCC). It enables mobile and cellular phones to process 911 emergency calls and route them to the closest public safety answering point (PSAP).

Website: http://www.fcc.gov/pshs/services/911-services/.

ICS: The ICS is an incident management approach developed by the Federal Emergency Management Agency (FEMA). ICS provide a standardized system that enables the interoperability of various agencies in response to any type of incidents.

Website: http://www.fema.gov/emergency/nims/IncidentCommandSystem.shtm.

FERPA: The Family Educational Rights and Privacy Act is a federal law that regulates access to students' academic records.

Website: http://www2.ed.gov/policy/gen/guid/fpco/ferpa/.

5.3.2. Technology Standards

Table 5-2. A list of Web Services specifications compiled by InfoQ [InfoQ 2007] (Used under the fair use guidelines, 2011)

Basic Profile 2.0	The Basic Profile provides implementation guidelines for how related set of non-proprietary Web Service specifications should be used together for best interoperability.
XML 1.1	Extensible Markup Language is a pared-down version of SGML, designed especially for Web documents. It allows one to create own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.
WS-Policy 1.5	Policy describes the capabilities and constraints of the policies on intermediaries and endpoints (e.g. business rules, required security tokens, supported encryption algorithms, privacy rules).
UDDI 3.0.2	Universal Description, Discovery and Integration (UDDI) defines a set of services supporting the description and discovery of businesses, organizations, and other Web Services providers, the Web Services they make available, and the technical interfaces which may be used to access those services.
WSDL 2.0 Core	Web Service Description Language 2.0 Core is an XML-based language for describing Web Services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.
BPEL 2.0	Business Process Execution Language for Web Services 2.0 (BPEL4WS) provides a language for the formal specification of business processes and business interaction protocols using Web Services.
WS-Security 1.1	WS-Security is a communications protocol providing a means for applying security to Web Services.
WS-Security: **SOAP Message Security 1.1**	WS-Security: SOAP Message Security describes enhancements to SOAP messaging to provide message integrity and confidentiality. Specifically, this specification provides support for multiple security token formats, trust domains, signature formats and encryption technologies. The token formats and semantics for using these are defined in the associated profile documents.
WS-SecurityPolicy 1.1	WS-SecurityPolicy defines how to describe policies related to various features defined in the WS-Security specification.
SOAP 1.2	SOAP is an XML-based protocol for the exchange of information in a decentralized, distributed environment.
WS-Notification	WS-Notification is a family of related white papers and specifications that define a standard Web Services approach to notification using a topic-based publish/subscribe pattern.
XML Process Definition Language	XML Process Definition Language (XPDL) provides an XML file format that can be used to interchange process models between tools.
Simple SOAP Binding Profile 1.0	Simple SOAP Binding Profile – The Simple SOAP Binding Profile consists of those Basic Profile 1.0 requirements related to the serialization of the envelope and its representation in the message.

5.4. CV-1: VISION FOR THE ARCHITECTURE CAPABILITIES

This model describes the CV-1 DoDAF-described model for SINERGY architecture specification. It describes the overall vision of SINERGY capabilities.

5.4.1. Vision

SINERGY strategic vision is to transform the way campus safety is provided. Currently solutions are characterized by the deployment and use of stovepipe solutions without much integration. Campus surveillance is operated by the police, notification systems are operated by the public relations office, and the website is managed by IT Office. Calls to 911 are routed to a single dispatch system. There is no integration of all these capabilities to create a common picture of campus at any point in time.

SINERGY architecture describes a solution that integrates existing and new technologies through the adoption of service orientation principles. It envisions a network-centric solution where everything is connected to everything else in order to create an augmented SA about the campus environment shared not only among administrators and law enforcement personnel but also among students, faculty, and staff.

5.4.2. Capabilities Supporting the Vision

Campus safety should be thought of in terms of three essential capabilities: SA, SC, and ERM. Table 5-3 summarizes the list of capabilities that we enlisted from stakeholders. The capabilities are categorized by the essential capabilities.

Table 5-3. A list of SINERGY capabilities

Security Control	
1	Diverse incident detection capabilities
2	Diverse incident reporting capabilities
3	On-demand surveillance capabilities
Situational Awareness	
4	Common operational picture capabilities
5	Geographical Information System and campus visualization capabilities
6	Audible situational awareness capabilities
7	Visual situational awareness capabilities
8	Electronic text situational awareness capabilities
Emergency Response Management	
9	Direct communication with emergency response personnel capabilities
10	Response coordination capabilities
11	Documentation capabilities for decisions, expenses, and damages
12	Training capabilities for personnel and public

5.4.3. Services Supporting the Capabilities

SINERGY provides a set of four types of services:

- Core Web Services
- Mobile app services
- Data access services
- Business process services

Core Web Services are basic Web Services that provide the core functionality. Pure Web Services do not provide a graphical user interface to the consumer. Rather, these services are consumed through service interfaces specified in WSDL. Application Web Services are services that provide functionality through a browser or a client application. They are consumed through a graphical user interface.

Mobile app services are services that provide functionality optimized to run on mobile devices through a mobile app. Consumers download an app on their mobile devices and access the functionality through the app.

Data access services are services that provide access to data of interest to consumers. These services also do not provide a graphical user interface, and they are consumed through service interfaces.

Business process services are services provided by human elements to complement the operations of the overall system.

The list of services that SINERGY provides includes:

- Business Process Services:
    1. Process fax report service
    2. Process 911 call report service
    3. Process in-person report service
    4. Process two-way radio report service
- System Security:
    5. Authenticate user service
    6. Authorize user service
    7. Validate policy service
- Incident Reporting:
    8. E-mail report service
    9. Text message report service
    10. Mobile app report service
    11. Police in-car report service
    12. Web portal report service
- Incident Detection:
    13. Register sensor service
    14. De-register sensor service
    15. Update sensor service
    16. Manage sensor service
    17. Sensor report service
    18. Register camera service
    19. De-register camera service

20. Manage camera service
21. Camera report service

- Notifications:
  22. Update classroom electronic boards service
  23. Update digital signage service
  24. Invoke siren light service
  25. Invoke fire alarm system lights service
  26. Invoke siren service
  27. Push voicemail service
  28. Update homepage service
  29. Push text message service
  30. Push instant message service
  31. Push mobile app alert service
  32. Push desktop application alert service
- Data Access:
  33. Access building schematic service
  34. Access campus digital map service
  35. Access GIS data service
  36. Access medical records service
  37. Access HR records service
  38. Access student records service
  39. Access police criminal records service
  40. Alert fire department service
  41. Alert hospital service
  42. Access sensed data store service
  43. Access surveillance data store service
- Enterprise Service Bus:
  44. Register a service
  45. Update a service
  46. Revoke a service
  47. Lockup a service
  48. Invoke a service
  49. Convert data service
  50. Route protocol service

- Common Operational Picture:
  51. Create COP service
  52. Update COP service
  53. Share COP service

5.4.4. Capability Development Timeline

The deployment of these capabilities should follow a logical order to avoid development and deployment bottlenecks. This logical order is summarized in the following acronym: DAPR (Detect, Alert, Plan, and Respond). We list each phase with it corresponding capabilities.

- Phase 1: Detect
  a. Diverse incident detection tools
  b. Diverse incident reporting tools
  c. On-demand surveillance to avoid privacy issues
- Phase 2: Alert
  a. Audible situational awareness capabilities
  b. Visual situational awareness capabilities
  c. Electronic text situational awareness capabilities
- Phase 3: Plan
  a. Common operational picture
  b. GIS and campus visualization capabilities
  c. Direct communication with emergency response personnel
- Phase 4: Respond
  a. Response coordination capabilities
  b. Documentation and bookkeeping of decisions, expenses, and damages
  c. Training and education of personnel and the public

5.5. CV-2: CAPABILITY TAXONOMY

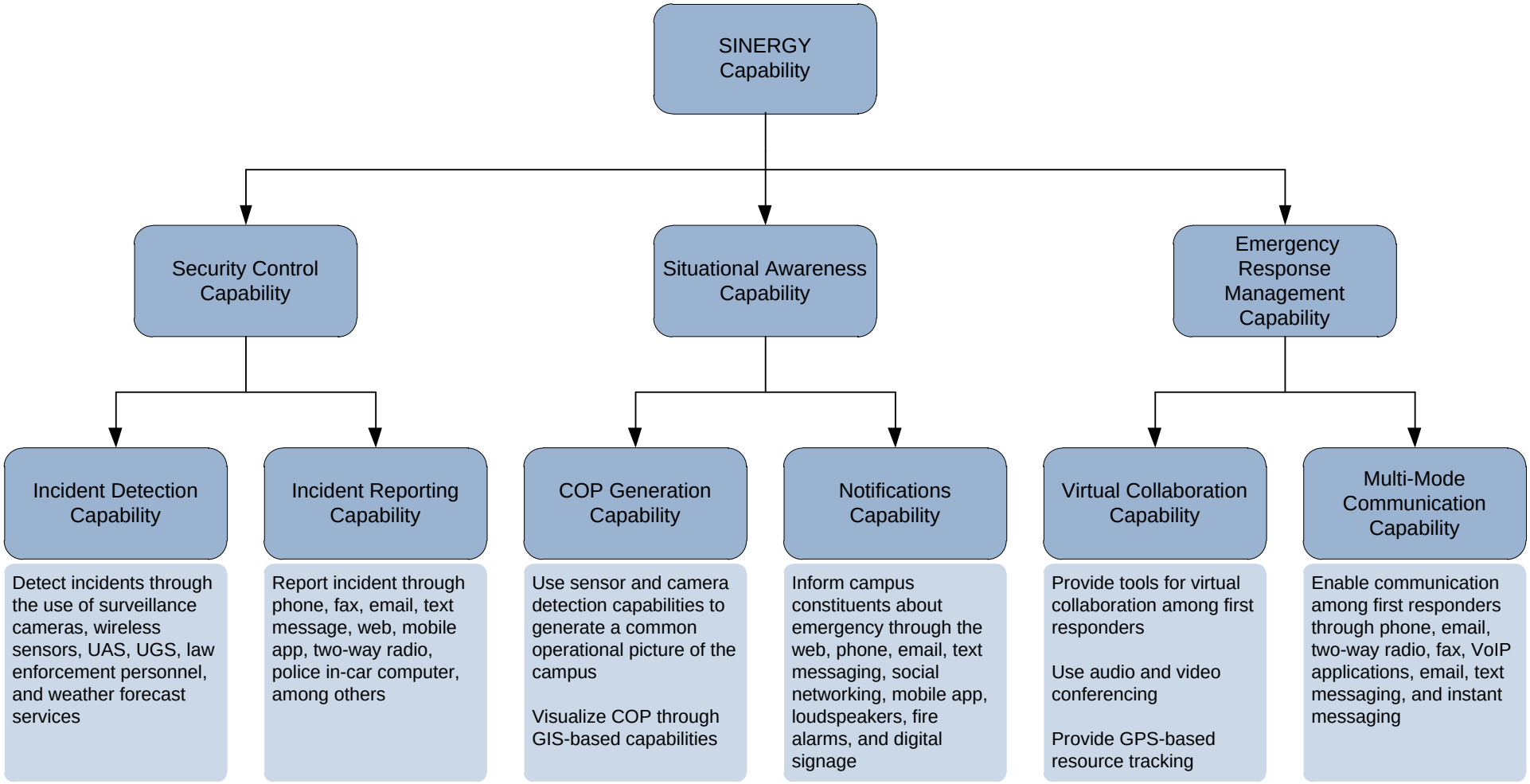


Figure 5-2. CV-2: Capability Taxonomy

5.6. CV-4: CAPABILITY DEPENDENCIES

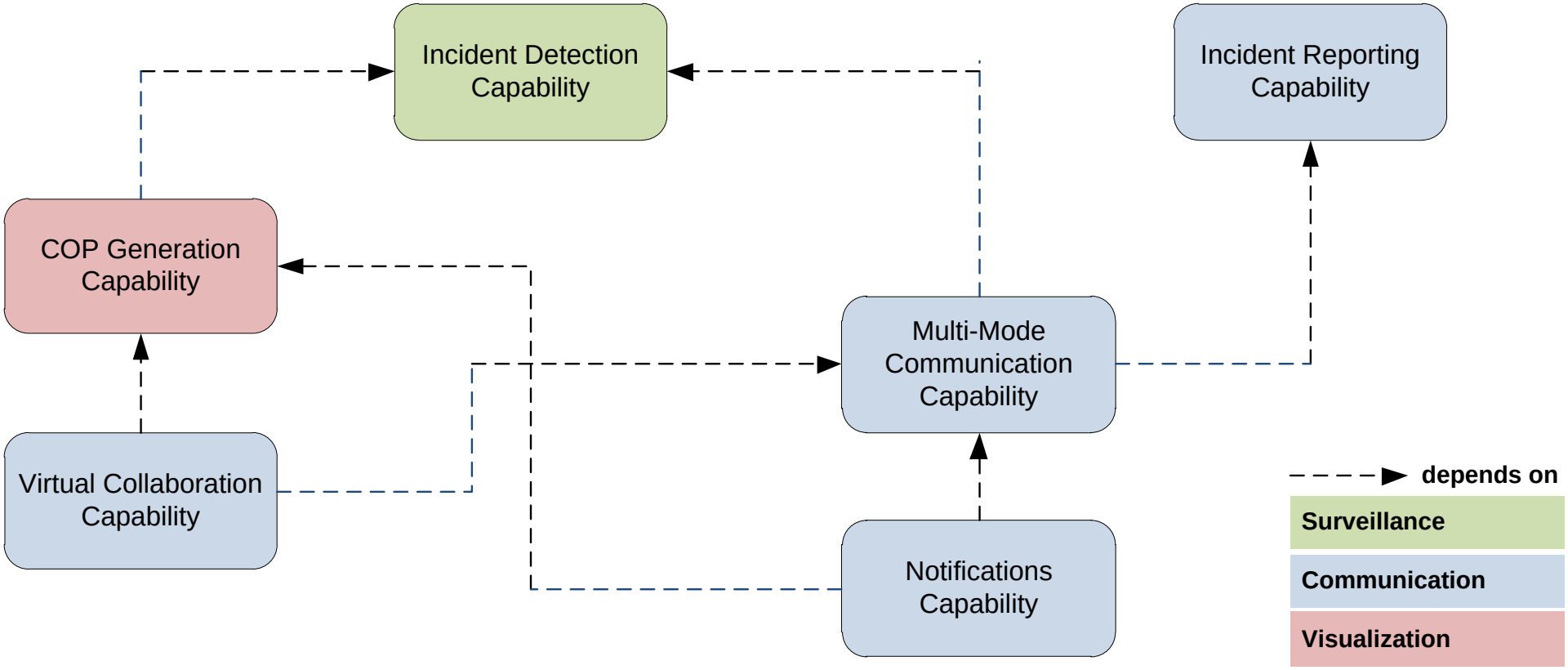


Figure 5-3. CV-4: Capability Dependencies

5.7. CV-6: CAPABILITY TO OPERATIONAL ACTIVITIES MAPPING

	Incident Detection Capability	Incidents Reporting Capability	COP Generation Capability	Notifications Capability	Virtual Collaboration Capability	Multi-Mode Communication Capability
Identify Incident	●	●				
Plan Response				●	●	
Execute Response				●	●	
Receive Incident Information	●	●				
Establish Incident Command				●	●	●
Create COP			●		●	●
Execute Incident Action Plan					●	
Detect Incident	●	●				
Receive Incident Report	●	●				
Verify Incident Information	●		●			
Inform Command Center		●				
Communicate with First Responders					●	●
Inform Constituents				●		●
Coordinate Response Efforts					●	●
Document Decisions/Expenses						●
Monitor AOIs	●		●			
Assess Situation	●		●			
Make Decision			●		●	
Manage Sensors Network	●		●			
Manage Surveillance Camera	●		●			
Manage Patrolling	●		●		●	
Assess COP			●			
Assess Incident Reports		●				
Identify Incident Type	●	●				
Initiate ERM Plan				●		●

Figure 5-4. CV-6: Capability to operational activities mapping

5.8. CV-7: CAPABILITY TO SERVICES MAPPING

	Incident Detection Capability	Incidents Reporting Capability	COP Generation Capability	Notifications Capability	Virtual Collaboration Capability	Multi-Mode Communication Capability	SOA Infrastructure Capability
Send recorded data service	●						
Send sensed data service	●						
Area surveillance service	●						
Area sensing service	●						
Networking service						●	
Process sensor data service			●				
Process camera data service			●				
Process UAS data service			●				
Proces UGS data service			●				
Generate COP service			●				
Manage COP service				●			
Share COP service							
Login service	●	●	●	●	●	●	
System control service	●	●	●	●	●	●	
Authenticate use service	●	●	●	●	●	●	
Authorize user service	●	●	●	●	●	●	
Validate policy service	●	●	●	●	●	●	
Send email service		●		●		●	
Send voicemail service		●		●		●	
Send SMS/MMS service		●		●		●	
Send IM service		●		●		●	
Activate alarm service		●					
Push app alert service				●			
Push desktop application alert service				●			
Update homepage service				●			
Update digital signage service				●			
Update social networks status service				●			
Register WS service							●
Update WS service							●
Lookup WS service							●
Revoke WS service							●
Invoke WS service							●
Convert data service							●
Route protocol service							●
Command service							●
COP display service			●	●			
Coordination service					●		
Communication service						●	
Resource tracking service					●		
Report in-person service		●					
Report via phone service		●					
Send email report service		●					
Send SMS/MMS report service		●					
Send app report service		●					
Send web portal report service		●					
Activate fire alarm service		●					
Process radio call service			●				
Process fax report service			●				
Process 911 report service			●				
Process in-person report person			●				
Decision-making support service							
Manage first responders service							
Document decisions/expenses service							
Report via radio service		●					
In-car computer report service		●					
Provide area surveillance service	●						
Faciliate communication with command center service						●	

Figure 5-5. CV-7: Capability to services mapping

5.9. OV-1A: ERM OPERATIONAL MODEL

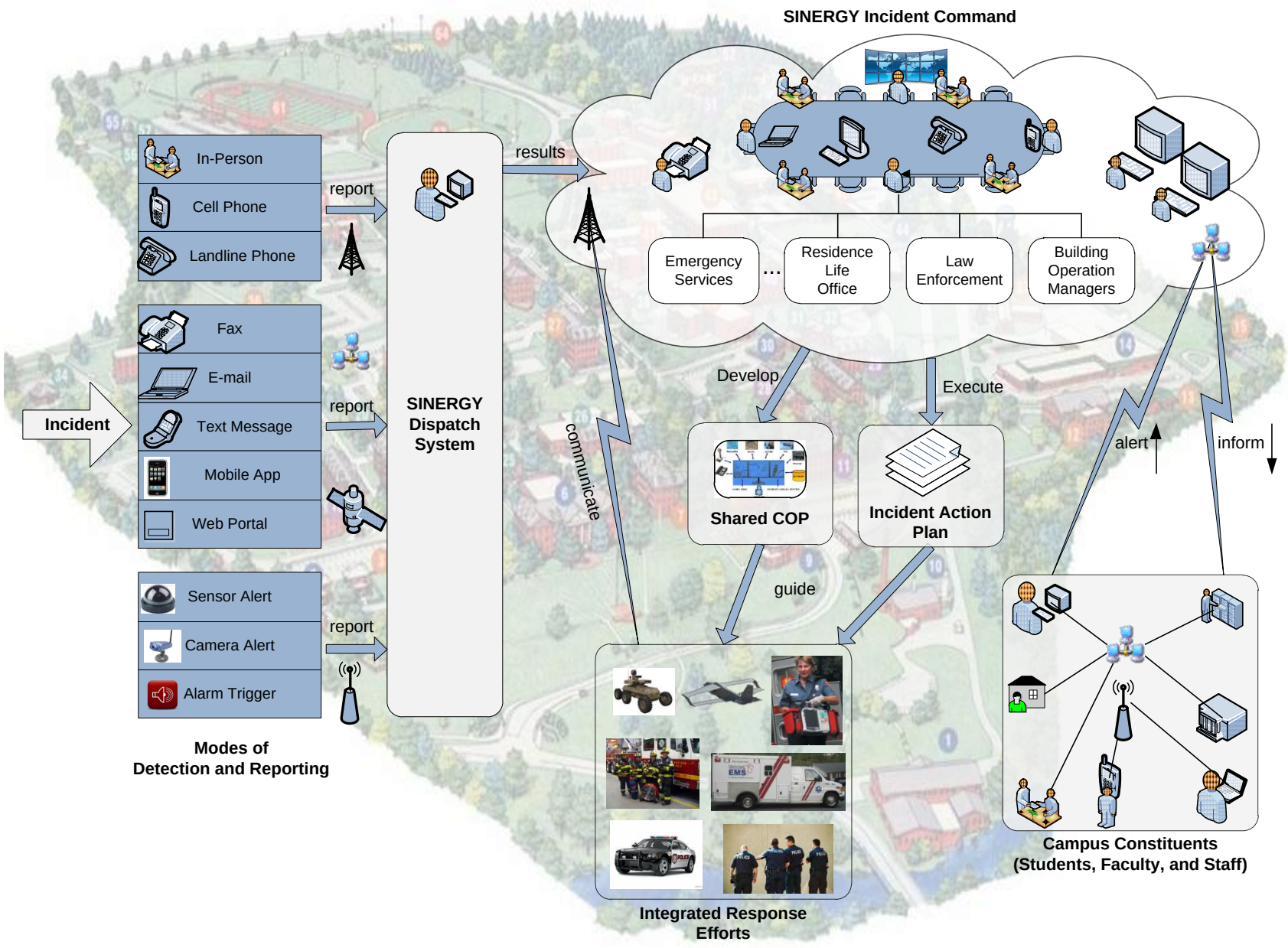


Figure 5-6. OV-1a: ERM operational model

5.10. OV-1B: SA AND SC OPERATIONAL MODEL

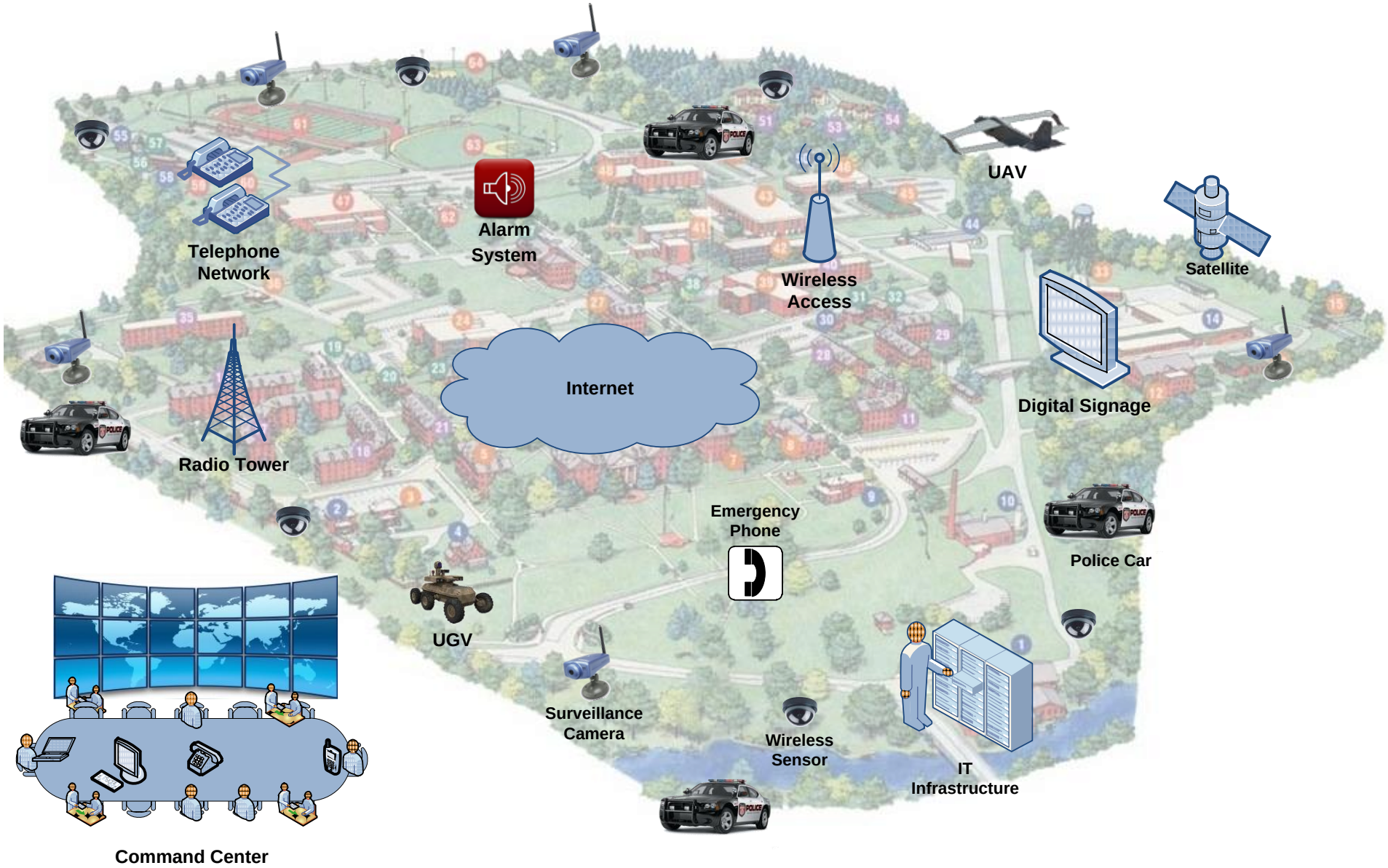


Figure 5-7. OV-1b: SA and SC operational mode

5.11. OV-2A: ERM OPERATIONAL RESOURCE FLOW DESCRIPTION

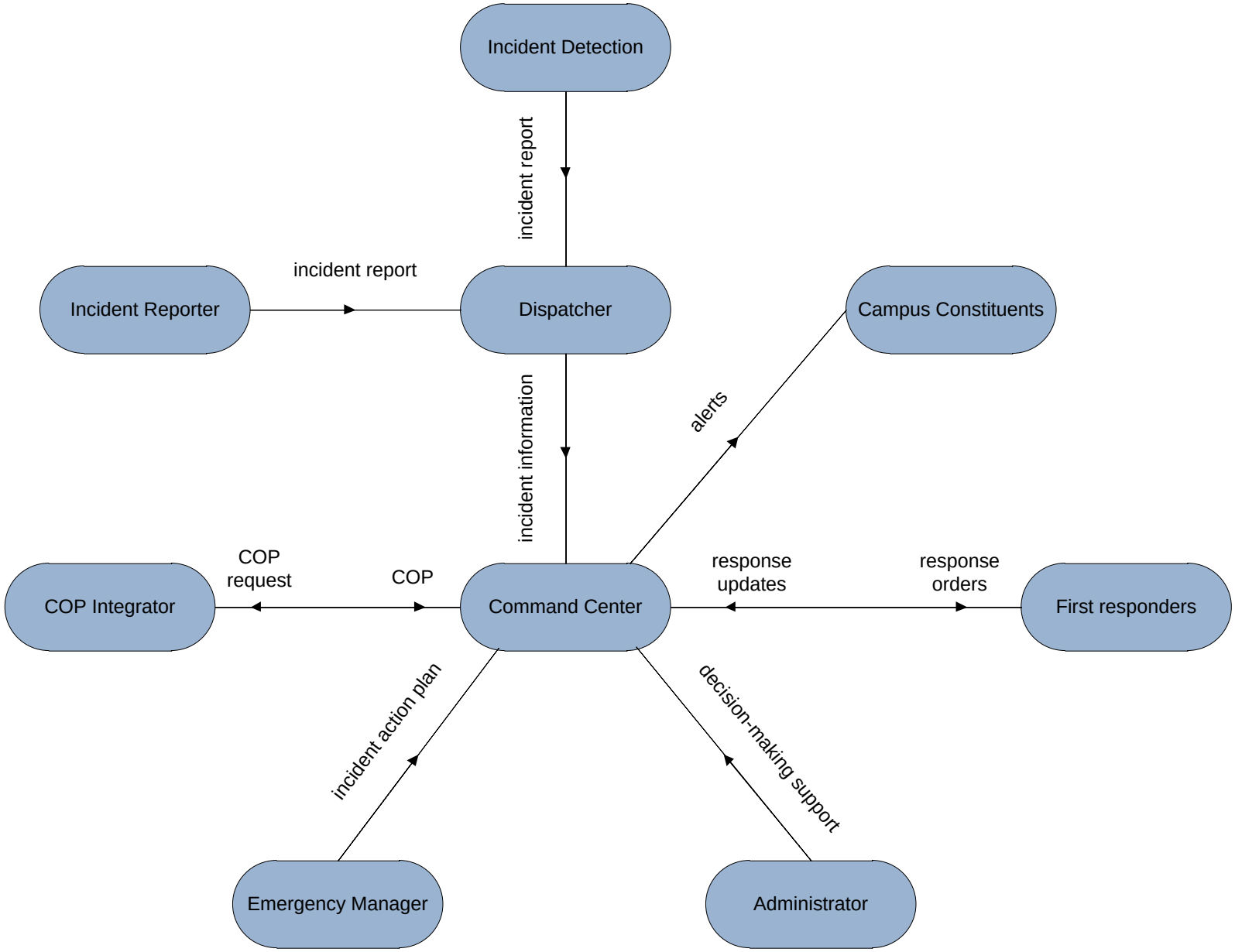


Figure 5-8. OV-2a: ERM operational resource flow description

5.12. OV-2B: SA AND SC OPERATIONAL RESOURCE FLOW DESCRIPTION

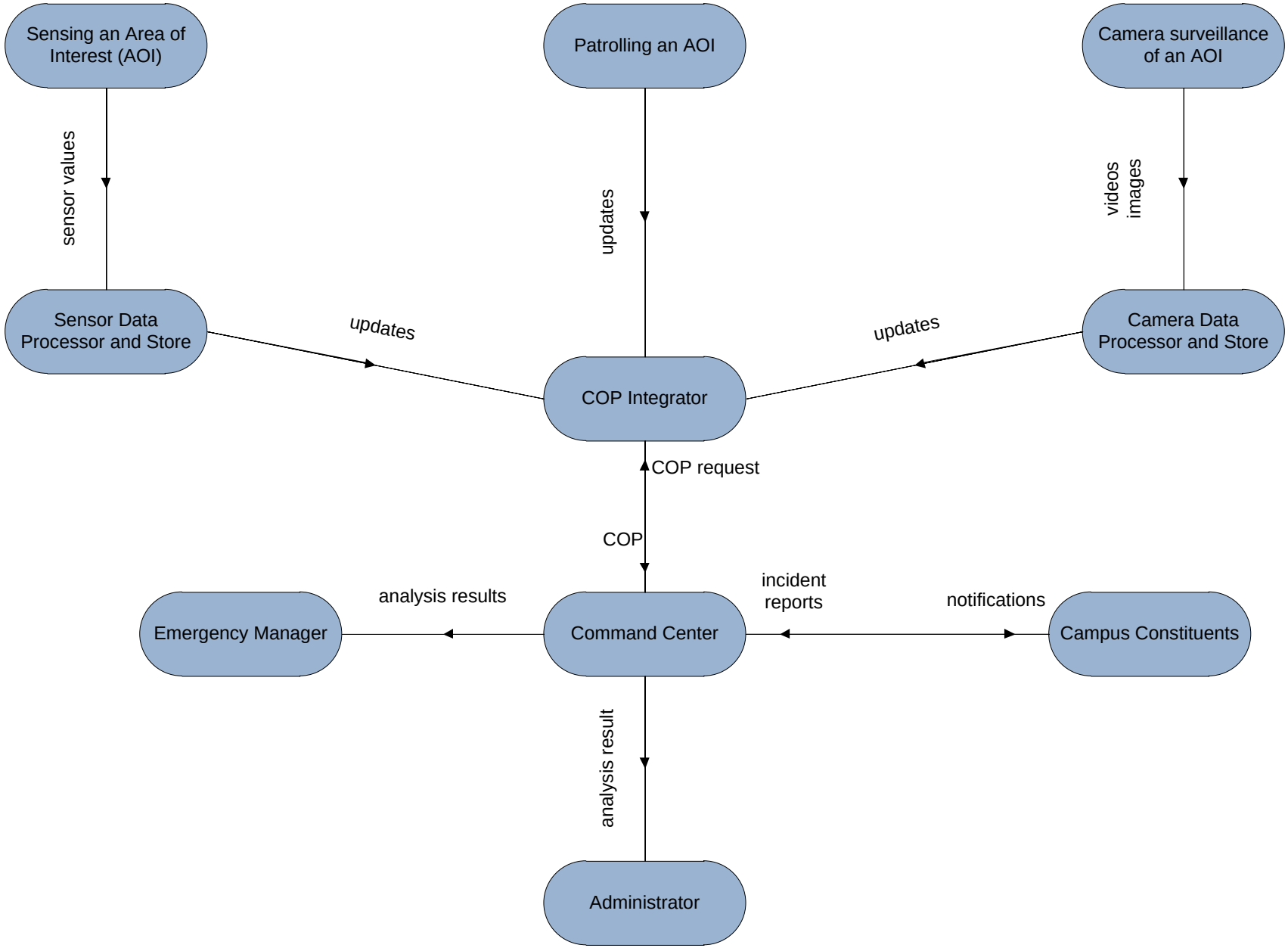


Figure 5-9. OV-2b: SA and SC operational resource flow description

5.13. OV-4A: ORGANIZATIONAL RELATIONSHIPS CHART

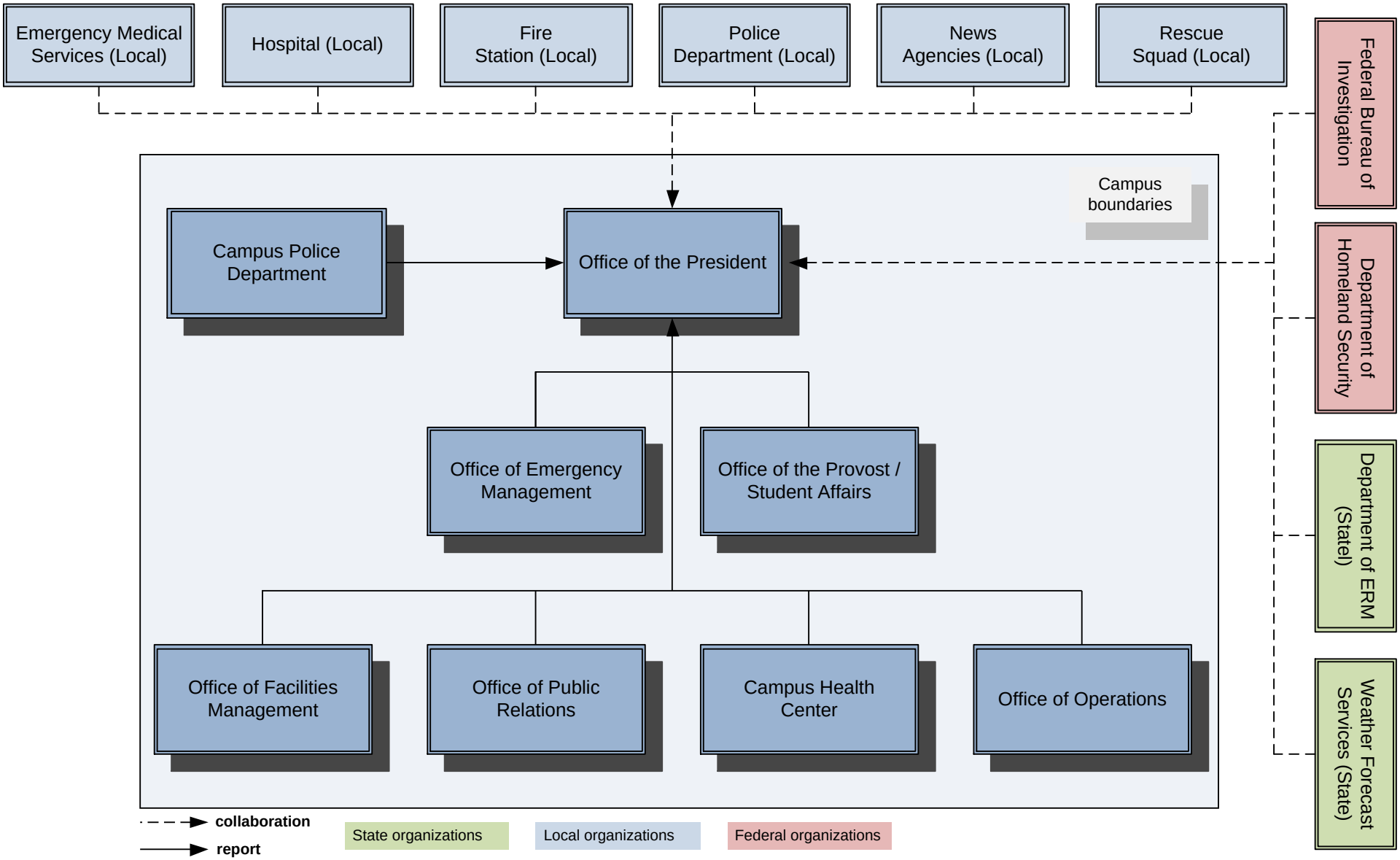


Figure 5-10. OV-4a: Organizational relationships chart

5.14. OV-4B: ROLE-BASED ORGANIZATIONAL RELATIONSHIPS CHART

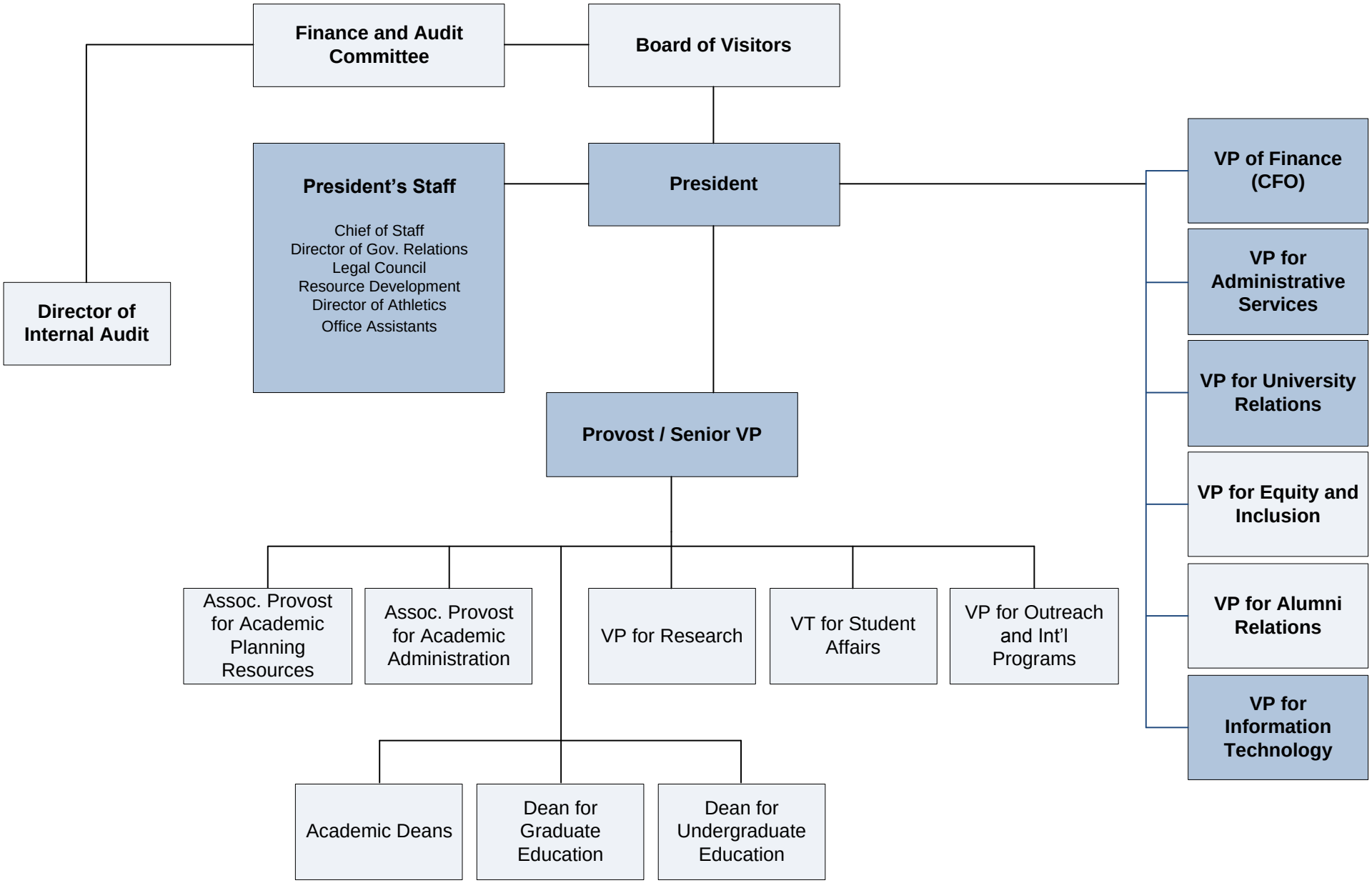


Figure 5-11. OV-4b: Role-based organizational relationship chart

5.15. OV-5A: ERM OPERATIONAL ACTIVITY DECOMPOSITION TREE

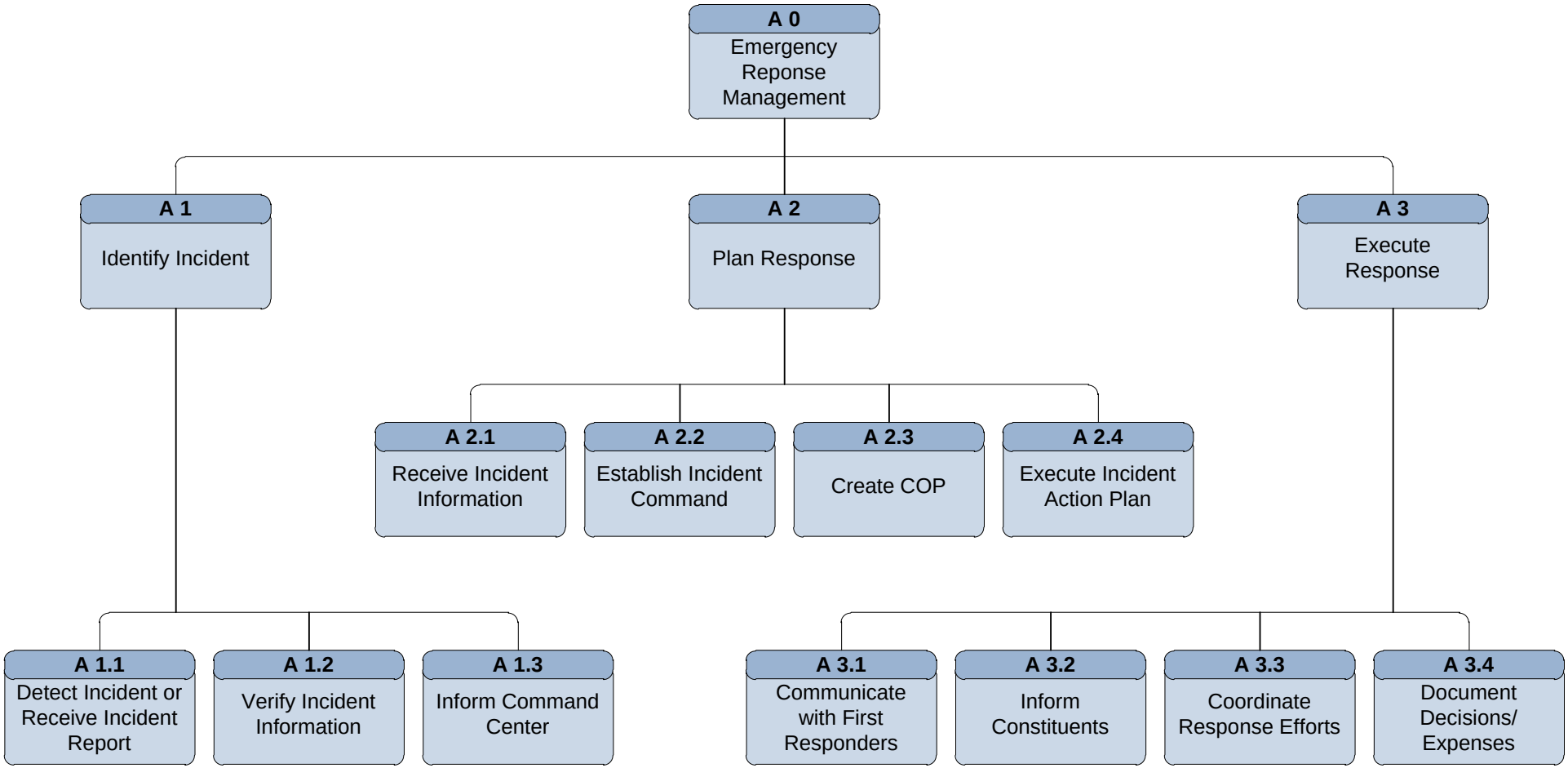


Figure 5-12. OV-5a: ERM operational activity decomposition tree

5.16. OV-5A: SA AND SC OPERATIONAL ACTIVITY DECOMPOSITION TREE

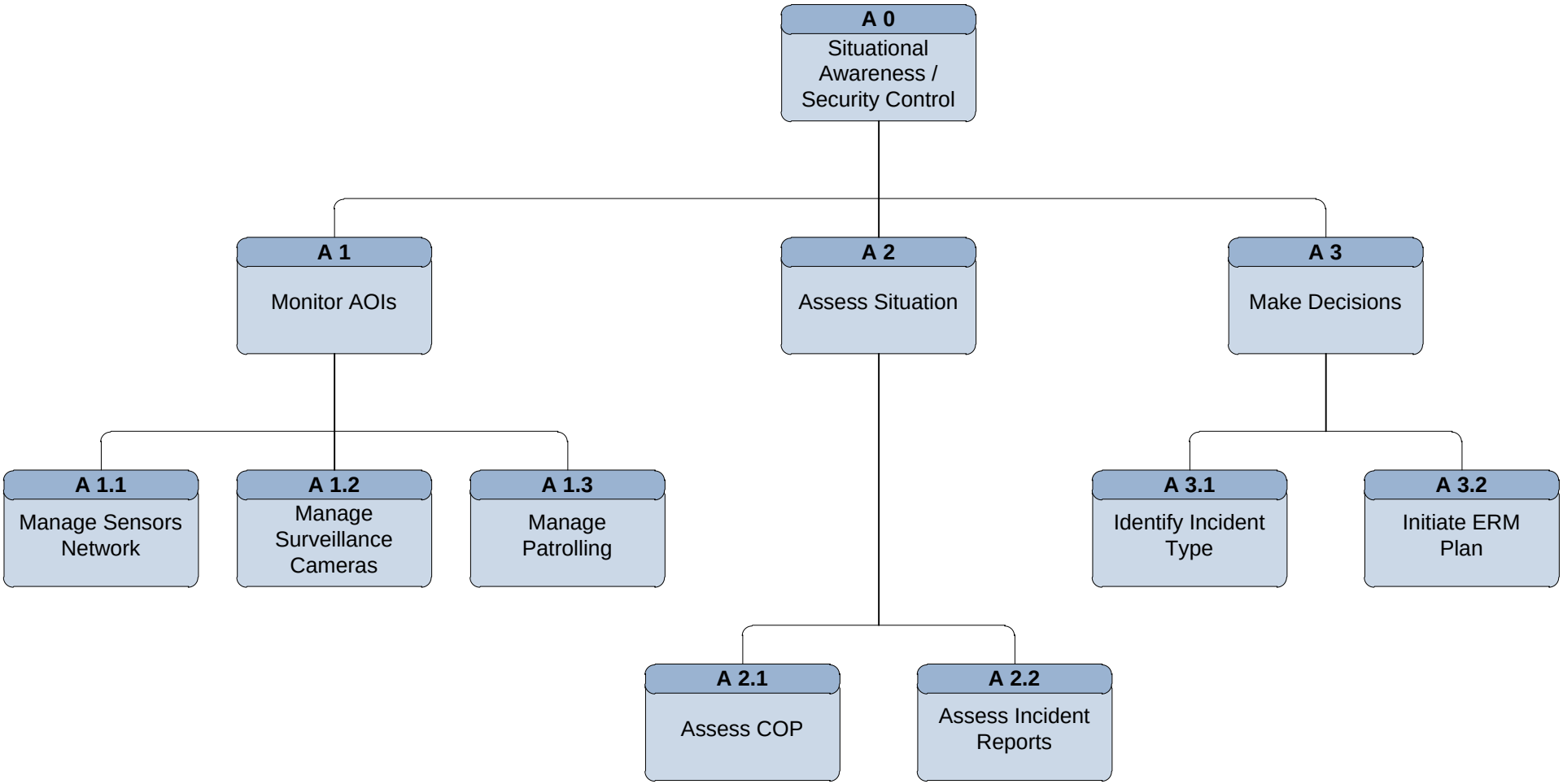


Figure 5-13. OV-5a: SA and SC operational activity decomposition tree

5.17. OV-5B: ERM OPERATIONAL ACTIVITY MODEL

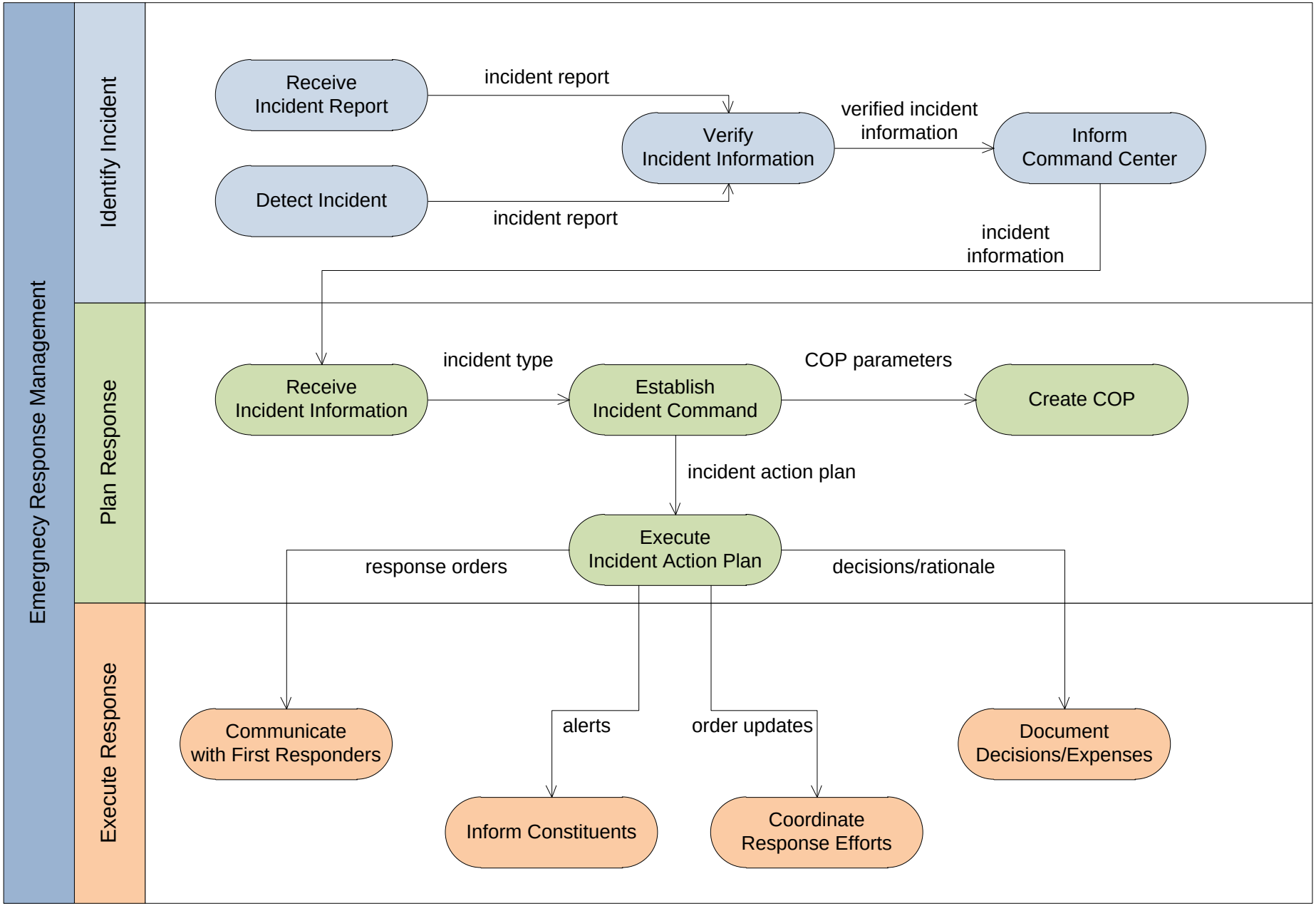


Figure 5-14. OV-5b: ERM operational activity model

5.18. OV-5B: SA AND SC OPERATIONAL ACTIVITY MODEL

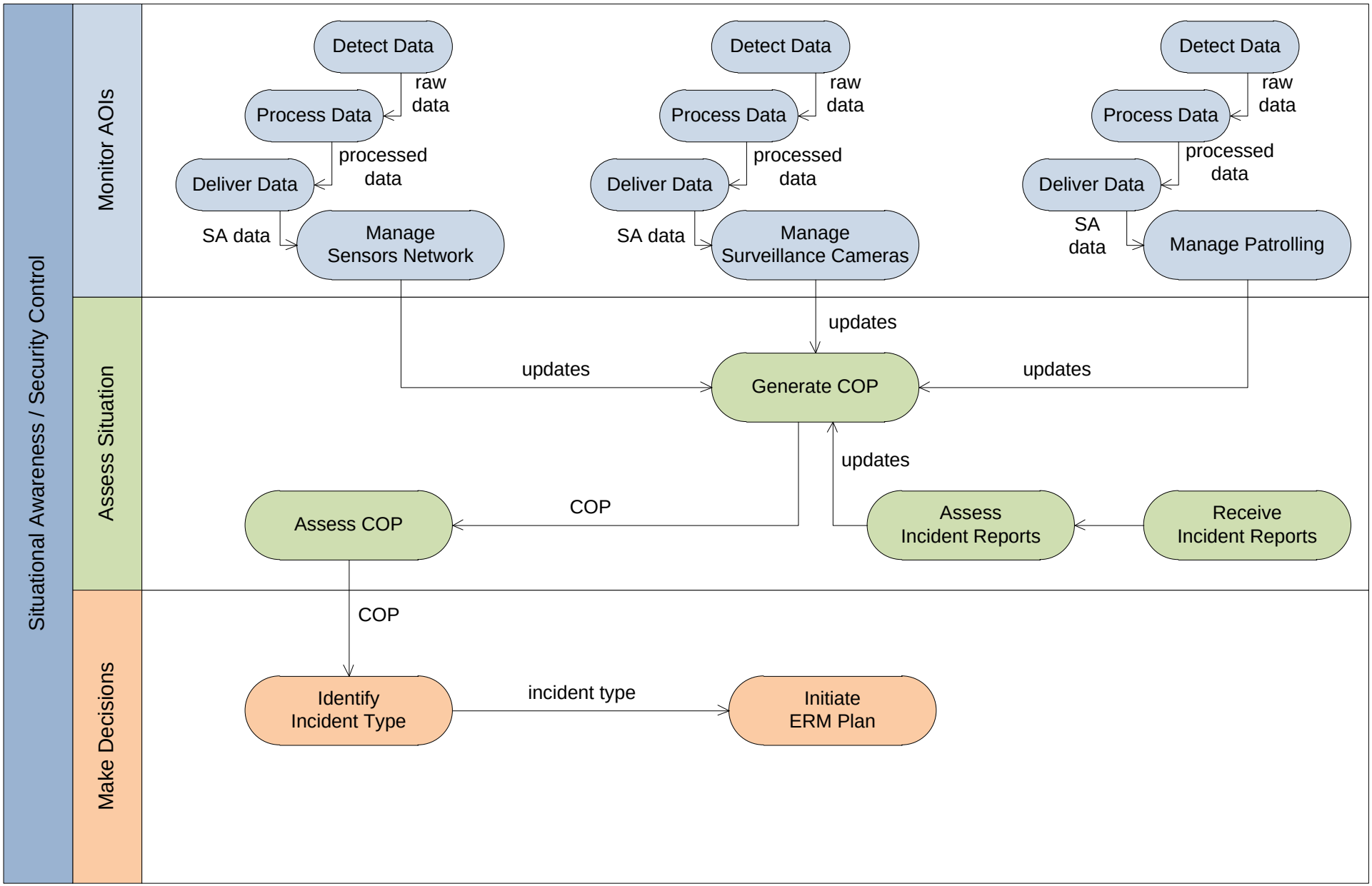


Figure 5-15. OV-5b: SA and SC operational activity model

5.19. SV-1A: SYSTEM CONTEXT MODEL

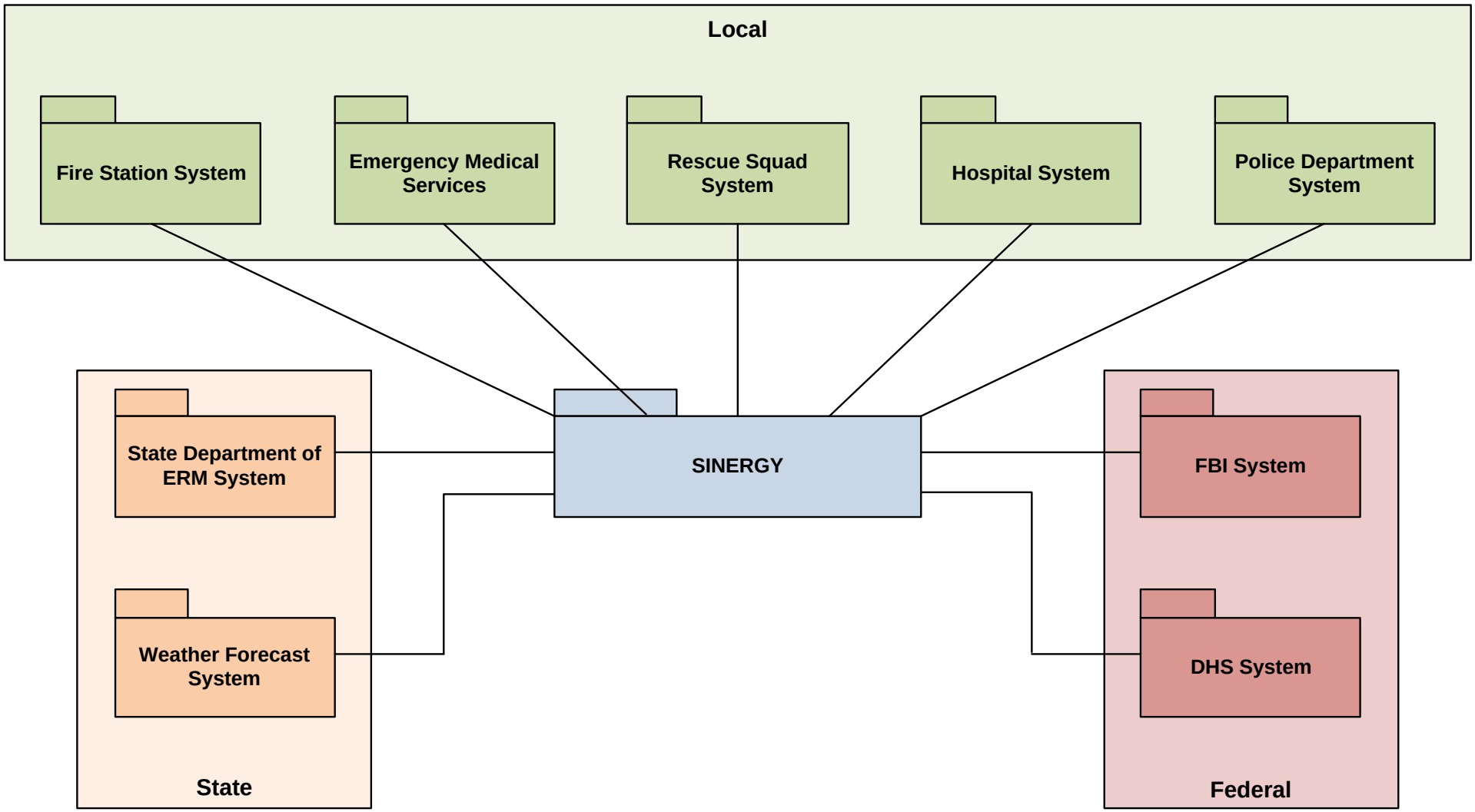


Figure 5-16. SV-1a: System context model

5.20. SV-1B: SYSTEMS INTERFACE DESCRIPTION

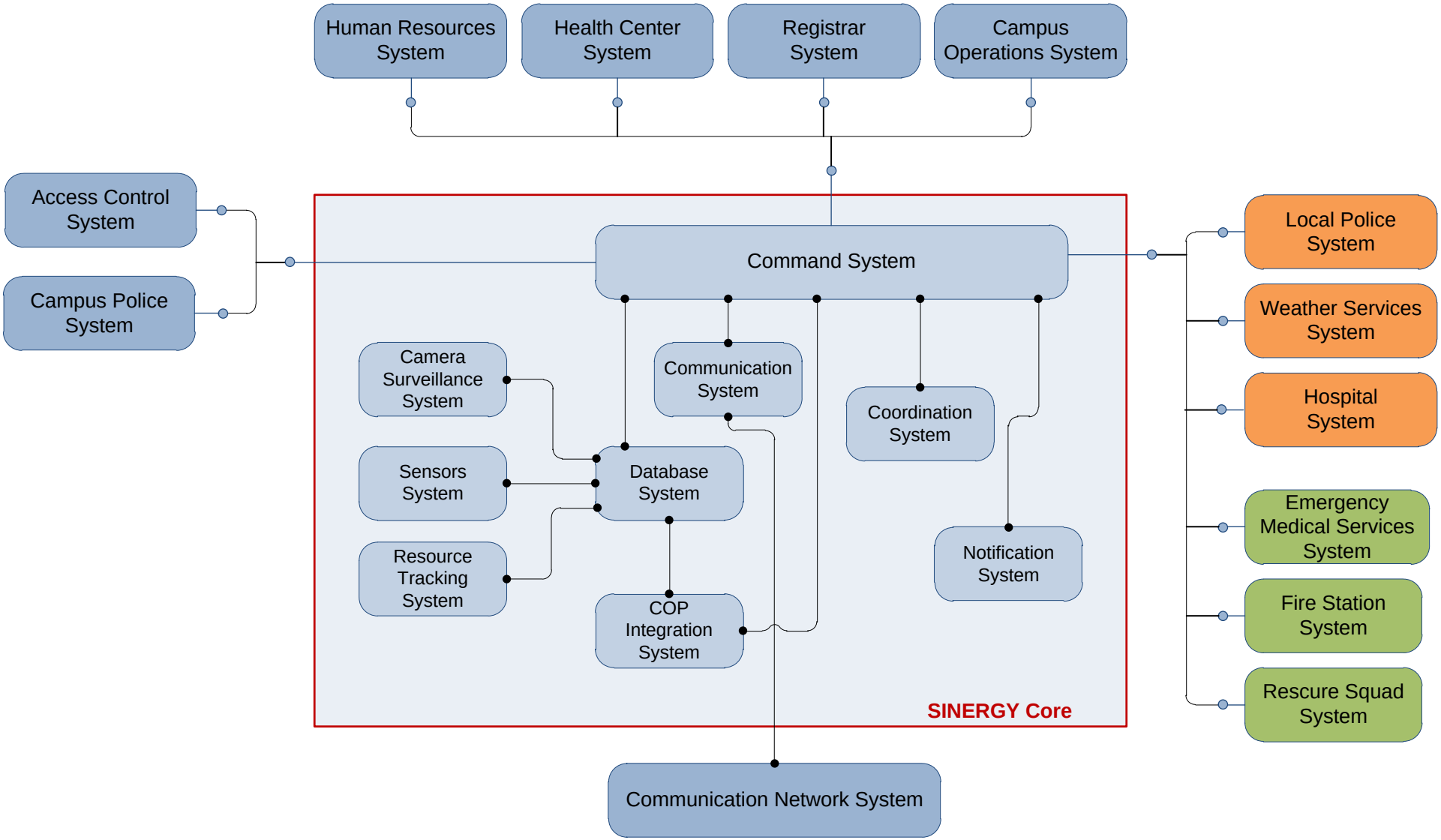


Figure 5-17. SV-1b: Systems interface description

5.21. SV-2: SYSTEMS RESOURCE FLOW DESCRIPTION

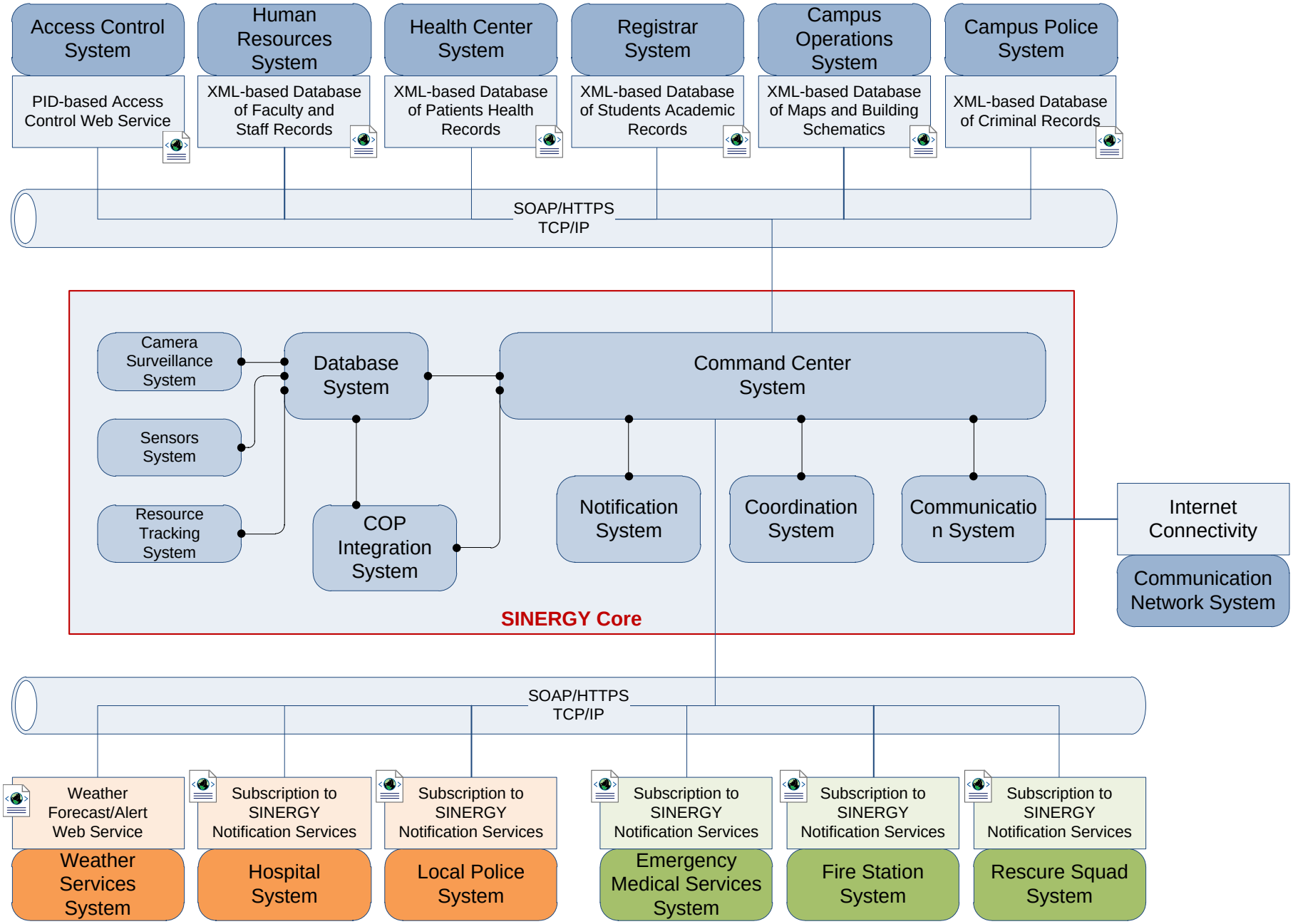


Figure 5-18. SV-2: Systems resource flow description

5.22. SV-3: SYSTEMS-SYSTEMS MATRIX

	Access Control System	Human Resources System	Health Center System	Campus Operations System	Campus Police System	Communication Network System	Weather Services System	Hospital System	Local Police System	EMS System	Fire Station System	Rescue Squad System	Command Center System	Notification System	Coordination System	Communication System	Database System	COP Integration System	Camera Surveillance System	Sensors System	Resource Tracking System
Access Control System													X								
Human Resources System													X								
Health Center System													X								
Campus Operations System													X								
Campus Police System													X								
Communication Network System																X					
Weather Services System													X								
Hospital System													X								
Local Police System													X								
EMS System													X								
Fire Station System													X								
Rescue Squad System													X								
Command Center System	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X			
Notification System													X								
Coordination System													X								
Communication System													X			X					
Database System													X						X	X	X
COP Integration System													X				X				
Camera Surveillance System																	X				
Sensors System																	X				
Resource Tracking System																	X				

Figure 5-19. SV-3: Systems-Systems Matrix

5.23. SV-4: SYSTEMS FUNCTIONALITY TAXONOMY

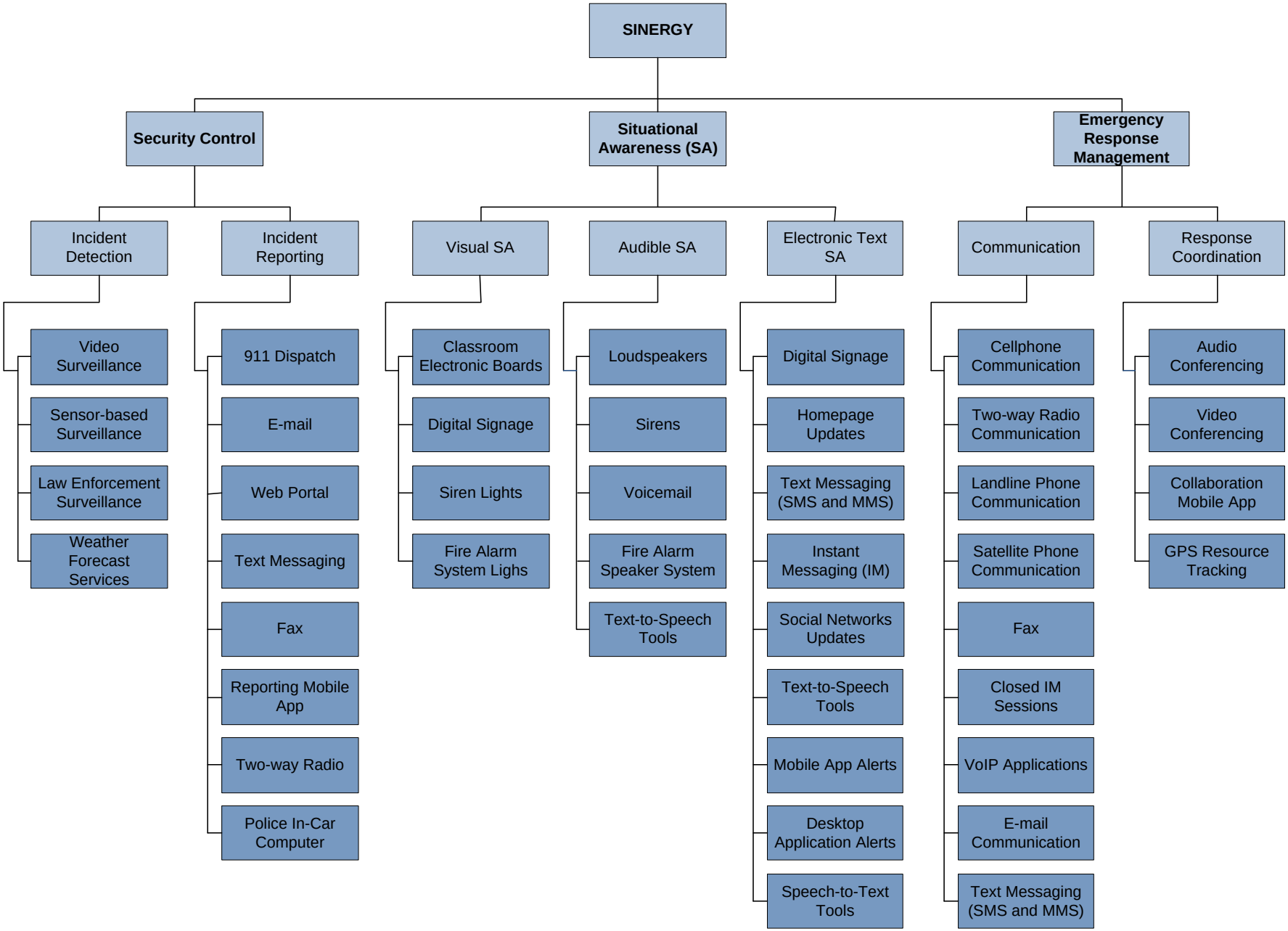


Figure 5-20. SV-4: Systems functionality taxonomy

5.24. SV-5B: OPERATIONAL ACTIVITY TO SYSTEMS TRACEABILITY MATRIX

		Access Control System	Human Resources System	Health Center System	Campus Operations System	Campus Police System	Communication Network System	Weather Services System	Hospital System	Local Police System	EMS System	Fire Station System	Rescue Squad System	Command Center System	Notification System	Coordination System	Communication System	Database System	COP Integration System	Camera Surveillance System	Sensors System	Resource Tracking System
Emergency Response Management Operational Activities	Receive Incident Report						X	X									X			X	X	
	Detect Incident					X				X										X	X	
	Verify Incident Report					X				X				X								
	Inform Command Center	X			X	X	X			X							X					
	Receive Incident Information					X	X			X				X			X					
	Establish Incident Command		X	X	X	X			X	X	X	X	X	X		X						X
	Create COP	X					X							X			X	X	X			
	Execute Incident Action Plan	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
	Communicate with First Responders			X		X	X		X	X	X	X	X	X		X						X
	Inform Constituents		X				X							X	X		X					
	Coordinate Response Efforts						X							X		X	X		X			X
	Document Decisions/ Expenses	X												X				X				X
Situational Awareness/Security Control Operational Activities	Detect Data					X		X		X										X	X	
	Process Data					X		X		X				X					X	X	X	
	Deliver Data						X									X	X					X
	Manage Sensors Network													X							X	
	Manage Surveillance Network													X						X		
	Manage Patrolling					X				X												
	Generate COP													X				X	X			
	Assess COP													X					X			
	Assess Incident Reports					X				X				X								
	Receive Inciden Reports					X				X				X								
	Identify Incident Type													X								
	Initiate ERM Plan													X								

Figure 5-21. SV-5b: Operational activity to systems traceability matrix

5.25. SVCV-1A: SOA CONCEPTUAL LAYERS

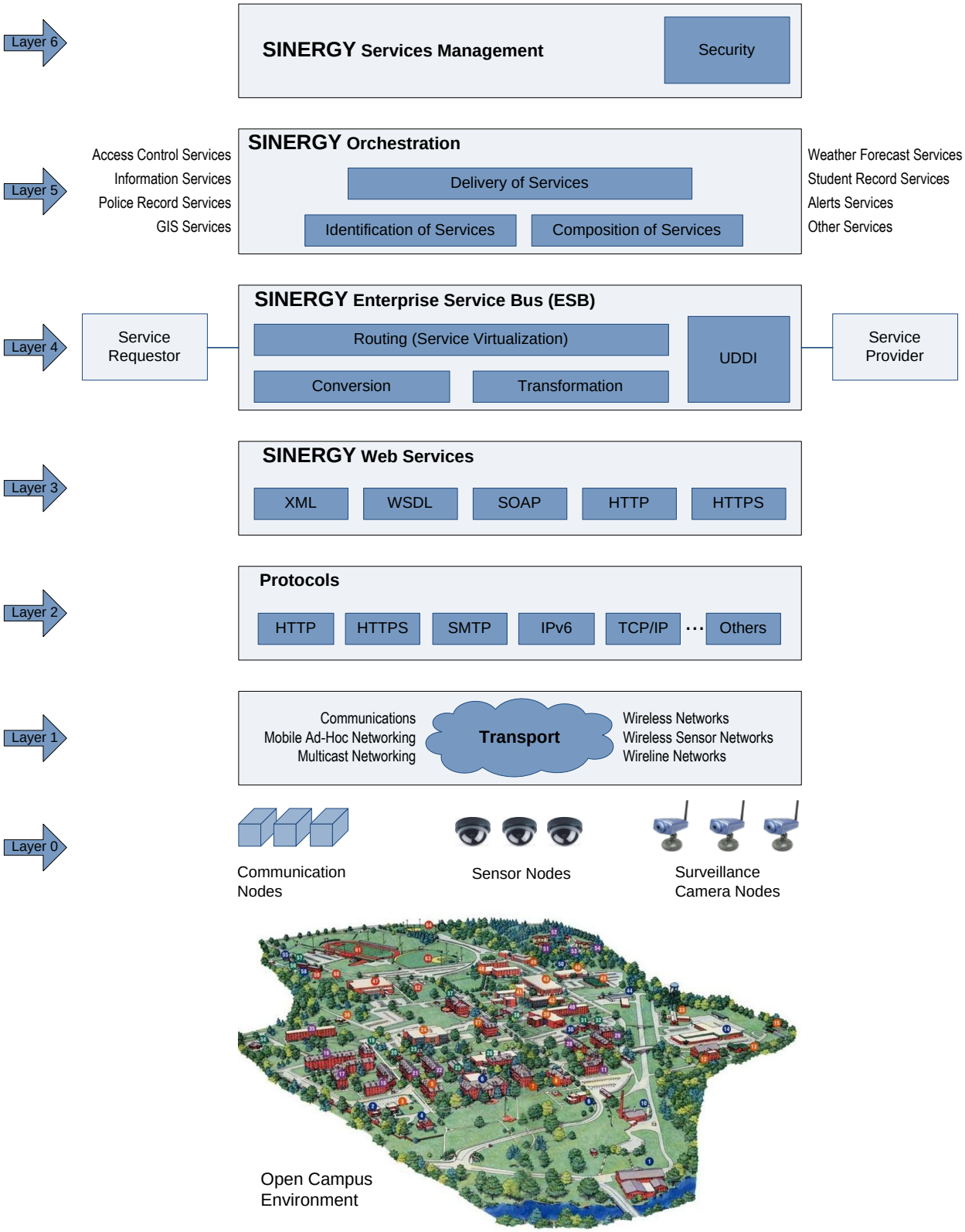


Figure 5-22. SvcV-1a: SOA conceptual layers

5.26. SVCV-1B: SINERGY SOA-BASED ARCHITECTURE

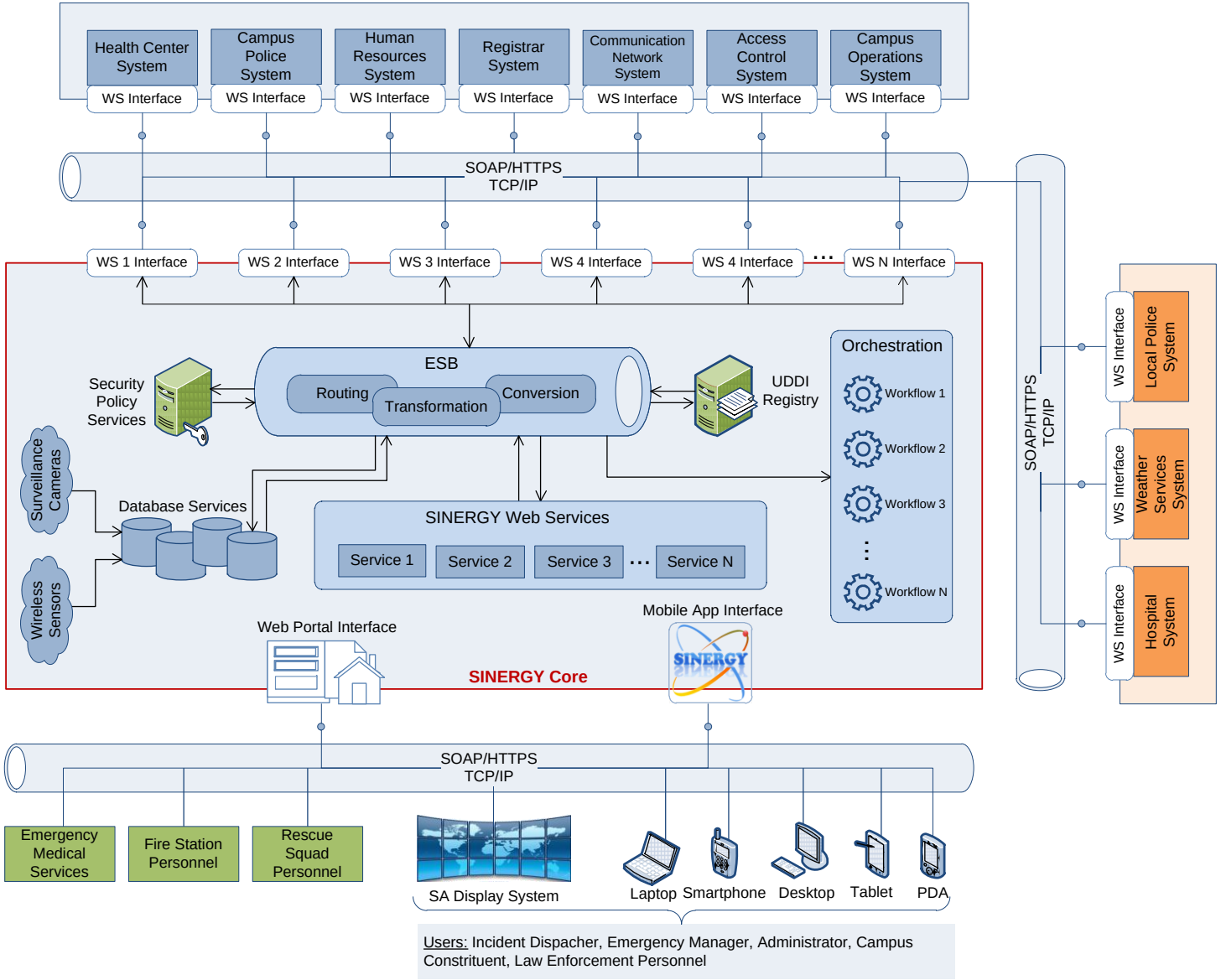


Figure 5-23. SvcV-1b: SINERGY SOA-based architecture

5.27. SVCV-1C: SERVICES CONTEXT DESCRIPTION

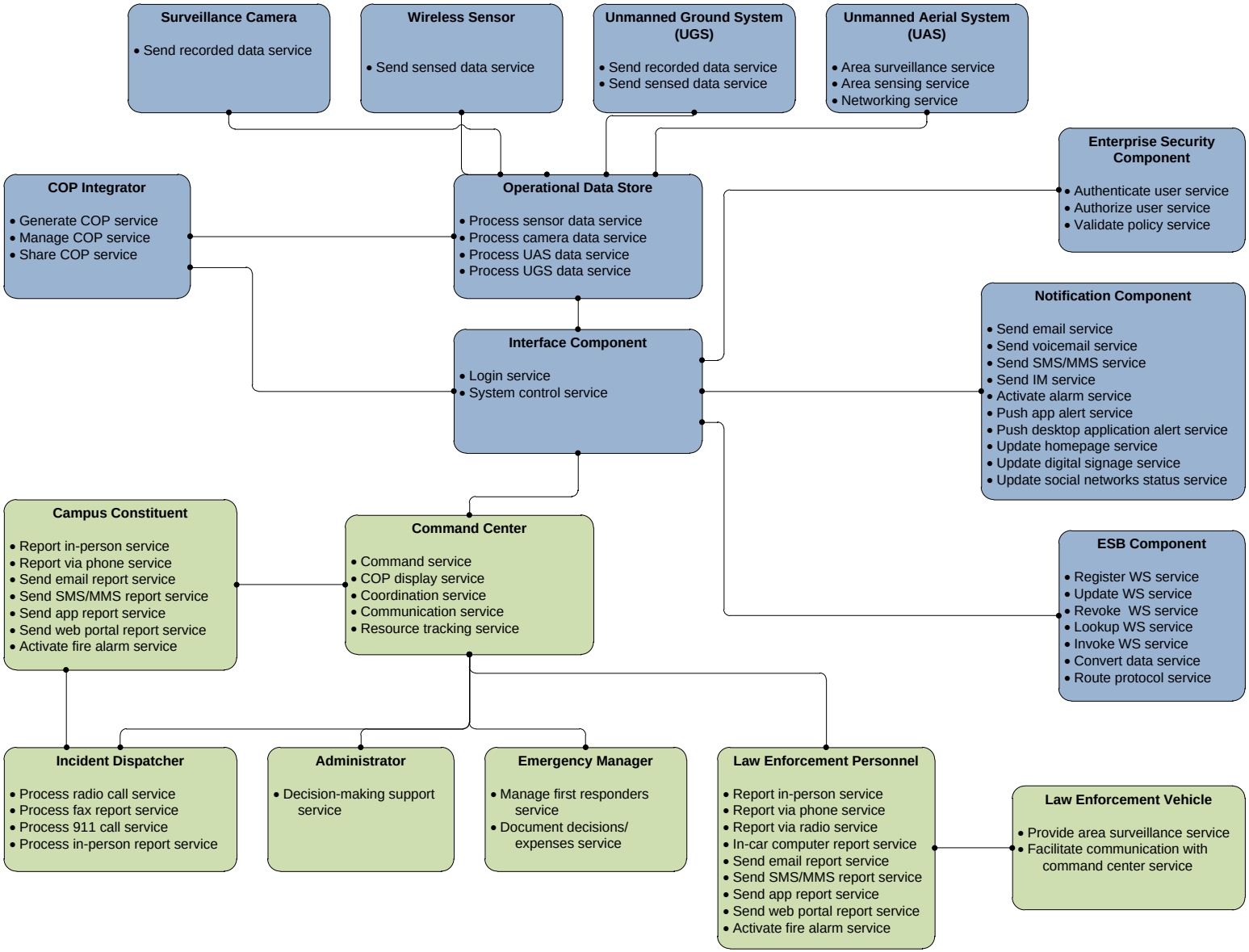


Figure 5-24. SvcV-1c: Services context description

5.28. SVCV-2: SERVICES RESOURCE FLOW DESCRIPTION

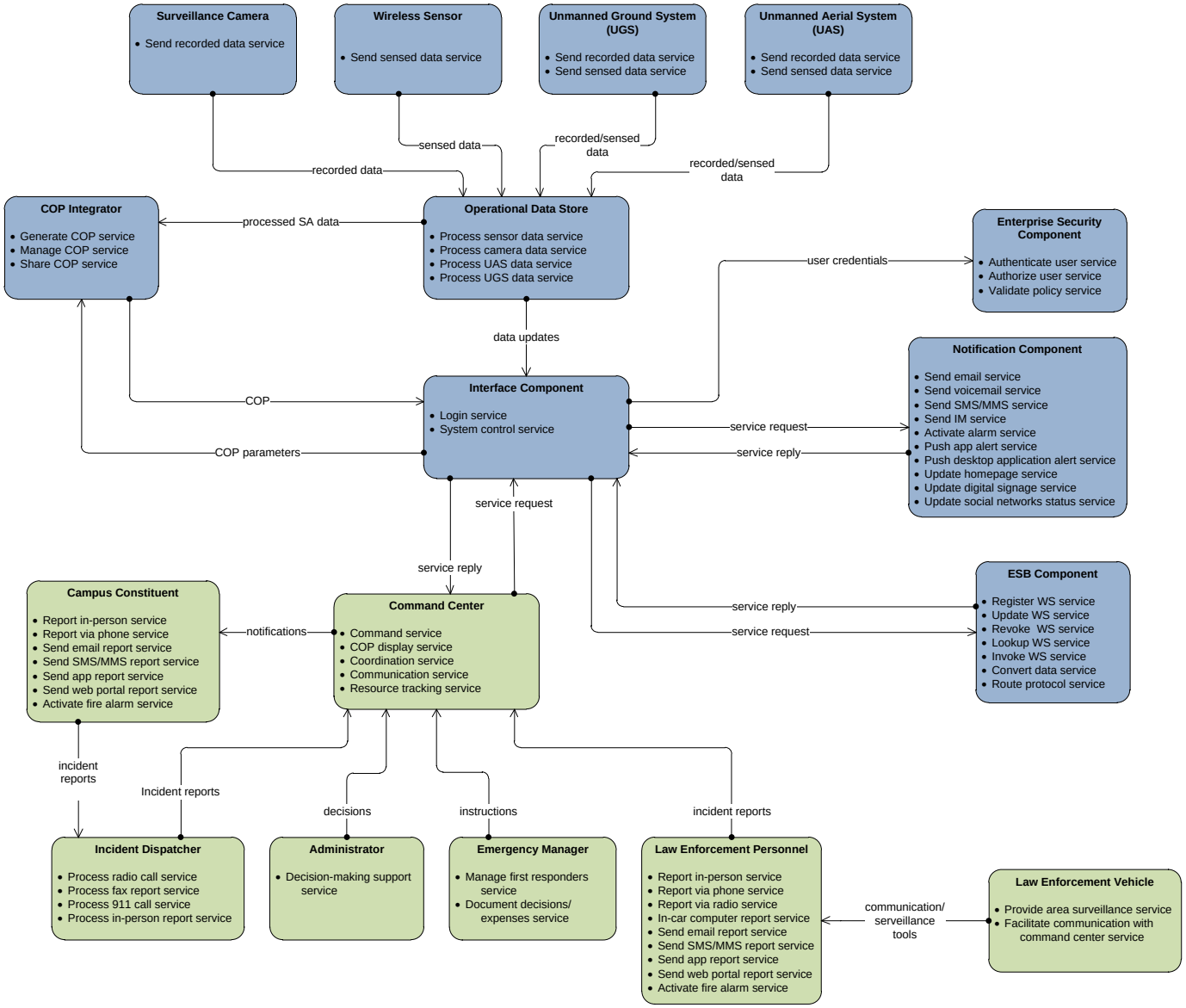


Figure 5-25. SvcV-2: Services resource flow description

5.29. SVCV-3A: SYSTEMS-SERVICES MATRIX

	Access Control System	Camera Surveillance System	Campus Operations System	Campus Police System	Command Center System	Communication Network System	Communication System	Coordination System	COP Integration System	Database System	EAS System	Fire Station System	Health Center System	Hospital System	Human Resources System	Local Police System	Notification System	Rescue Squad System	Resource Tracking System	Sensors System	Weather Services System
Activate Alarm Service				X												X	X				
Activate Fire Alarm Service				X												X	X				
Authenticate User Service	X																				
Authorize User Service	X																				
Command Service					X					X	X	X	X	X	X			X	X		X
Communication Service						X	X														
Convert Data Service								X		X									X		
Coordination Service								X	X												
COP Display Service					X				X												
Decision-Making Support Service			X	X	X					X			X		X	X			X		
Document Decisions/Expenses Service			X		X														X		
Facilitate Communication with Command Center Service			X		X	X	X				X	X	X	X	X			X			X
Generate COP Service					X				X												
In-Car Computer Report Service				X												X					
Invoke WS Service																					
Login Service	X																				
Lookup WS Service								X		X											
Manage COP Service					X				X												
Manage First Responders Service					X			X			X	X	X	X	X			X	X		X
Process 911 Call Service				X	X					X						X					
Process Camera Data Service		X		X					X	X						X					
Process Fax Report Service				X						X						X					
Process In-Person Report Service				X	X					X						X					
Process Radio Call Service					X					X											
Process Sensor Data Service									X	X											
Process UAS Data Service		X							X	X											
Process UGS Data Service		X							X	X											
Provide Area Surveillance Service		X		X	X											X					
Push App Alert Service																					
Push Desktop Application Alert Service																					
Register WS Service								X													
Report In-Person Service																					
Report Via Phone Service																					
Report Via Radio Service																					
Resource Tracking Service																					
Revoke WS Service								X													
Route Protocol Service						X	X	X													
Send App Report Service																					
Send App Report Service																					
Send Email Report Service																					
Send Email Report Service																					
Send Email Service					X												X				
Send IM Service					X												X				
Send Recorded Data Service					X																
Send Sensed Data Service					X																
Send SMS/MMS Report Service																					
Send SMS/MMS Report Service																					
Send SMS/MMS Service					X												X				
Send Voicemail Service					X												X				
Send Web Portal Report Service																					
Send Web Portal Report Service																					
Share COP Service					X				X								X				
System Control Service					X																
Update Digital Signage Service					X												X				
Update Homepage Service					X												X				
Update Social Networks Status Service					X												X				
Update WS Service								X													
Validate Policy Service	X																				

Figure 5-26. SvcV-3a: Systems-services matrix

5.30. SVCV-4: SERVICES CATEGORIZATION

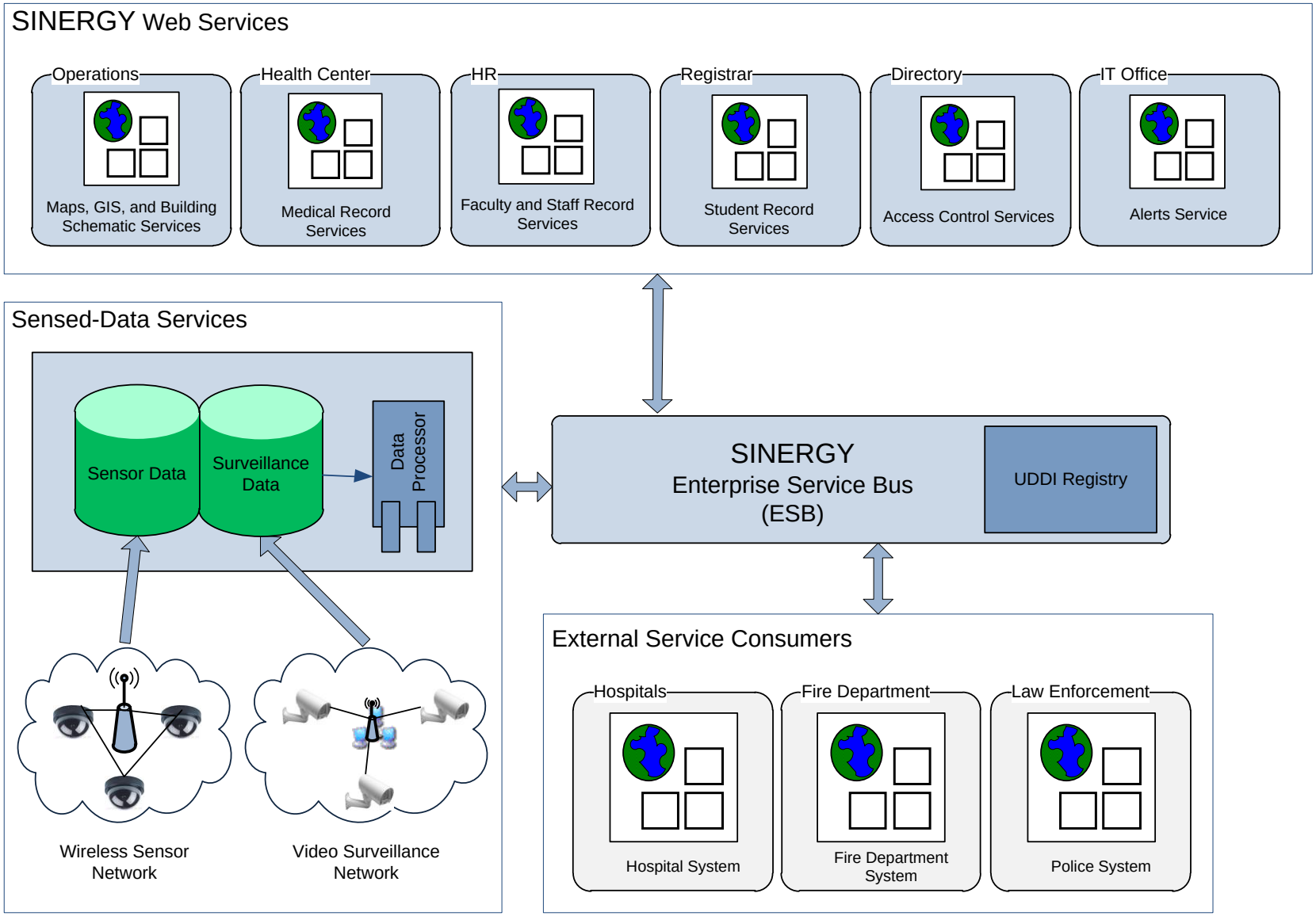


Figure 5-27. SvcV-4: Services categorization

5.31. SVCV-5: OPERATIONAL ACTIVITY TO SERVICE TRACEABILITY MATRIX

	Emergency Response Management Operational Activities												Situational Awareness/Security Control Operational Activities												
	Receive Incident Report	Detect Incident	Verify Incident Report	Inform Command Center	Receive Incident Information	Establish Command	Create COP	Execute Incident Action Plan	Communicate With Other Responders	Inform Complaints	Coordinate Disaster Efforts	Document Disaster Expenses	Detect Data	Process Data	Deliver Data	Manage Sensor Network	Manage Surveillance Network	Manage Featuring	Generate COP	Assess COP	Assess Incident Risk Score	Receive Incident Risk Score	Identify Incident Type	Initiate Incident Plan
Activate Alarm Service	X									X												X		
Activate Fire Alarm Service	X									X												X		
Authenticate User Service			X					X				X				X	X			X	X			X
Authorize User Service			X					X				X				X	X			X	X			X
Command Service	X		X	X	X	X		X	X		X	X				X	X	X	X	X	X	X	X	X
Communication Service	X			X	X	X		X	X	X	X		X		X			X				X		X
Convert Data Service				X	X																			
Coordination Service								X	X		X												X	X
COP Display Service			X				X												X	X				
Decision-Making Support Service			X	X		X		X		X	X	X						X	X	X	X		X	X
Document Decisions/Expenses Service								X				X												X
Facilitate Communication with Command Center Service	X			X				X	X		X							X		X		X	X	X
Generate COP Service			X				X												X	X				
In-Car Computer Report Service	X																					X		
Invoke WS Service																								
Login Service			X	X	X			X				X				X	X			X	X			X
Lookup WS Service																								
Manage COP Service			X				X												X	X				
Manage First Responders Service									X		X													
Process 911 Call Service			X																		X			
Process Camera Data Service			X										X	X							X			
Process Fax Report Service			X																		X			
Process In-Person Report Service			X																		X			
Process Radio Call Service			X															X			X			
Process Sensor Data Service			X										X	X		X					X			
Process UAS Data Service		X	X										X	X		X	X				X			
Process UGS Data Service		X	X										X	X		X	X				X			
Provide Area Surveillance Service		X	X								X							X						
Push App Alert Service										X														
Push Desktop Application Alert Service										X														
Register WS Service																								
Report In-Person Service	X																					X		
Report Via Phone Service	X																					X		
Report Via Radio Service	X																	X				X		
Resource Tracking Service									X															
Revoke WS Service																								
Route Protocol Service					X																			
Send App Report Service	X																					X		
Send App Report Service	X																					X		
Send Email Report Service	X																					X		
Send Email Report Service	X																					X		
Send Email Service										X														
Send IM Service										X														
Send Recorded Data Service		X											X		X		X							
Send Sensed Data Service		X											X		X	X								
Send SMS/MMS Report Service	X																					X		
Send SMS/MMS Report Service	X																					X		
Send SMS/MMS Service										X														
Send Voicemail Service										X														
Send Web Portal Report Service	X																					X		
Send Web Portal Report Service	X																					X		
Share COP Service			X				X				X								X	X				
System Control Service	X		X		X	X		X	X		X	X								X	X	X	X	X
Update Digital Signage Service										X														
Update Homepage Service										X														
Update Social Networks Status Service																								
Update WS Service																								
Validate Policy Service	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 5-28. SvcV-5: Operational activity to services traceability matrix

CHAPTER 6: SINERGY ARCHITECTURE EVALUATION

6.1. INTRODUCTION

Architecture is an important life cycle product and represents the most critical decisions about development, deployment, and management of the system. These architectural decisions are often costly to change once the development starts. For instance, a change that requires the migration into an SOA-based architecture from P2P architecture is technically challenging and can be substantially expensive. Furthermore, architecture is important because it affects the entire project and involves many stakeholders of the system. It serves as a communication means between various stakeholders. Thus, architecture should be analyzed and evaluated early to assess whether it meets it stated objectives.

A good architecture is not a predictor of the quality of the system. In other words, one can still develop a bad quality system based on a good architecture. However, an ill-articulated architecture is definitely a precursor for a bad quality system. Therefore, the primary purpose of architecture evaluation is to assess the consequences of architectural decisions in light of quality attribute requirements. It serves primarily as a risk identification mechanism to avoid developing a system that may not meet its IUs.

Architecture is evaluated with regards to a set of quality attributes that it needs to enable the system to exhibit. Quality attributes should be identified from different stakeholder perspectives to ensure that the evaluation is comprehensive. End users, for instance, care about usability, performance, availability, and security. Developers care about system maintainability, reusability, and testability. The business community care about time to market, cost and benefits, and integration with legacy systems.

Architecture evaluation definitely has a cost, especially in delaying the start of system development. However, the benefits outweigh the cost in many ways. First, architecture decisions are costly to change; therefore, evaluating these decisions early is cost saving and helps identify architecture errors before they are built into the system. Second, conducting an architecture evaluation forces the development team to have a fully-articulated and -described architecture, which in of itself is an asset to the organization. The reason is that architecture is a blueprint of the entire system and is used as a reference through the development life cycle. Finally, evaluation makes certain that the architecture ensures that quality attributes are thought about, tradeoffs are addressed, and risks are mitigated.

The evaluation of SINERGY architecture is a complex endeavor and requires a methodological approach to address this complexity. This architecture influences many system quality attributes such as dependability, interoperability, maintainability, modifiability, and performance. The goal of this evaluation is to find out how well SINERGY architecture supports the desired set of quality characteristics under the IUs of SINERGY. This chapter describes the objective, approach, and results of this evaluation.

6.2. OBJECTIVE

Any evaluation effort leads to optimal results only when it is characterized by technical, managerial, and financial independence [IEEE 1998]. Technical independence requires that people other than those who developed the architecture do the evaluation. Managerial independence requires that the management of the evaluation effort be independent from the management of the architecture development effort. Finally, financial independence requires that the control over the evaluation budget be independent from that of the architecture development effort. For the purposes of this research, independence is not part of our evaluation approach.

The objective of SINERGY evaluation is twofold. First, we intend to subjectively perform a self-evaluation of the architecture models we created with regards to the quality attributes specified in SINERGY capabilities specification. Evaluations performed by the same architecture teams who developed the architecture is common in many in-house development efforts where architects are the ones who conduct architecture evaluations to catch architectural defects. The second component of our objective for self-evaluation is to show how an architecture evaluation of a complex system of systems is conducted and contribute this experience as a case study to the architecture knowledge, which lacks real-life architecture examples. The rest of this chapter focuses explicitly on the first part of the objective, and implicitly addresses the second.

To understand the second part of the objective, we need to look at the state of the art in comparison with the state of the practice of architecture knowledge. As we discussed in Chapter 2, architecture is becoming mainstream. However, teaching and training architects lacks real-life architecture examples to demonstrate how to apply methods and tools for architecting, describing, and evaluating architectures. The main reason is that organizations consider their architectures as proprietary and valued assets. Furthermore, open-source communities focus primarily on source code and other non-architecture information. Therefore, even for open-source projects, architecture knowledge that could be reused, learned, or taught is lacking. On the other hand, researchers use their methods and tools on simple proof-

of-concept systems, which are often orders of magnitude less in terms of complexity in comparison with large-scale systems of systems. For these reasons, this chapter serves an additional purpose. We intend to demonstrate how to apply the Military System Architecture Assessment Methodology (MSAAM) on a system of systems architecture, and make that knowledge available for the architecture community.

6.3. APPROACH

As we argued in the description of the Process of Architecting in Chapter 3, a risk-driven approach is advocated for architecture evaluation. The evaluation should consider all potential risks including the following: Acceptance risk, integration risk, interface risk, performance risk, reliability risk, reproducibility risk, supportability risk, technological risk, and utility risk. The purpose of this assessment is to ensure that major pitfalls and "show stoppers" are mitigated early on in the process before major resources are invested.

We employ MSAAM as the evaluation methodology of choice. MSAAM advocates a risk-driven, mission-oriented approach. It facilitates the decomposition of the architecture quality into a manageable set of indicators that can be evaluated directly. The evaluation results of each indicator are then aggregated to form an overall assessment of the quality of the architecture.

MSAAM consists of seven stages and each stage consists of defined steps as described in [Balci and Ormsby 2008]. These seven stages are:

1. Identification of system missions
2. Description of use cases illustrating the missions
3. Use of SMEs
4. Identification of quality indicators for the architecture
5. Establishing relative criticality weighting of indicators
6. Score assignments for indicators
7. Aggregation of evaluation results.

Architecture evaluation is considered to be a confidence building activity. The more comprehensive and detailed the evaluation is, the more confidence we can gain in the evaluation. Four major perspectives or four Ps influence the architecture quality: Product, process, people, and project. Architecture evaluation can be approached from any one of the four Ps, but a combination of all four provides the best balance and results in a much higher level of confidence in the evaluation. Therefore, an architecture should be assessed from the four perspectives by way of assessing:

- the architecture itself as a **product**,
- the **process** used in creating the architecture,
- the quality of the **people** employed in creating the architecture, and
- architecture development **project** characteristics (e.g., capability maturity, documentation, planning, risk management).

6.4. RESULTS

SINERGY architecture evaluation follows the stages of MSAAM as explained below.

6.4.1. Missions and Use Cases

SINERGY missions come from its main goal: To achieve SA, SC, and ERM. Figure 6-1 depicts the taxonomy of missions for SINERGY. These missions fall within the purview of the IUs of SINERGY described in the capabilities specification. They represent what the system should enable its users to accomplish.

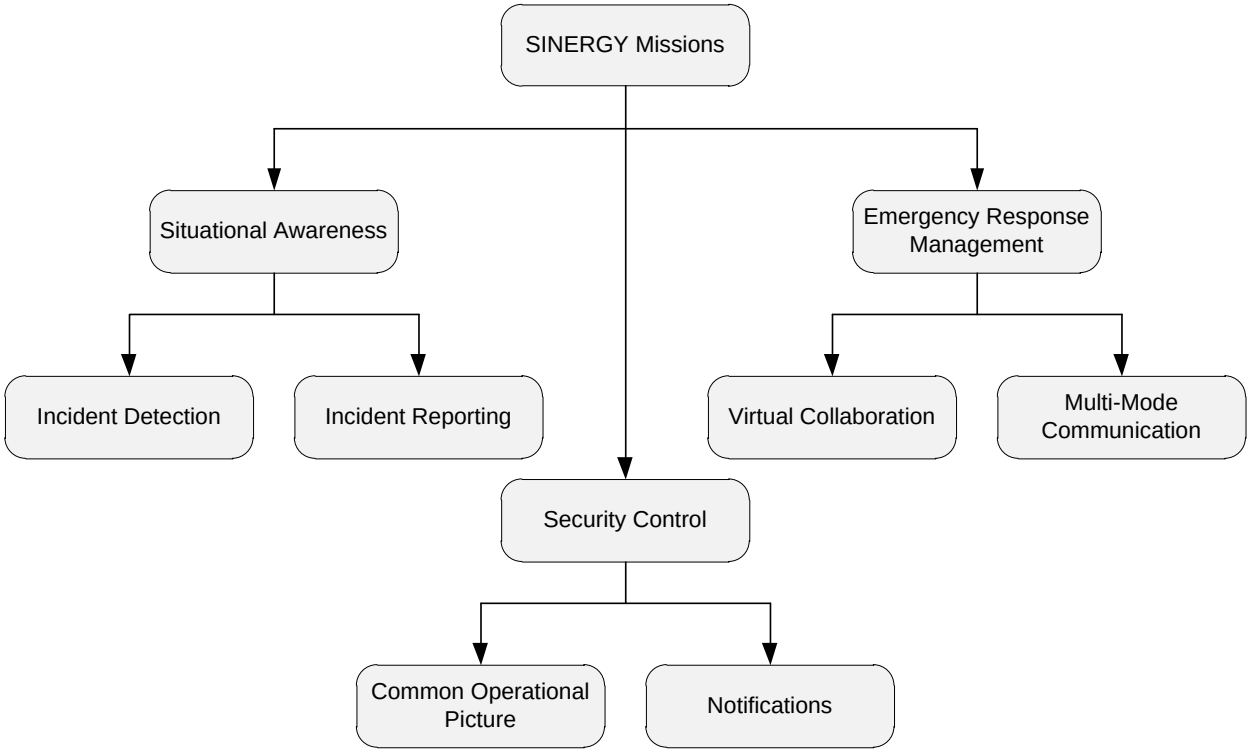


Figure 6-1. A taxonomy of missions for SINERGY

Use cases illustrate how SINERGY missions will be accomplished. Uses cases exercise the capabilities of the SINERGY to assess whether it will express and meet the stated quality attributes. In this section, we group the use cases by the corresponding mission they achieve and evaluate whether it is covered by SINERGY architecture models. Table 6-1 shows the mapping between the missions, uses cases, and some of SINERGY DoDAF models that address those missions.

Table 6-1. List of use cases by mission and corresponding DoDAF models

Use Case	Is Covered?	DoDAF Model(s)
Incident Detection		
Detect by video surveillance camera	✔	OV-1b, OV-2b, SV-5b
Detect by surveillance sensor	✔	OV-1b, OV-2b, SV-5b
Detect by law enforcement	✔	OV-1b, OV-2b, SV-5b
Detect thru weather service	✔	OV-1b, OV-2b, SV-5b
Incident Reporting		
Report by phone to 911 dispatch	✔	OV-1a, OV-2a, SV-5s
Report by e-mail	✔	OV-1a, OV-2a, SV-5b
Report thru web portal	✔	OV-1a, OV-2a, SV-5b
Report by text message	✔	OV-1a, OV-2a, SV-5b
Report by fax	✔	OV-1a, OV-2a, SV-5b
Report thru mobile app	✔	OV-1a, OV-2a, SV-5b
Report by two-way radio	✔	OV-1a, OV-2a, SV-5b
Report thru police in-car computer	✔	OV-1a, OV-2a, SV-5b
Common Operational Picture		
Establish command center	✔	OV-5a SA/SC
Manage video surveillance network	✔	OV-5a SA/SC
Manage sensor surveillance network	✔	OV-5a SA/SC
Access information from local databases	✔	OV-5a SA/SC
Query information from external databases	✔	OV-5a SA/SC
Notifications		
Notify thru text message	✔	OV-1a, OV-5b
Notify by e-mail	✔	OV-1a, OV-5b
Notify thru website update	✔	OV-1a, OV-5b
Notify thru voicemail	✔	OV-1a, OV-5b

Notify by sirens	✔	OV-1a, OV-5b
Notify thru digital signage message	✔	OV-1a, OV-5b
Notify thru social network updates	✔	OV-1a, OV-5b
Notify thru mobile app alert	✔	OV-1a, OV-5b
Notify thru desktop alert	✔	OV-1a, OV-5b
Virtual Collaboration		
Setup video conference	✔	SvcV-1c, OV-5b
Setup audio conference	✔	SvcV-1c, OV-5b
Establish mobile app collaboration	✔	SvcV-1c, OV-5b
Track GPS resources	✔	SvcV-1c, OV-5b
Multi-Mode Communication		
Establish phone communication	✔	SvcV-1c, OV-5b
Establish two-way radio communication	✔	SvcV-1c, OV-5b
Establish satellite phone communication	✔	SvcV-1c, OV-5b
Establish fax communication	✔	SvcV-1c, OV-5b
Establish closed IM session	✔	SvcV-1c, OV-5b
Establish VoIP communication	✔	SvcV-1c, OV-5b

6.4.2. Subject Matter Experts

The evaluation is conducted subjectively as a self-evaluation by the architect of SINERGY. We conduct this self-evaluation to assess whether the architecture models meet the quality attributes of SINERGY. Additionally, the purpose of this self-evaluation is to demonstrate how architecture evaluation is conducted using MSAAM. Therefore, this exercise will serve as a learning tool for architects who conduct architecture evaluations.

6.4.3. Risks

Table 6-2 illustrates the assessment of a set of risks relevant to SINERY, and shows the rationale for each assessment. We use the following score set to rate each risk:

- Low (L): The risk is sufficiently addressed in the architecture.
- Medium (M): The risk is adequately addressed in the architecture but further consideration is still needed during downstream life cycle phases.

- High (H): The risk is not addressed in the architecture and is considered a priority during the next downstream development.

Table 6-2. Assessment of risk factors and its rationale

Risk	Level	Rationale
Acceptance Risk	**L**	Acceptance criteria are focused on meeting stakeholder needs and quality attributes, which were the basis for building the architecture
Integration Risk	**L**	SOA conceptual layering addresses integration issues through virtualization provided by the ESB
Interface Risk	**M**	Web Services standards (i.e., WSDL and XML) address interfacing requirements by providing industry standards for communication among services. However, wrapping legacy systems may be needed
Performance Risk	**H**	Performance need to be further addressed based on where system components are deployed geographically and based on network technologies adopted
Reliability Risk	**M**	Reliability is addressed through multi-mode communication and collaboration capabilities. However, separation of normal operation infrastructure from emergency operation infrastructure is key to mitigate this risk
Reproducibility Risk	**M**	Open industry standards ensure replacement of components with minimal effort focusing only on interfaces of components
Supportability Risk	**L**	Having open, well-documented architecture based on DoDAF facilitates supporting maintenance and other activities by avoiding reliance on implicit knowledge kept with architects
Technological Risk	**L**	Separation of concerns and SOA layers makes it manageable to adopt new and future technologies
Utility Risk	**M**	Architecture is build to the specification of a set of representative stakeholders. However, the specific context of implementation may require further assessment of this risk

6.4.4. Four Perspectives

6.4.4.1. Process Assessment

The development of SINERGY architecture followed the Process of Architecting that we defined and described in Chapter 3. It is a comprehensive process that describes architecting in terms of specific goals and specific related practices. It covers all phases of architecting from requirements identification to architecture evaluation. However, when an independent party evaluates an architecture, the following process indicators, shown in Table 6-3, should be considered.

Table 6-3. List of process indicators

Process Indicators	
1.	Does the process have a specific practice for identifying architecturally-relevant requirements?
2.	Does the process have a specific practice for decomposing the system into components?
3.	Does the process have a specific practice for assessing component decomposition?
4.	Does the process have a specific practice for establishing structural relationships among components?
5.	Does the process have a specific practice for establishing behavioral components among components?
6.	Does the process have a specific practice for creating the architecture from an existing one?
7.	Does the process have a specific practice for creating a composite architecture?
8.	Does the process have a specific practice for creating the architecture from scratch?
9.	Does the process have a specific practice for describing the architecture from an overall perspective?
10.	Does the process have a specific practice for describing the architecture from a capability perspective?
11.	Does the process have a specific practice for describing the architecture from a data and information perspective?
12.	Does the process have a specific practice for describing the architecture from an operational perspective?
13.	Does the process have a specific practice for describing the architecture from a project perspective?
14.	Does the process have a specific practice for describing the architecture from a services perspective?
15.	Does the process have a specific practice for describing the architecture from a standards perspective?
17.	Does the process have a specific practice for describing the architecture from a systems perspective?
18.	Does the process have a specific practice for evaluating the architecture from a project perspective?
19.	Does the process have a specific practice for evaluating the architecture from a process perspective?
20.	Does the process have a specific practice for evaluating the architecture from a people perspective?
21.	Does the process have a specific practice for evaluating the architecture from a product perspective?
22.	Does the process have a specific practice for evaluating the architecture based on a risk-driven approach?
23.	Does the process have a specific practice for evaluating the architecture based on a set of scenarios?

6.4.4.2. People Assessment

This is a subjective evaluation and therefore the people perspective is not applicable. However, when an independent party evaluates an architecture, the following people indicators should be considered. These indicators are the result of a collaborative effort by the architecture community to define the skills, knowledge, and duties of the architects. These are compiled into a model described in [Clements *et al.* 2007].

In addition, People Capability Maturity Model Version 2.0 (P-CMM) [SEI 2001] provides a framework to evaluate an organization based on the capability and maturity of its human assets. This model can be used to identify indicators that are relevant to evaluating architects.

Table 6-4 lists high-level indicators that need to be considered when evaluating architects.

Table 6-4. List of people indicators

People Indicators	
1.	Knowledge of architecture concepts
2.	Knowledge of software engineering
3.	Knowledge of specific technologies and platforms
4.	Industry, domain, and enterprise knowledge
5.	Management skills
6.	Communication skills
7.	Inter-personal skills within team
8.	Inter-personal skills with other people
9.	Leadership skills
10.	Effectiveness in workload management
11.	Skills to excel in corporate environment
12.	Skills for handling information
13.	Skills for handling uncertainty and ambiguity
14.	Skills for learning

6.4.4.3. Project Assessment

This project was part of a Ph.D. dissertation research and followed an established schedule based on three milestones: A proposal defense, a research defense, and final defense. Table 6-5 shows the timeline for each of these milestones.

Table 6-5. Project major milestones and delivery dates

Milestone	Start Date	Due Date
Proposal defense	January 1, 2008	April 21, 2009
Research defense	April 22, 2009	December 10, 2010
Final defense	December 11, 2010	April 6, 2011

6.4.4.4. Product Assessment

We use quality attributes as indicators to indirectly assess the quality of SINERGY architecture. *Quality attributes* of a system are those characteristics that affect the quality of the entire system. They are also referred to as quality characteristics and non-functional requirements. Examples include security, interoperability, and performance. The goal of the product evaluation is to find out how well SINERGY architecture will enable a SINERGY implementation to possess a desired set of quality indicators. The following is the list of quality indicators that we use to evaluate SINERGY architecture.

Security: SINERGY shall allow authorized users to control and manage the dissemination of official information to campus constituents through university official channels (e.g., website, alerts, digital signs, e-mails, twitter updates, and Facebook updates.)

Response Flexibility: SINERGY shall provide the capability to disseminate official information to all campus constituents through one or all communication outlets based on the situation to achieve maximum reach and flexibility of the response.

Interoperability: SINERGY shall be interoperable on three levels.

1. *Terminology:* SINERGY shall be compliant with the ICS, which advocates a common terminology defining organizational functions, facilities, resource description, and position titles.
2. *Interfaces:* SINERGY shall use Web Services standards to provide interoperable interfaces among system components, and wrappers for legacy systems.

3. *Data:* SINERGY architecture shall be based on SOA, which prescribes an ESB layer that provides seamless transformation of data shared among services.

Compliance with Standards: SINERGY shall comply with the required technology, legal, and institutional standards.

Information Availability: SINERGY shall provide real-time access to information in a pervasive manner through two complimentary mechanisms: 1) push mechanisms such as brief SMS alerts pushed to campus constituents wherever they are, and 2) pull mechanisms such as website updates where the information is updated regularly but constituents need to go to the website and read (pull) it.

Infrastructure Availability: SINERGY infrastructure shall be available 24 hours a day, 7 days a week. This quality attribute shall be achieved in the following manner:

- SINERGY shall use dedicated physical resources hosted on site.
- SINERGY shall use redundant physical resources hosted in national and international off-site locations to ensure the availability of capabilities when campus infrastructure is compromised.
- SINERGY command center critical facilities shall be duplicated on a mobile vehicle that can be moved closer or away from an emergency scene as needed.

Fault Tolerance: SINERGY shall be fault tolerant and resilient to failure through the adoption of the architecture and design concepts that promote minimal dependencies among system hardware components and among system software components, graceful degradation of performance during failures, and robust response to errors, attacks, failures, and exceptions.

Usability: SINERGY shall use usability and human factors best practices of simplicity to achieve the needs of the stakeholders by adopting popular technologies among students as well as traditional technologies used by faculty, staff, and emergency personnel. In addition, user-facing system interfaces shall make use of simple and efficient design choices to reduce effect on system performance and scalability especially during emergencies.

Reliability is the extent to which SINERGY provides its capabilities without failure under normal conditions and during emergencies within the specified performance parameters.

Maintainability is the extent to which SINERGY facilitates changes to its components. Four types of maintenance exist. Adaptive maintainability is concerned with adaptations required as SINERGY external environment evolves. Corrective maintainability is concerned with fixing bugs and making corrections to SINERGY components. Perfective maintainability is concerned with enhancements requested because of changing stakeholder requirements. Preventive maintainability is concerned with preventing potential problems for reengineering.

Performance is the extent to which SINERGY executes its tasks in an efficient manner to support the decision-making process of the user.

Scalability is the extent to which SINERGY maintains its functional correctness as its workload is increased within some pre-defined limits.

Adaptability is the extent to which the architecture enables the system to be easily modified to satisfy changing requirements.

Dependability is the degree to which the architecture enables the system to deliver services when requested, as specified, and without catastrophic failure.

Extensibility is the extent to which the architecture enables the system to be capable of growing by including more and a greater diversity of subsystems, and to facilitate the extension of its capabilities by modifying current features or adding new features.

Modifiability is the extent to which the architecture enables the system to be easily changed.

Openness is the extent to which the architecture enables the system to execute its work in a speedy, efficient, and productive manner.

Testability is the extent to which the architecture enables the system to facilitate the creation of test criteria and conduct tests to determine whether those criteria have been met.

6.4.5. Relative Criticality Weighting

The parent indicator in this evaluation is “SINERGY Architecture Quality”. We decompose this indicator into several indicators, which represent SINERGY quality attributes. Each child indicator influences the

parent indicator in different amounts, which are called weights. A weight is a fraction between 0 and 1. The total of all weights of child indicators must sum to 1. The weighting process is called relative criticality weighting, because each child indicator is weighted with respect to its siblings.

We use the Analytic Hierarchy Process (AHP) for relative criticality weighting of SINERGY architecture quality indicators. AHP is a mathematical technique used for multi-criteria decision making [Saaty 1994]. Table 6-6 shows the criticality weighting of each quality attribute using AHP. Table 6-7 and Table 6-8 show the intermediate calculation of pairwise comparison among key quality attributes and expected quality attributes, respectively.

Table 6-6. Results of relative criticality weighting of indicators using AHP

		Weight	Quality Attribute	Relative Weight	Overall Weight
SINERGY Architecture Quality	**Key Quality Attributes**	**0.7000**	Security	0.3313	0.2319
			Response Flexibility	0.2307	0.1615
			Interoperability	0.1572	0.1100
			Information Availability	0.1059	0.0741
			Compliance with Standards	0.0709	0.0496
			Infrastructure Availability	0.0477	0.0334
			Fault Tolerance	0.0327	0.0229
			Usability	0.0236	0.0165
			Consistency Ratio: 0.0295	**= 1.0000**	**= 0.7000**
	Expected Quality Attributes	**0.3000**	Performance	0.2745	0.0824
			Dependability	0.2055	0.0617
			Reliability	0.1513	0.0454
			Scalability	0.1099	0.0330
			Maintainability	0.0756	0.0227
			Adaptability	0.0569	0.0171
			Modifiability	0.0445	0.0134
			Extensibility	0.0339	0.0102
			Openness	0.0204	0.0061
			Testability	0.0153	0.0046
			Survivability	0.0123	0.0037
			Consistency Ratio: 0.0735	**= 1.0000**	**= 0.3000**

Table 6-7. Intermediate calculation of pairwise comparison among key quality attributes

Key Quality Attributes	Security	Response Flexibility	Inter-operability	Compliance with Standards	Information Availability	Infrastructure Availability	Fault Tolerance	Usability
Security	1	**2**	**3**	**5**	**4**	**6**	**7**	**8**
Response Flexibility	1/2	1	**2**	**4**	**3**	**5**	**6**	**7**
Interoperability	1/3	1/2	1	**3**	**2**	**4**	**5**	**6**
Compliance with Standards	1/5	1/4	1/3	1	1/2	**2**	**3**	**4**
Information Availability	1/4	1/3	1/2	**2**	1	**3**	**4**	**5**
Infrastructure Availability	1/6	1/5	1/4	1/2	1/3	1	**2**	**3**
Fault Tolerance	1/7	1/6	1/5	1/3	1/4	1/2	1	**2**
Usability	1/8	1/7	1/6	1/4	1/5	1/3	1/2	1

Table 6-8. Intermediate calculation of pairwise comparison among expected quality attributes

Expected Quality Attributes	Reliability	Maintainability	Performance	Scalability	Adaptability	Dependability	Extensibility	Modifiability	Openness	Testability	Survivability
Reliability	1	3	1/3	2	4	1/2	5	6	7	8	9
Maintainability	1/3	1	1/5	1/2	2	1/4	3	3	5	6	7
Performance	3	5	1	4	6	2	7	8	9	9	9
Scalability	1/2	2	1/4	1	3	1/3	4	5	6	7	8
Adaptability	1/4	1/2	1/6	1/3	1	1/5	2	3	4	5	6
Dependability	2	4	1/2	3	5	1	6	7	8	9	9
Extensibility	1/5	1/3	1/7	1/4	1/2	1/6	1	1/6	3	4	5
Modifiability	1/6	1/3	1/8	1/5	1/3	1/7	6	1	2	3	4
Openness	1/7	1/5	1/9	1/6	1/4	1/8	1/3	1/2	1	2	3
Testability	1/8	1/6	1/9	1/7	1/5	1/9	1/4	1/3	1/2	1	2
Survivability	1/9	1/7	1/9	1/8	1/6	1/9	1/5	1/4	1/3	1/2	1

6.4.6. Score Assignment and Rationale

Assigning a score to each indicator is a quality judgment. Therefore, we use a nominal score set that helps answer the question of "*how well does the architecture enable the system to achieve the desired quality attributes under the IUs of the system?"* Table 6-9 shows the nominal score set we use to judge the quality of the architecture in terms of its quality attributes.

Table 6-9. Nominal score set for judging the architecture based on its quality attributes

Nominal Score	Numerical Score	Description
Excellent	[81-100]	Architecture comprehensively supports achieving quality attribute and resolves all related tradeoffs among attributes
Good	[61-80]	Architecture explicitly supports achieving quality attribute and resolves major related tradeoffs
Average	[41-60]	Architecture supports achieving quality attribute but major tradeoffs remain to be resolved
Marginal	[21-40]	Architecture implicitly supports achieving quality attribute but directly conflicts with one or more quality attributes
Poor	[0-20]	Architecture is not suitable for achieving quality attribute

In each section below, we list a quality attribute, then we show the nominal score we give to each quality attributes.

Security: Security is addressed at the services management level through the creation of an enterprise-wide security services that are separate from the business logic of service consumers and providers. Creating a layer for security services is complemented by adding local security handlers that communicate with central security services to avoid performance issues.

✔				
Excellent	Good	Average	Marginal	Poor

Response Flexibility: The architecture incorporates a comprehensive set of notification technologies that accommodate the diversity of campus constituents. Notification technologies include traditional ones such as landline telephone as well as cutting-edge technologies such as mobile app alerts.

✔				
Excellent	Good	Average	Marginal	Poor

Interoperability: The architecture is compliant with the ICS, which advocates a common terminology defining organizational functions, facilities, resource description, and position titles. In addition, the architecture is based on SOA; therefore, it uses Web Services standards to provide interoperable interfaces among system components. However, legacy systems issues remain to be resolved.

	✔			
Excellent	Good	Average	Marginal	Poor

Compliance with Standards: The architecture complies with the required technology, legal, and institutional standards specified in the StdV-2.

✔				
Excellent	Good	Average	Marginal	Poor

Information Availability: SINERGY architecture supports two mechanisms to make information available to campus constituents. The push mechanism is represented by technologies such as SMS and mobile app alerts. The pull mechanism is represented by technologies such as website and social networks updates.

	✔			
Excellent	Good	Average	Marginal	Poor

Infrastructure Availability: The architecture addresses infrastructure availability issue by proposing a mobile command center on a vehicle that can be moved closer or away from an emergency scene as needed. This approach is a novel as it enables campus to connect to its resources through satellite communication in case campus networks are compromised.

		✔		
Excellent	Good	Average	Marginal	Poor

Fault Tolerance: SOA promotes minimal dependencies among system hardware components and among system software components. The architecture supports graceful degradation of performance during failures, and robust response to errors, attacks, failures, and exceptions.

		✔		
Excellent	Good	Average	Marginal	Poor

Usability: The architecture proposes system interfaces with simple and efficient design choices to reduce effect on system performance and scalability especially during emergencies.

	✔			
Excellent	Good	Average	Marginal	Poor

Reliability: How well does the architecture enable the system provide capabilities without failure under normal conditions and during emergencies within the specified performance parameters?

		✔		
Excellent	Good	Average	Marginal	Poor

Maintainability: How well does the architecture enable the system to facilitate changes to its components?

✔				
Excellent	Good	Average	Marginal	Poor

Performance: How well does the architecture enable the system to execute tasks in an efficient manner to support the decision-making process of the user?

		✔		
Excellent	Good	Average	Marginal	Poor

Scalability: How well does the architecture enable the system to maintain its functional correctness as its workload is increased within some pre-defined limits?

	✔			
Excellent	Good	Average	Marginal	Poor

Adaptability: How well does the architecture enable the system to be easily modified to satisfy changing requirements?

✔				
Excellent	Good	Average	Marginal	Poor

Dependability: How well does the architecture enable the system to deliver services when requested, as specified, and without catastrophic failure?

	✔			
Excellent	Good	Average	Marginal	Poor

Extensibility: How well does the architecture enable the system to be capable of growing by including more and a greater diversity of subsystems, and to facilitate the extension of its capabilities by modifying current features or adding new features?

	✔			
Excellent	Good	Average	Marginal	Poor

Modifiability: How well does the architecture enable the system to be easily changed?

	✔			
Excellent	Good	Average	Marginal	Poor

Openness: How well does the architecture enable the system to execute its work in a speedy, efficient, and productive manner?

✔				
Excellent	Good	Average	Marginal	Poor

Testability: How well does the architecture enable the system to facilitate the creation of test criteria and conduct tests to determine whether those criteria have been met?

		✔		
Excellent	Good	Average	Marginal	Poor

Survivability: How well does the architecture enable the system to satisfy and continue to satisfy specified critical requirements (e.g. security, reliability, real-time responsiveness, and accuracy) under adverse conditions?

		✔		
Excellent	Good	Average	Marginal	Poor

6.4.7. Overall Evaluation Results

The overall results are shows in Table 6-10. We list each quality attribute with the following information:

- Overall weight: The weight of the indicator in influencing the parent indicator "SINERGY Architecture Quality"
- Interval score: Interval score based on the nominal score given to each indicator
- Mid-value: We consider the middle value of the interval score as the corresponding crisp score for the indicator

- Weighed score: The score of the indicator relative to all other indicators
- Total score: The crisp score for the parent indicator

Table 6-10. Weighted scores for each quality attribute

Quality Attribute	Overall Weight	Interval Score	Mid-Value	Weighted Score
Security	0.2319	[81-100]	90	20.87
Response Flexibility	0.1615	[81-100]	90	14.54
Interoperability	0.1100	[61-80]	70	7.70
Information Availability	0.0741	[61-80]	70	5.19
Compliance with Standards	0.0496	[81-100]	90	4.46
Infrastructure Availability	0.0334	[41-60]	50	1.67
Fault Tolerance	0.0229	[41-60]	50	1.15
Usability	0.0165	[61-80]	70	1.16
Performance	0.0824	[41-60]	50	4.12
Dependability	0.0617	[61-80]	70	4.32
Reliability	0.0454	[41-60]	50	2.27
Scalability	0.0330	[61-80]	70	2.31
Maintainability	0.0227	[81-100]	90	2.04
Adaptability	0.0171	[81-100]	90	1.54
Modifiability	0.0134	[61-80]	70	0.94
Extensibility	0.0102	[61-80]	70	0.71
Openness	0.0061	[81-100]	90	0.55
Testability	0.0046	[41-60]	50	0.23
Survivability	0.0037	[41-60]	50	0.19
				Total Score: 75.94

Based on the overall score, we deduce the following:

- Interval score: [61-80]
- Nominal score: Good

Therefore, SINERGY architecture does a *Good* job enabling the system to exhibit the desired quality attributes.

CHAPTER 7: A HIERARCHY OF INDICATORS FOR SOA SECURITY

Security evaluation of SOA-based network-centric systems of systems is a complex task due the nature of business practices today, which emphasize dynamic sharing of information and services among independent partners. To conduct security evaluation effectively in these complex environments, there is a need for a structured approach. In this chapter, we provide such an approach. We propose a hierarchy of security indicators that address the security concerns associated with SOA-based solutions. To show the usefulness of the proposed hierarchy, we incorporate the hierarchy of indicators into the EE methodology to provide a structured approach for assessing the security of SOA-based solutions.

7.1. INTRODUCTION

The nature of business practices today emphasizes the dynamic sharing of information and services among independent partners. As a result, the line between internal and external organization networks and services has been blurred [Rosen *et al*. 2008] making it complex to assess the security quality of SOA environments. We propose a solution for assessing the security of an SOA-based network-centric system of systems in a structured way. The first component of this solution is the development of a hierarchy of indicators that reflect the concerns associated with SOA-based solutions. The development of these indicators is based on surveying the published SOA literature to identify the security concerns that are specific to SOA. Based on these concerns, we derive security indicators that can be decomposed further into other indicators until there is a leaf indicator that can be evaluated directly. The assessment of all leaf indicators allows for the formulation of a holistic view about the security quality of the system being assessed. The EE methodology is designed to support each stage of this process. Our research focus is on developing the hierarchy of indicators and evolving the EE methodology to be applied in the assessment of the security of SOA-based solutions.

The remainder of this chapter will detail our approach to conducting this part of the research and the results we have thus far. It is organized as follows. Section 2 talks about our research approach to addressing this problem and lays out the details of our preliminary results. Section 3 discusses the remaining work and our validation plan of our results.

7.2. APPROACH AND RESULTS SUMMARY

The approach to solve this problem centers on the idea of decomposing security quality into a manageable set of security indicators that are specific to SOA and that can be assessed directly. The results of the

assessment of each leaf indicator are then aggregated to provide a holistic view of the security quality in the system being evaluated. The end result is a hierarchy of indicators that helps security experts and architects conduct security assessment in a structured way.

Figure 7-1 shows the steps that comprise our solution approach. First, we surveyed the literature and identified the security concerns that are specific to Web Services. These concerns were grouped based on the four elements of SOA [Glass 2008] and served as high-level indicators in the hierarchy. The next step was to decompose these indicators further into low-level indicators until we reached a leaf indicator that can be evaluated directly. The result is a hierarchy (an acyclic graph) of security indicators that when assessed can provide a holistic picture of the security quality of the system.

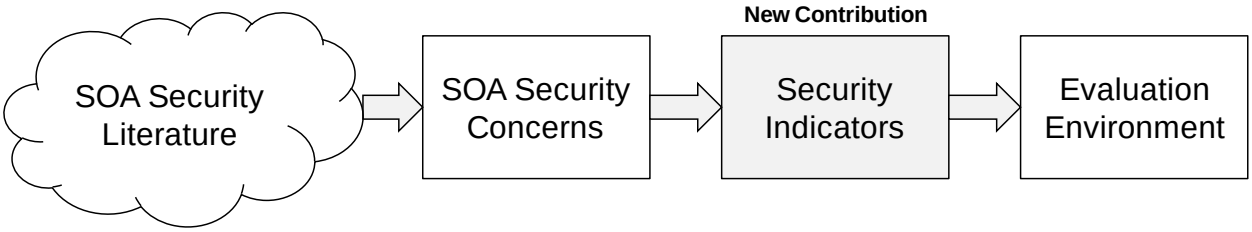


Figure 7-1. Steps to eliciting indicators of SOA security

Based on the SOA conceptual layers [Glass 2008], we describe our hierarchy of indicators based on the SOA essential components.

7.2.1. Web Services Security Indicators

The security indicators that need to be addressed at this layer are shown in Figure 7-2.

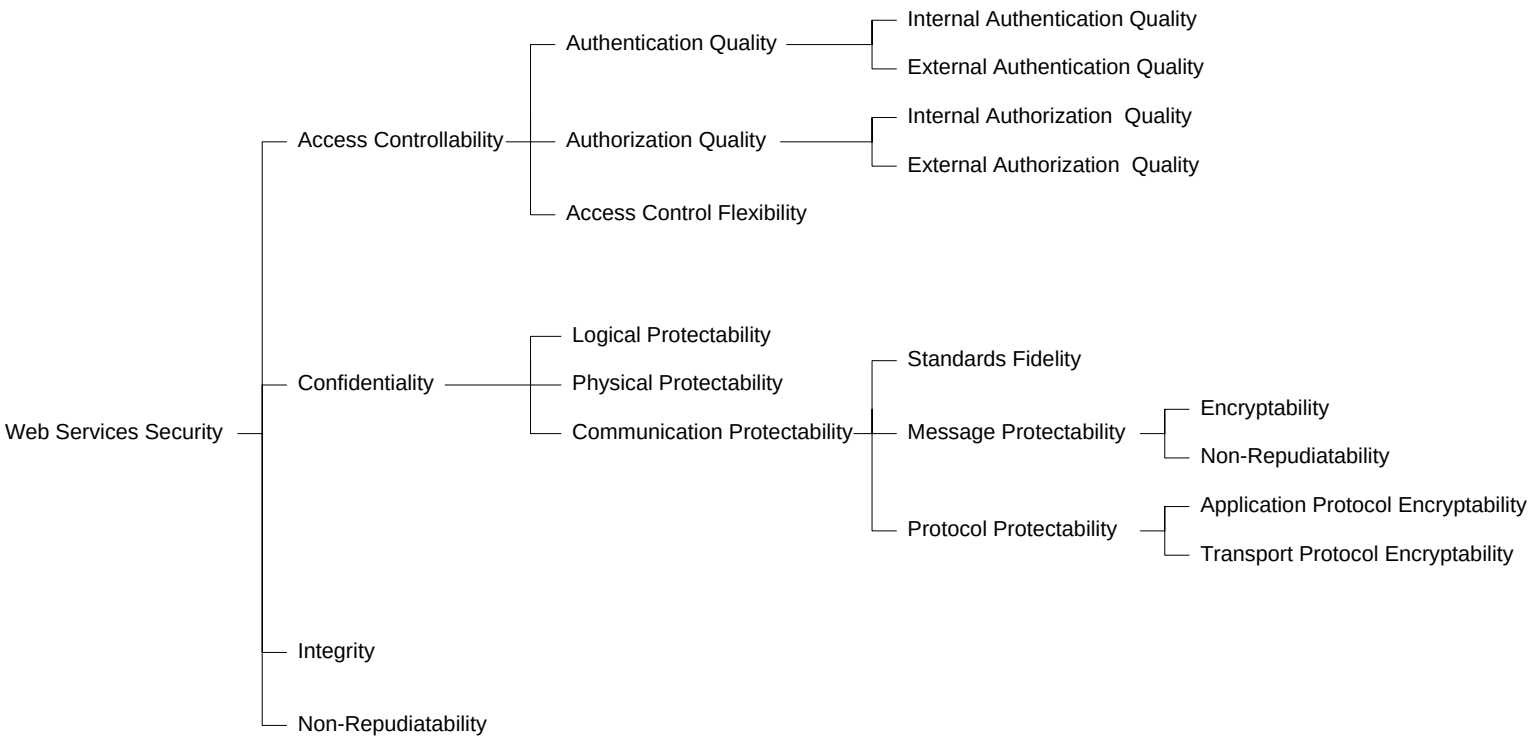


Figure 7-2. Web Services security indicator hierarchy

7.2.2. ESB Security Indicators

The security indicators that need to be addressed at this layer are shown in Figure 7-3.

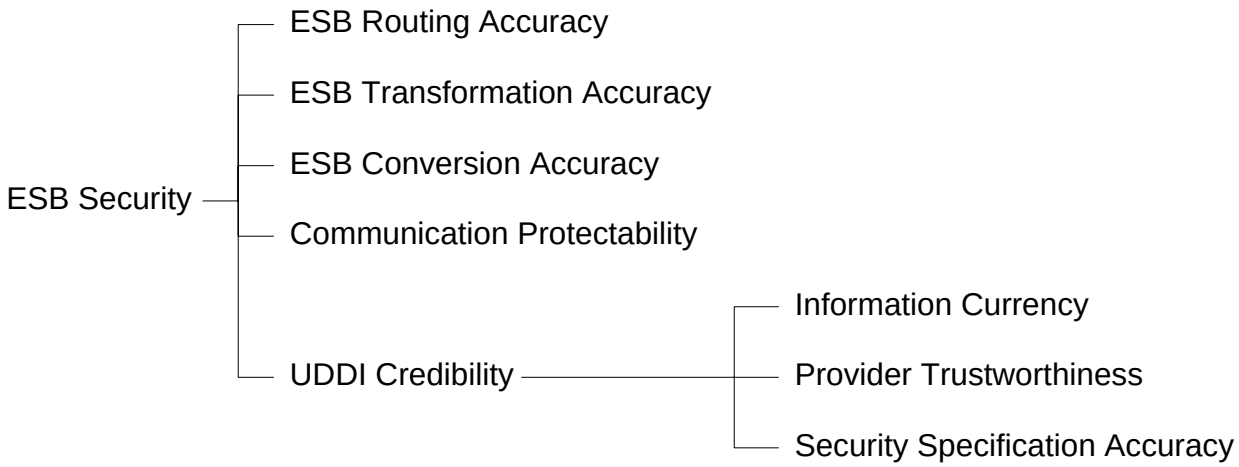


Figure 7-3. ESB security indicator hierarchy

7.2.3. Orchestration Security Indicators

The security indicators that need to be addressed at this layer are shown in Figure 7-4.

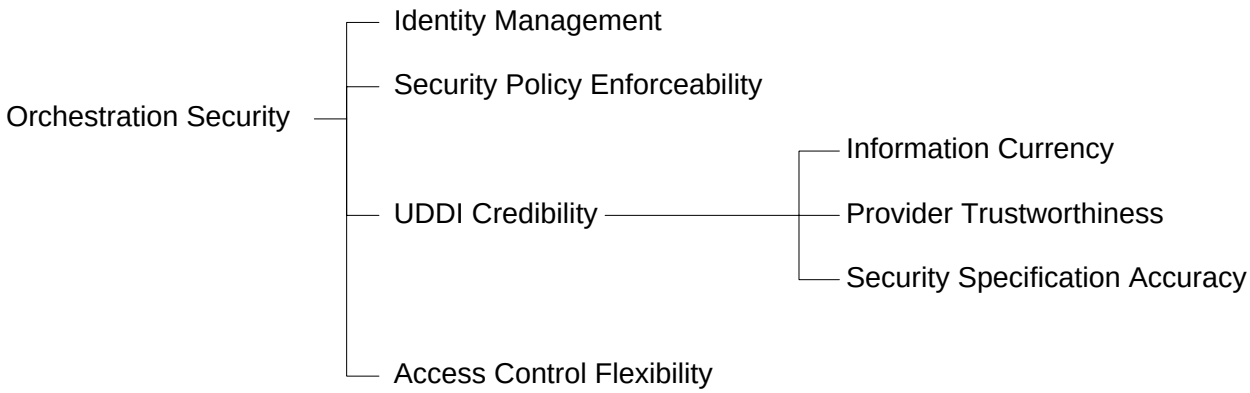


Figure 7-4. Orchestration security indicator hierarchy

7.2.4. Services Management Security Indicators

The security indicators that need to be addressed at this layer are shown in Figure 7-5.

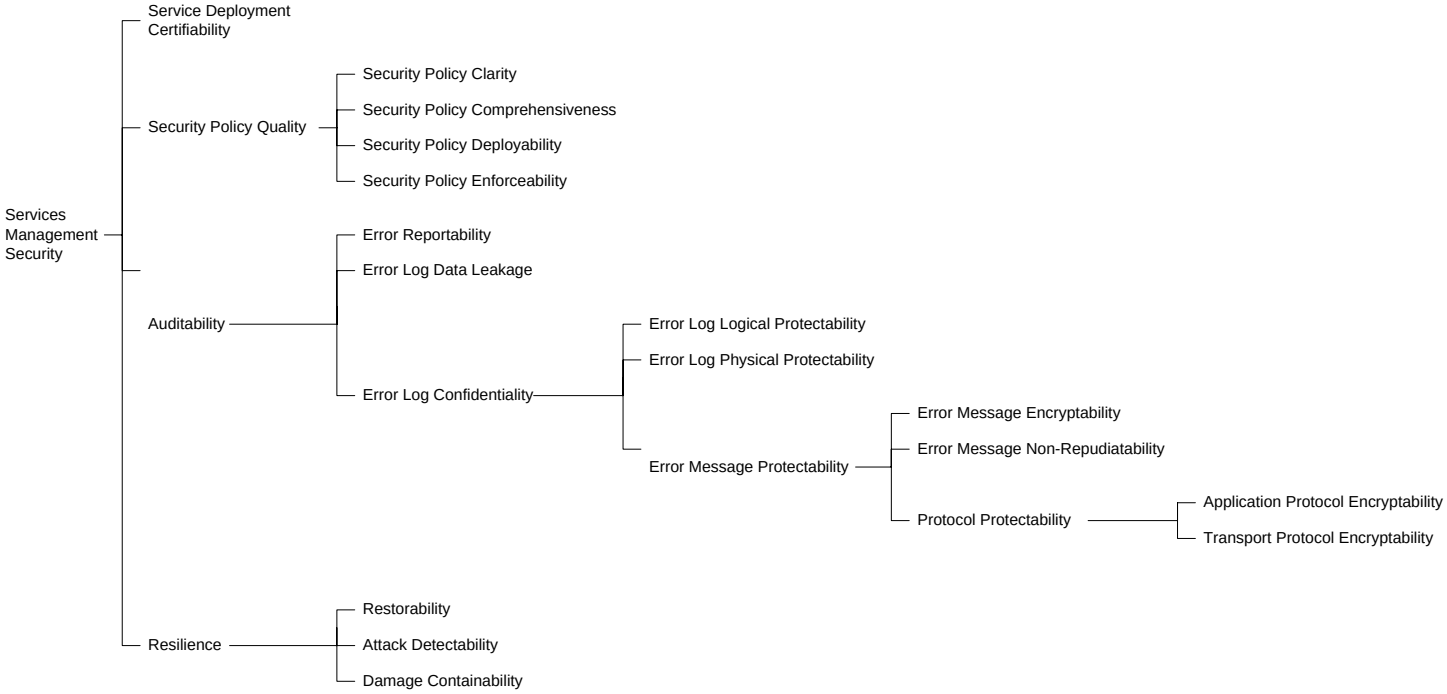


Figure 7-5. Services management security indicator hierarchy

7.3. DESCRIPTION OF THE INDICATORS

Security is a critical quality characteristic of SOA-based network-centric systems of systems. Security evaluation of such systems is a complex task due the nature of business practices today, which emphasize dynamic sharing of information and services among independent partners. To conduct security evaluation effectively in these complex environments, we decompose it into a hierarchy of security indicators that address various aspects of security concerns in SOA solutions.

7.3.1. Web Services Security

The basic profile (or best practices) for Web Services includes XML, HTTP, WSDL, and SOAP. It describes the standards, and their version, that can be used to develop and deploy Web Services.

Web Services Security is the extent to which Web Services and the messages exchanged among them are secure in terms of access control to resources, confidentiality of the information exchanged, non-repudiation, and integrity of messages sent.

7.3.1.1. Access Controllability

Access Controllability is the extent to which the system is capable of governing access to its resources (exposed as Web Services) by internal and external subjects. A subject can be a user, a web service, a computer, or an application.

7.3.1.1.1. Authentication Quality

Authentication Quality is the extent to which the system (and its Web Services) validates the identity of a subject before granting access to resources. Authentication Quality is concerned with the quality of the process used to verify that a subject is who it claims it is. It is the first step of Access Control that is typically followed by Authorization.

7.3.1.1.1.1. Internal Authentication Quality

Internal Authentication Quality is the extent to which the system (and its Web Services) validates the identity of an internal subject before granting access to resources. Internal subjects are those that belong to the same security domain of the enterprise. For instance, within a given department certain services (resources) can be used locally as well as exposed to the rest of the enterprise. Internal security measures to authenticate internal subjects need not be as rigorous as those required from external subjects.

7.3.1.1.1.2. External Authentication Quality

External Authentication Quality is the extent to which the system (and its Web Services) validates the identity of an external subject before granting access to resources. External subjects are those that belong to a different security domain from that of the enterprise providing the resource. Proper mechanisms of federated access control must be put in place in order for services to be provided to external subjects.

7.3.1.1.2. Authorization Quality

Authorization Quality is the extent to which the system determines the privileges and permissions a subject has before granting access to resources. Logically, authorization is preceded by authentication.

7.3.1.1.2.1. Internal Authorization Quality

Internal Authorization Quality is the extent to which the system determines the privileges and permissions an internal subject has before granting access to resources. This indicator is concerned with the quality of the process the system uses to check the permissions of authenticated internal subjects before allowing access to a system's resource.

7.3.1.1.2.2. External Authorization Quality

External Authorization Quality is the extent to which the system determines the privileges and permissions an external subject has before granting access to resources. This indicator is concerned with the quality of the process the system uses to check the permissions of authenticated external subjects before allowing access to a system's resource.

7.3.1.1.3. Access Control Flexibility

Access Control Flexibility is the extent to which the system supports architectural flexibility for the authorization of subjects. This flexibility is achieved by dividing the duties of access control logically into Policy Decision Points (PDP) and Policy Enforcement Points (PEP). This flexibility influences security in the sense that it separates the concern of providing a particular service from the task of managing security policy and credentials. For instance, a service can be deployed on a web server. At that point, we can establish a PEP. A separate server, PDP, can host the security service that will make the decision of whether access to resources can be granted. Upon the receipt of a service request, the PEP service queries PDP whether access can be granted, gets a response, and acts accordingly on it. This architectural configuration allows for the service policies and measures to change without affecting the services provided.

7.3.1.2. Confidentiality

Confidentiality is the extent to which the information transmitted between authorized entities is protected against eavesdropping. Confidentiality is a security goal for protecting sensitive information and it is done through encryption of the information carried inside the messages transmitted between services.

7.3.1.2.1. Logical Protectability

Logical Protectability is the extent to which the architecting of the data and the databases (data stores) is properly done to ensure the confidentiality of information. This indicator assesses how confidential data is stored, how it is accessed, and what mechanisms are used to protect intentional and/or unintentional exposure of confidential data to unauthenticated/unauthorized entities.

7.3.1.2.2. Physical Protectability

Physical Protectability is the extent to which the hardware of the data stores (databases) is protected and secured against stealing, damage caused by the elements, and from outside security attacks.

7.3.1.2.3. Communication Protectability

Communication Protectability is the extent to which the communication (or messaging) among services is protected against malicious intrusion. The communication among services can be assessed at two levels: Message and Protocol level.

7.3.1.2.3.1. Standards Fidelity

Standards Fidelity is the extent to which the system properly employs SOA security standards to achieve end-to-end security and message-level security. End-to-end security is concerned with the security provided by the transport layer protocols such as SSL and TSL. Message-level security is concerned with the security mechanisms deployed at the message level such as Digital Signatures and XML Encryption.

7.3.1.2.3.2. Message Protectability

Message Protectability is the extent to which the messages exchanged between services are protected against malicious intrusion. At the message level, we assess the confidentiality and non-repudiation of the message.

7.3.1.2.3.2.1. Encryptability

Encryptability is the extent to which the information transmitted between authorized entities is protected against eavesdropping. Encryptability is a security goal for protecting sensitive information and it is done through encryption of the information carried inside the messages transmitted between services.

7.3.1.2.3.2.2. Non-Repudiatability

Non-Repudiatability is the extent to which a subject can prove that it actually signed the message sent requesting something from/responding to a resource. This indicator assesses the extent to which a message can be proven to have been sent by a source. For instance, using digital signature cryptography, the identity of the signer can be tied to the contents of the data being signed.

7.3.1.2.3.3. Protocol Protectability

Protocol Protectability is the extent to which the network protocol(s) used to transmit the messages between subjects and resources are secure protocols.

7.3.1.2.3.3.1. Application Protocol Encryptability

Application Protocol Encryptability is the extent to which the application protocol(s) used to transmit the messages between services is encrypted. For instance, at the application level, this indicator will assess whether the transmission is done through HyperText Transfer Protocol Secure (HTTPS) instead of non-secure HTTP.

7.3.1.2.3.3.2. Transport Protocol Encryptability

Transport Protocol Encryptability is the extent to which the transport protocol(s) used to transmit the messages between services are encrypted. For instance, at the transport level, this indicator will assess whether the transmission is done through SSL (Secure Sockets Layer) or TLS (Transport Secure Layer).

7.3.1.3. Integrity

Integrity is the degree to which the system protects the data from being tampered with in transit or at rest. Integrity deals with validating that the data has not been altered from its original state when sent by a subject.

7.3.1.4. Non-Repudiatability

Non-Repudiatability is the extent to which a subject can prove that it actually signed the message sent requesting something from/responding to a resource. This indicator assesses the extent to which a message can be proven to have been sent by a source. For instance, using digital signature cryptography, the identity of the signer can be tied to the contents of the data being signed.

7.3.2. ESB Security

The ESB is a combination of software and hardware middleware that offers a standard way for services to interact and communicate with one another.

ESB Security is the extent to which the ESB securely abstracts its resources while facilitating interaction and communication among Web Services. For instance, consider the case when the ESB needs to properly handle different mechanisms of securing Web Services. When service A invokes a credit card processing service using HTTPS, the communication between the ESB and the credit card service has to be over HTTPS or a similar encrypted protocol.

The ESB must be able to, at least, provide the following capabilities: Connect, Transform, and Route.

- Transformation: An ESB should provide the capability to connect to Web Services through either WSDL or adapters in case of legacy systems.
- Conversion: An ESB should perform data transformation to and from data types that are required by the Web Services to which it connects.
- Routing: An ESB should route messages from one service to the others using appropriate communication protocols for each service.
- UDDI is a platform-independent, XML-based registry for service providers to list their services on the Internet. UDDI is an open industry initiative, sponsored by OASIS, enabling businesses to publish service listings, discover each other, and define how the services or software applications interact over the Internet [OASIS 2004].

7.3.2.1. ESB Routing Accuracy

ESB Routing Accuracy is the extent to which the routing between different protocols preserves the secure transmission of the messages exchanged between the services. This indicator assesses situations similar to the following: Service A sends a request-for-service message to the ESB using SOAP over HTTPs with XML-Encrypted message. The service provider, however, communicates with the ESB using

SOAP/HTTP without any security mechanism. Such configuration will expose the information sent by Service A while it is transmitted from the ESB to the service provider.

7.3.2.2. ESB Transformation Accuracy

ESB Transformation Accuracy is the extent to which the ESB provides the capability to connect to Web Services through either WSDL or adapters in case of legacy systems.

7.3.2.3. ESB Conversion Accuracy

ESB Conversion Accuracy is the extent to which the ESB performs accurate data transformation to and from data types that are required by the Web Services to which it connects without exposing this information to unauthorized users or tempering the content of this information. This indicator impacts security in that it can cause the exposure of confidential data if not done properly. In addition, poorly transformed data can cause unexpected behavior that may result in unexpected security vulnerabilities in the system.

7.3.2.4. Communication Protectability

Communication Protectability is the extent to which the communication (or messaging) among services is protected against malicious intrusion. The communication among services can be assessed at two levels: Message and Protocol level.

7.3.2.5. UDDI Credibility

UDDI Credibility is the extent to which the information published about Web Services in the UDDI is accurate, current, and trustworthy. This indicator also assesses how often this information is updated and how it is verified.

7.3.2.5.1. Information Currency

Information Currency is the extent to which the information about the services published in the UDDI is updated. Security is greatly impacted if the UDDI is not up to date with the latest versions of the services published in it. From an orchestration security perspective, mechanisms have to be put in place to ensure that the services used to provide a particular capability are current, trustworthy, and specified correctly.

7.3.2.5.2. Provider Trustworthiness

Provider Trustworthiness is the extent to which the service providers that publish their service descriptions to the UDDI are trusted. This indicator assesses the mechanisms used to guide who and what can be published in the UDDI to prevent malicious access to it.

7.3.2.5.3. Security Specification Accuracy

Security Specification Accuracy is the extent to which WSDL descriptions in the UDDI accurately specify the security mechanisms that are used to communicate securely with the service provided. Assessing this indicator prevents malicious access to and manipulation of services by hackers.

7.3.3. Orchestration Security

In SOA, orchestration is concerned with the issue of assembling services together dynamically to provide the desired capabilities. The challenge is that it has to be done at runtime, posing the question of how we can know all potential service configurations at the architecting phase. We achieve orchestration of services by creating BPEL workflows that map to business processes, which may extend across one or more organizations. Service orchestration follows these workflows at runtime.

Orchestration Security is the extent to which the dynamic assembling and mashing of services preserves the security quality of the system. Security becomes a bigger challenge because it has to be reasoned about when these BPEL workflows are created at the architecting phase. Therefore, this indicator is concerned with assessing how security measures are preserved as these services are configured at runtime.

7.3.3.1. Identity Manageability

Identity Manageability is the extent to which the identity of the subject requesting a resource is securely propagated in multi-hop communications. For instance, most often than not, there are intermediaries between the subject and the resource in any non-trivial SOA implementation. This indicator assesses how the credentials of the authenticated and authorized subject are propagated through the intermediaries.

7.3.3.2. Security Policy Enforceability

Security Policy Enforceability is the extent to which the established security policy is enforced. From an orchestration perspective, this indicator ensures that the orchestration workflows do not violate the expressed security policies. For instance, the security policy states that access to Service A cannot be

granted except to subjects that have Admin privileges to the system. When a workflow is being created to include Service A in a configuration that includes several services, no requests to Service A should be made except from subjects who have the Admin privileges. This ensures that at the orchestration layer, security policy is enforced through the architecting activities of the system.

From a management perspective, we assess enforceability based on the different approaches used to enforce security policy including a centralized approach, a decentralized (or local) approach, and a hybrid approach. Each is suitable for a particular context.

7.3.3.3. UDDI Credibility

UDDI Credibility is the extent to which the information published about Web Services in the UDDI is accurate, current, and trustworthy. This indicator also assesses how often this information is updated and how it is verified.

7.3.3.4. Access Control Flexibility

Access Control Flexibility is the extent to which the system supports architectural flexibility for the authorization of subjects. This flexibility is achieved by dividing the duties of access control logically into PDPs and PEPs. This flexibility influences security in the sense that it separates the concern of providing a particular service from the task of managing security policy and credentials. For instance, a service can be deployed on a web server. At that point, we can establish a PEP. A separate server, PDP, can host the security service that will make the decision of whether access to resources can be granted. Upon the receipt of a service request, the PEP service queries PDP whether access can be granted, gets a response, and acts accordingly on it. This architectural configuration allows for the service policies and measures to change without affecting the services provided.

7.3.4. Services Management Security

Due the complexity of the network-centric systems of systems we build today, any non-trivial SOA-based network-centric system of systems will be composed of hundreds, if not thousands, of services. At this scale, management of these services as enterprise assets becomes critical. Proper management of these services will yield higher return on investment as more and more services are reused in different configurations to satisfy more and more business needs. Bad management will result in redundancy, configuration problems, versioning issues, and security vulnerabilities.

Services Management Security is the extent to which the Web Services are managed in such a way that security vulnerabilities are not introduced as services are used in new systems and by new consumers.

7.3.4.1. Service Deployment Certifiability

Service Deployment Certifiability assesses the process by which a service is approved (or certified) for deployment. Many security attacks can be launched from within an organization. In addition, vulnerabilities could be intentionally coded into services by insiders. From a management perspective, assuring service consumers that there is a process of certifiability followed before a service is published is a crucial indicator to ensure the security of the system.

7.3.4.2. Security Policy Quality

Security Policy Quality is the extent to which the established security policy is clear, comprehensive, deployable, and enforceable. Coming up with the appropriate security policy of the system is critical and difficult to do. This indicator looks at various key indicators to ensure that the policy provides the required security quality that the system should exhibit.

7.3.4.2.1. Security Policy Clarity

Security Policy Clarity is the extent to which the established security policy can be uniformly understood by all security stakeholders including architects, designers, and developers.

7.3.4.2.2. Security Policy Comprehensiveness

Security Policy Comprehensiveness is the extent to which the established security policy addresses all the security requirements including access control requirements, data confidentiality and integrity requirements, and subject's identity management across enterprise boundaries.

7.3.4.2.3. Security Policy Deployability

Security Policy Deployability is the extent to which the established security policy is deployable. The security standards used in the establishment of the security policy must be supported by the platforms upon which the services will be deployed.

7.3.4.2.4. Security Policy Enforceability

Security Policy Enforceability is the extent to which the established security policy is enforced. From an orchestration perspective, this indicator ensures that the orchestration workflows do not violate the

expressed security policies. For instance, the security policy states that access to Service A cannot be granted except to subjects that have Admin privileges to the system. When a workflow is being created to include Service A in a configuration that includes several services, no requests to Service A should be made except from subjects who have the Admin privileges. This ensures that at the orchestration layer, security policy is enforced through the architecting activities of the system.

From a management perspective, we assess enforceability based on the different approaches used to enforce security policy including a centralized approach, a decentralized (or local) approach, and a hybrid approach. Each is suitable for a particular context.

7.3.4.3. Auditability

Auditability is the extent to which the system is able to keep an on-going log of who accessed what and when. This indicator is crucial to ongoing security assessment and maintenance.

7.3.4.3.1. Error Reportability

Error Reportability is the degree to which the system reports accurately and securely information related to system failures. A vulnerability of the system from this perspective could allow sensitive information to be leaked into the error logs, which might not be stored or transmitter securely. In addition, improper failure reporting could lead to not recognizing an attack when it is happening or after it has happened.

7.3.4.3.2. Error Log Data Leakage

Error Log Data Leakage is the extent to which the system leaks confidential information into the error logs during an error or a failure. This indicator assesses how errors are logged, what information is logged, and whether any confidential information is logged.

7.3.4.3.3. Error Log Confidentiality

Error Log Confidentiality is the extent to which error messages are transmitted securely and are protected from eavesdropping. This indicator is also concerned with the level of security with which the logs are stored.

7.3.4.3.3.1. Error Log Logical Protectability

Error Log Logical Protectability is the extent to which the error log(s) are properly designed to ensure the confidentiality of information contained in them. This indicator assesses whether confidential data that

must be stored in an error log file is stored securely, how it is accessed, and what mechanisms are used to protect intentional and/or unintentional exposure of this confidential data to unauthenticated/unauthorized entities.

7.3.4.3.3.2. Error Log Physical Protectability

Error Log Physical Protectability is the extent to which the hardware where the data log(s) are stored is protected and secured against stealing, damage caused by the elements, and from outside security attacks.

7.3.4.3.3.3. Error Message Protectability

7.3.4.3.3.3.1. Error Message Encryptability

Error Message Encryptability is the extent to which the information transmitted within the error messages is protected against eavesdropping. Encryptability is a security goal for protecting sensitive information and it is done through encryption of the information carried inside the messages transmitted between services.

7.3.4.3.3.3.2. Error Message Non-Repudiatability

Error Message Non-Repudiatability is the extent to which a subject can prove that it actually signed the error message sent. This indicator assesses the extent to which a message can be proven to have been sent by a source. For instance, using digital signature cryptography, the identity of the signer can be tied to the contents of the data being signed.

7.3.4.3.3.3.3. Protocol Protectability

Protocol Protectability is the extent to which the network protocol(s) used to transmit the messages between subjects and resources are secure protocols.

a. Application Protocol Encryptability

Application Protocol Encryptability is the extent to which the application protocol(s) used to transmit the messages between services is encrypted. For instance, at the application level, this indicator will assess whether the transmission is done through HTTPS instead of non-secure HTTP.

b. Transport Protocol Encryptability

Transport Protocol Encryptability is the extent to which the transport protocol(s) used to transmit the messages between services are encrypted. For instance, at the transport level, this indicator will assess whether the transmission is done through SSL (Secure Sockets Layer) or TLS (Transport Secure Layer).

7.3.4.4. Resilience

Resilience is the extent to which the system is able to resist attacks and recovers from an attack or a failure without exposing security information. At the management level, anticipating that attacks and failures will happen is part of the concerns to be addressed. This indicator assesses the system's response to potential failures and attacks and how that affects the security.

7.3.4.4.1. Restorability

Restorability is the extent to which the system can be restored to its normal secure state after an attack or failure.

7.3.4.4.2. Attack Detectability

Attack Detectability is the extent to which the system possesses mechanisms to detect attacks and intrusions against its security features. This indicator assesses the quality of software and hardware attack detection tools used, if any.

7.3.4.4.3. Damage Containability

Damage Containability is the extent to which the system is able to contain the damage of an attack by limiting the attack surface. For instance, if an intruder gets into the database, is the database designed in such a way that additional authentication and authentication measures are in place to limit the exposure of confidential data? Services and resources should be managed in such a way that the vulnerability found in one resource does not jeopardize the entire system.

7.4. CONCLUDING REMARKS

This hierarchy provides the right first step to begin to think about security as an architecture issue rather than an implementation issue only. Each of the indicators identified corresponds to a rich literature about how to address it. To effectively use this hierarchy, it should be part of a comprehensive assessment methodology, such as MSAAM, to show its usefulness.

CHAPTER 8: CONCLUDING REMARKS AND FUTURE WORK

8.1. CONCLUDING REMARKS

The outcome of the study we conducted shows a clear contrast between what is currently practiced and deployed and what is actually needed to fully prepare open campuses to man-made disasters. It provides a supporting case to the argument that current technology solutions address the problem from one or few perspectives, when it should be addressed comprehensively. The nature of the problem of balancing openness and safety on open campus environments is multi faceted; thus, its solution(s) should also be multi faceted.

We approach this complexity through capabilities specification and architecting to devise a systems of systems solution. The architecture specification described herein provides the foundation for the design, implementation, and acquisition of such a solution. It utilizes DoDAF models to describe the solution architecture from different viewpoints, which correspond with a diverse set of stakeholders. The capabilities specification, on the other hand, provides a design framework that could be generalized to any campus safety system that supports group decision making.

While the literature is not lacking architecture knowledge, the lack of a defined process for architecting and real architecture examples of how to apply the steps of such a process makes this research a significant contribution to the state of art and practice. Our research provides a case study for how to conduct an architecting effort of a complex system addressing a real-life problem from establishing architecturally-relevant requirements to evaluating the architecture.

Finally, as service orientation takes roots as *the* architectural model for network centricity, and as Web Services continue to be the established best practice for implementing SOA-based systems, security will present itself as one of the most critical challenges of the network-centric era. Therefore, it is no longer enough to address this challenge at the application or code levels only. Addressing security and privacy concerns in today's complex systems should be at the architecture level before major resources are invested in the development of these systems. This research identifies a list of indicators, organized into a hierarchy, to help security experts and architect assess the security of a system beginning at the architecture level all the way down to the application level.

8.2. SUMMARY OF CONTRIBUTIONS

This research had three related objectives to achieve. The first objective was to propose a description of the process of architecting from systems of systems perspective. The second objective was to apply this process in the development of a system architecture that provides SA, SC, and ERM capabilities on open campus environments. The third objective was to define a hierarchy of indicators to assess security in SOA-based network-centric systems of systems. These objectives were met as detailed in this dissertation, and resulted in the following contributions to the software/system engineering body of knowledge.

8.2.1. Architecting as a Defined Process Area

We have defined and described the Process of Architecting from the perspective of systems of systems. The definition of this process was presented in the form of a CMMI process area focusing on the specific goals and related specific practices of the process area. The process provides guidance to architects on how to approach conceiving, describing, and evaluating an architecture for a system of systems. On each phase of the process, we provide guidance on architectural approaches, documentation perspectives, and evaluation methodologies. This contribution is published in [Chigani and Balci 2011].

8.2.2. SINERGY Architecture

To address the challenges and issues that were raised in the aftermath of April 16 shootings at Virginia Tech, we have developed an architecture for campus SA and ERM system called SINERGY. The development, description, and the evaluation of this architecture served as an application example for the Process of Architecting that we have defined. In the following sections, we describe the different components of this contribution.

- **Problem Formulation:** Due to the lack of proper problem formulation, campus safety solutions that exist today lack coherence and efficacy. Only through considering institutions as a coherent enterprise with external partners can we address this issue properly. Through the study of the panel reviews analyzing the events of April 16 and other similar incidents, we established a comprehensive framework from which this issue can be addressed. This framework is built on three different components: SA, SC, and ERM. The scoping of the problem domain was discussed in an extended abstract and a presentation at SATURN 2010 [Chigani 2010].

- **Capabilities Specification:** An alternative approach to baselining requirements before initiating the architecting process is to describe the system in terms of its capabilities. We followed this approach to document the results of the study we conducted to illicit the requirements of SINERGY. The result of this study was compiled as design science framework for developing group decision support systems and submitted as a journal article for publication to the Decision Support Systems journal [Chigani *et al.* 2011].

- **Architecture Specification:** The architecture models that describe SINERGY are one of the first complete examples of DoDAF 2.0 based architecture descriptions available for public use. To serve this purpose, we have created an HTML-based presentation of the architecture specification and published it in a website for public use [Chigani and Balci 2011].

- **Architecture Evaluation:** We have followed MSAAM as methodology to evaluate SINERGY architecture. This dissertation documents how an architecture evaluation is conducted. Therefore, it serves as a learning tool to train architects on how to assess architectures and apply MSAAM.

8.2.3. Assessing SOA Security

Architecture is essential for achieving quality attributes such as security. In this work, we established an approach to address quality attributes that is based on 1) understanding the underpinnings of the quality attribute, 2) identifying the concerns related to that particular attribute, and 4) deriving indicators that help in assessing the attribute. We contributed a hierarchy of indicators to assess the quality of SOA system quality. An initial version of the hierarchy was presented and published in [Chigani and Balci 2009].

8.3. RECOMMENDATIONS FOR FUTURE WORK

Building on the results of this research, future research agenda could focus on the following directions.

8.3.1. Product Line Architectures

The architecture specification described in this dissertation serves as a foundation for developing a product line architecture. Currently, the general assumptions that have been made regarding SINEGY architecture could be specified to particular types of campuses. For each campus type, a pertinent architecture can be proposed using the product line architecture.

8.3.2. Prototyping

To show the fitness for use and help the design and development of a SINERGY implementation, prototyping could be explored to investigate detailed requirements and design issues. This research direction is best suited for development-driven research and could involve both undergraduate and graduate student researchers.

8.3.3. Agile Architecting

Software architecture is synonymous to the idea of "the big picture." However, the notion of big-design-up-front flies in the face of most agile believers and is often considered a deviation from the esteemed agile manifesto. In reality, though, organizations that use agile often tweak existing agile methods to incorporate some form of architecting in their approaches–whether they call it the envisioning, iteration zero, or planning. To scale agile methods up, it necessitates some sort of knowing the big picture. This is increasingly and openly expressed by agileists as evidenced in Scott Ambler's twit around the time of Agile 2009 Conference saying that "*high-level requirements and architecture […]*" are needed to enable "*[…] agile projects get off to the right start."*

Agile is a philosophy (or a way of thinking) rather than a set of software development practices (e.g., TDD, Pair-Programming) and process models (e.g. eXtreme Programming (XP), Scrum, Lean, Feature-Driven Development (FDD)). From an architecture perspective, there is value in incorporating agile principles (and some practices) into architecture-centric methods to accommodate the changes in architectural drivers during the development of a large-scale system. The main agile-like practices that can be adopted in the architecting process include:

- User stories, which can describe the functional capabilities that the system must provide and that can be mapped to quality scenarios;
- Quality scenarios, which can describe one or more quality characteristics of the system (e.g., maintainability, interoperability, and usability);
- The "architecture wall" concept, which literally imitates the wall agileists use to post and organize index cards and post-it notes describing the architecturally-relevant capabilities and their corresponding architectural blueprints; and
- Architectural refactoring, which refers to the way index cards and post-it notes are shifted around in a structured manner to ensure the integrity of the existing relationships among the user stories, among quality scenarios, and between stories and quality scenarios.

8.3.4. Uses of Architecture Beyond Quality Attributes

The "system of systems" perspective dominates much of the engineering now as solutions are more and more network-centric and are composed of multiple interconnected systems to support emerging missions and business goals. Under this paradigm, new IUs of architecture emerge. Typically, architecture artifacts are used to guide downstream development processes (e.g. design, development, deployment, and maintenance) in achieving system quality attributes. However, architecture specifications can serve higher-level purposes beyond actual system development. Further research can be conducted to explore how architecture specifications can answer the following questions:

- How does architecture support decision makers in acquisition decisions about whole systems, system components, and technologies?
- How can architecture facilitate capabilities gap analysis of an enterprise and create an overall strategy for aligning independent systems (existing and new) into a common use or purpose.
- How can architecture be used to drive innovative thinking through defining the "true and real" requirements of a system of systems early on in the life cycle?

DoDAF 2.0 advocates the description of architectures from perspectives that address issues beyond the mere development of a particular system. Therefore, as architecting practices mature and most importantly become widely adopted, the IUs of architecture specifications will continue to expand fulfilling the architecture community vision for a software/system engineering field that is primarily architecture-centric.

BIBLIOGRAPHY

IEEE (1998), "IEEE Standards for Software Verification and Validation," IEEE Standard 1012-1998.

Ackoff, R.L. (1971), "Towards A System of Systems Concepts," *Management Science 17, 11*, 661-671.

Adam, N.R., V. Atluti, S.A. Chun, J. Ellenberger, B. Shafiq, J. Vaidya, and H. Xiong (2008), "Secure Information Sharing and Analysis for Effective Emergency Management," In *Proceedings of the 2008 International Conference on Digital Government Research, ACM International Conference Proceeding Series*, Digital Government Society of North America, pp. 407-408.

Agrawala, A., J. Krause, and S. Vestal (1992), "Domain-Specific Software Architectures for Intelligent Guidance, Navigation and Control," In *Proceedings of the IEEE Symposium on Computer-Aided Control System Design*, Napa, CA, pp. 110-116.

Albers, M.J. (1999), "Information Design Considerations for Improving Situation Awareness in Complex Problem-Solving," In *Proceedings of the 17th Annual international Conference on Computer Documentation*, ACM, New York, NY, pp. 154-158.

Alberts, D.S. (2002), *Information Age Transformation: Getting to a 21st Century Military*, CCRP Publications Series, Washington, DC.

Alberts, D.S., J.J. Garstka, and F.P. Stein (2000), *Network Centric Warfare: Developing and Leveraging Information Superiority (2nd Edition Revised)*, CCRP Publications Series, Washington, DC.

Allen, B.B. (2005), "A Strategy for Managing the Development and Certification of Net-Centric Services within the Global Information Grid," Technical Report Draft v0.96, Defense Information Systems Agency (DISA).

Allen, R. and D. Garlan (1994), "Formalizing Architectural Connection," In *Proceedings of the 16th International Conference on Software Engineering, International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos, CA, pp. 71-80.

Amouzegar, H., S. Mohammadi, M.J. Tarokh, and A.N. Hidaji (2008), "A New Approach on Interactive SOA Security Model Based on Automata," In *Proceedings of 7th IEEE/ACIS International Conference on Computer and Information Science*, IEEE Computer Society, Washington, DC, pp.667-671.

Anderson, R. (2001), *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, NY.

Anley, C., J. Heasman, F. Linder, and G. Richarte (2004), *The Shellcoder's Handbook: Discovering and Exploiting Security Holes*, John Wiley & Sons, Hoboken, NJ.

Balci, O. (2001), "A Methodology for Certification of Modeling and Simulation Applications," *ACM Transactions on Modeling and Computer Simulation 11, 4*, 352–377.

Balci, O. (2011), "Architecting Network-Centric System of Systems," http://manta.cs.vt.edu/cs6704/.

Balci, O. and W.F. Ormsby (2006), "Quality Assessment of Modeling and Simulation of Network-Centric Military Systems," In *Modeling and Simulation Tools for Emerging Telecommunications Networks: Needs, Trends, Challenges and Solutions*, A.N. Ince and E. Topuz, Eds, Springer, New York, NY, Chapter 19, pp. 365-382.

Balci, O. and W.F. Ormsby (2007), "Scenario Simulation for Network-Centric Technology Assessment," In *Recent Advances in Modeling and Simulation Tools for Communication Networks and Services*, A.N. Ince and A. Bragg, Eds, Springer, New York, NY, Chapter 10, pp. 203-226.

Balci, O. and W.F. Ormsby (2008), "Network-Centric Military System Architecture Assessment Methodology," *International Journal of System of Systems Engineering 1, 1-2,* 271-292.

Banbury, S.P., D.G. Croft, W.J. Macken, and D.M. Jones (2004), "A cognitive Streaming Account of Situation Awareness," In S. Banbury and S. Tremblay (Eds) *A Cognitive Approach to Situation Awareness: Theory and Application.* (Ashgate: Aldershot).

Barnett, M. (1999), "The Seven Deadly Sins of Network-Centric Warfare," Technical Report, Center for Naval Analyses, Alexandria, VA.

Bass, L., P. Clements, and R. Kazman (2003), *Software Architecture in Practice*, Addison-Wesley, Reading, MA.

Baumgartner, N.W. Retschitzegger, and W. Schwinger (2008), "A Software Architecture for Ontology-Driven Situation Awareness," In *Proceedings of the 23rd ACM Symposium on Applied Computing*, Fortaleza, Cear´a, Brazil, pp. 2326-2330.

Bell, M. (2008), *Service-Oriented Modeling*, John Wiley & Sons, Hoboken, NJ.

Bellovin M.S., T.V. Benzel, B. Blakley, D.E. Denning, W. Diffie, J. Epstein, and P. Verissimo (2008), "Information Assurance Technology Forecast 2008," *IEEE Privacy and Security 6, 1*, 16-23.

Bernstein, L. (2005), *Young People's Concerts*, Amadeus Press LLC, Pompton Plains, NJ.

Bhargavan, K., C. Fournet, A.D Gordon, and S. Tse (2008), "Verified Interoperable Implementations of Security Protocols," *ACM Transactions on Programming Languages and Systems 31, 1*, 1-61.

Bhargavan, K., C. Fournet, and A.D. Gordon (2004), "Verifying Policy-based Security for Web Services," In *Proceedings of the 11th ACM conference on Computer and Communications Security*, ACM, New York, NY, pp. 268–277.

Boehm, B. (2008), "Making a Difference in the Software Century", *IEEE Computer 44, 3*, 32-38

Boehm, B. and J.A. Lane (2006), "21st Century Processes for Acquiring 21st Century Software-Intensive Systems of Systems," *CrossTalk, 9, 11*, 5-9.

Boeing (2006), "System-Wide Information Management (SWIM)," The Boeing Company http://www.boeing.com/.

Bose, I. and X. Chen (2009), "A Framework for Context Sensitive Services: A Knowledge Discovery based Approach," *Decision Support Systems 48,* 158-168.

Burstein, F., Brezillon, P., and Zaslavsky, A. (2011), *Supporting real time decision making: The role of context in decision support on the move*, Springer, New York, NY.

C-CERT (2008), "Training for Campus Emergency Response Management," Campus Community Emergency Response Team, http://www.c-cert.msu.edu/.

Campus Technology (2010), "Campus Technology Webcasts /Webinars," Campus Technology, http://campustechnology.com/.

Carey, J.M. (2008), "SOA What?," *IEEE Computer 44, 3*, 92-94

Chigani, A. (2007), "Guiding Network-Centric Architectural Design: A Style-based Approach," Master's Thesis etd-12262007-133121, Digital Library and Archives, Virginia Tech, Blacksburg, VA.

Chigani, A. (2010), "Agile Architecting: Using Agile Principles to Agilitize the Architecting Process," In *6th Software Architecture Technology User Network Conference (SATURN 2010)*, Minneapolis, MN.

Chigani, A. and J.D. Arthur (2007), "The Implications of Network-Centric Software Systems on Software Architecture: A Critical Evaluation," In *Proceedings of the 45th ACM Southeast Conference*, Winston-Salem, NC, pp. 75-80.

Chigani, A. and O. Balci (2009), "Assessing the Security of an SOA-based Network-Centric System," Doctoral Symposium, In *11th International Conference on Software Reuse*, Falls Church, VA.

Chigani, A. and O. Balci (2011), "SINERGY Architecture Specification," http://manta.cs.vt.edu/sinergy/

Chigani, A. and O. Balci (2011), "The Process of Architecting for Software/System Engineering," *International Journal of System of Systems Engineering (IJSSE)*.

Chigani, A., J.D. Arthur, and S. Bohner (2007), "Architecting Network-Centric Software Systems: A Style-based Beginning," In *Proceedings of the 31st Annual IEEE Software Engineering Workshop (in conjunction with 3rd IEEE Systems and Software Week)*, Columbia, Maryland, pp. 290-299.

Chigani, A., R. Barkhi, and O. Balci (2011), "Decision Support for Campus Emergency Response," submitted for publication.

Chmielewski, L., R. Brinkman, J.H. Hoepman, and B. Bos (2008), "Using JASON to Secure SOA," In *Proceedings of the 2008 Workshop on Middleware Security*, ACM, New York, NY, pp. 13-18.

Christensen, M.J. and R.H. Thayer (2001), *The Project Manager's Guide to Software Engineering's Best Practices*, IEEE Computer Society, Los Alamitos, CA.

Clements, P., F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford (2003), *Documenting Software Architectures: Views and Beyond*, Addison-Wesley, Reading, MA.

Clements, P., R. Kazman, M. Klein, D. Devesh, S. Reddy, and P. Verma (2007), "The Duties, Skills, and Knowledge of Software Architects," In *Proceedings of the Working IEEE/IFIP Conference on Software Architecture (WICSA)*, Mumbai, India, pp. 20-23.

Cusumano, M.A. (2008), "The Changing Software Business: Moving from Products to Services," *IEEE Computer 44, 1*, 20-27.

Dandashi, F. and H.W. Ang (2006), "Tailoring DoDAF for Service-Oriented Architecture: A Recommended Guide," In *Proceedings of the Military Communications Conference*, IEEE Computer Society, Piscataway, NJ, pp. 1-8.

Delessy, A.N. and E.B. Fernandez (2008), "A Pattern-Driven Security Process for SOA Applications," In *Proceeding of the 3rd International Conference on Availability, Reliability, and Security*, IEEE Computer Society, Los Alamitos, CA, pp. 416-421.

DeSanctis, G. and R.B. Gallupe (1987), "A foundation for the study of group decision support systems," Management Science 33, 5, 589-609.

DHRM (2007). "Adjunct Emergency Workforce," Virginia's Department of Human Resource Management, http://www.dhrm.state.va.us/AdjunctEmergencyWorkforce.html.

DHS (2006), "Security in the Software Lifecycle," U.S. Department of Homeland Security, http://www.stsc.hill.af.mil/.

DHS (2009), "Building Security In," U.S. Department of Homeland Security, https://buildsecurityin.us-cert.gov/daisy/bsi/547.html.

Dijkstra, E.W. (1982), "On the Role of Scientific Thought," In *Edger W. Dijkstra Selected Writings on Computing: A Personal Perspective*, Springer-Verlag, New York, NY, pp. 60-66.

Dineley, D. (2005), "Data-Centric Architectures," *InfoWorld 11*, 38-40.

DoDAF (2009a), "DoD Architecture Framework Version 2.0 Volume I: Introduction, Overview, and Concepts - Manager's Guide," Architecture Framework Working Group, Washington, DC.

DoDAF (2009b), "DoD Architecture Framework Version 2.0 Volume II: Architectural Data and Models - Architect's Guide," Architecture Framework Working Group, Washington, DC.

DoDAF (2009c), "DoD Architecture Framework Version 2.0 Volume III: DoDAF Meta-model Physical Exchange Specification - Developer's Guide," Architecture Framework Working Group, Washington, DC.

DoE (2008), "Readiness and Emergency Management for Schools Technical Assistance Center," U.S. Department of Education, http://rems.ed.gov.

Dolan, T.J (2002), "Architecture Assessment of Information-System Families," Ph.D. Dissertation, Department of Technology Management, Eindhoven University of Technology, Eindhoven, Netherlands.

DoT (2000), "TEAF: Treasury Enterprise Architecture Framework 1.0," Chief Information Officer Council, U.S. Department of Treasury, http://www.treasury.gov/Pages/default.aspx.

Dourish, P. and V. Bellotti (1992), "Awareness and Coordination in Shared Work Spaces", In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, Toronto, Ontario, Canada, pp. 107-114.

Durbeck, S., R. Schillinger, and J. Kolter (2007) "Security Requirements for a Semantic Service-oriented Architecture," In *Proceedings of Second International Conference on Availability, Reliability and Security*, IEEE Computer Society, New York, NY, pp. 366-373.

Echahed, R. and F. Prost (2005), "Security Policy in a Declarative Style," In P*roceedings of the 7th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, ACM, New York, NY, pp. 153-163.

Ellison, J.R. A.P. Moore, L. Bass, M. Klein, F. Bachmann (2004), "Security and Survivability Reasoning Frameworks and Architectural Design Tactics," Technical Note CMU/SEI-2004-TN-022, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

Endrei, M.J., J. Ang, A. Arsanjani, S. Chua, P. Comte, P. Krogdahl, M. Luo, and T. Newling (2004), *Patterns: Service-Oriented Architectures and Web Services*, IBM Redbooks.

Endsley, M. (2000), *Theoretical Underpinnings of Situation Awareness: A Critical Review*, Lawrence Erlbaum Associates Inc., Mahwah, NJ.

Epstein, J., S. Matsumoto, and G. McGraw (2006), "Software Security and SOA: Danger, Will Robinson!" *IEEE Privacy and Security 4, 1*, 80-83.

Erl, T. (2005), *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*, Prentice Hall, Boston, MA.

Erl, T. (2006), *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall, Boston, MA.

Erl, T. (2008), *Principles of Service Design*, Prentice Hall, Boston, MA.

Erlanger, L. (2005), "Secure Architectures," *InfoWorld 11*, 42-44.

ERMS (2008), "Notification and Emergency Management Solutions," Emergency Response Management Services Corporation, http://ermscorp.com/messenger-mass_notification_systems.

Erradi, A., S. Anand, and N. Kulkarni (2006), "SOAF: An Architectural Framework for Service Definition and Realization," In *Proceedings of the IEEE International Conference on Services Computing*, IEEE Computer Society, Washington, DC, pp. 151-158.

Farmer, D. and W. Venema (2004), *Forensic Discovery*, Addison-Wesley, Reading, MA.

Fay, D. (2003), "An Architecture for Distributed Applications on the Internet: Overview of Microsoft's .NET Platform," In *Proceedings of the International Parallel and Distributed Processing Symposium*, Nice, France, pp. 90

FEA (1999), "Federal Enterprise Architecture Framework 1.1," U.S. Federal Government Chief Information Officer (CIO), http://www.cio.gov/documents/fedarch1.pdf.

FEMA (2010a), "Incident Command System (ICS)," Federal Emergency Management Agency (FEMA), http://training.fema.gov/EMIWeb/IS/ICSResource/assets/reviewMaterials.pdf.

FEMA (2010b), "FEMA: What We Do," Federal Emergency Management Agency (FEMA), http://www.fema.gov/about/divisions/mitigation.shtm.

Fielding, R.T. (2000), "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. Dissertation, Information and Computer Science Department, University of California, Irvine, CA.

Freebersyser, J.A. and J.P. Macker (2001), "Realizing the Network-Centric Warfare Vision: Network Technology Challenges and Guidelines," Report Number A269464, DARPA and Naval Research Laboratory.

Friedman, T.L. (2005), *The World Is Flat: A Brief History of the Twenty-First Century*, Farrar, Straus & Giroux, New York, NY.

Garlan, D. (2000),"Software Architecture: A Roadmap," In *Proceedings of the Conference on the Future of Software Engineering*, ACM Press, Limerick, Ireland, pp. 91-101.

George F., M.T. Mattson, and K.D. Henderson (2001), *Campus Security: Situational Crime Prevention in High-Density Environments*, Criminal Justice Press-Willow Tree Press, Monsey, NY.

Glass, R. (2008), "Essentials of SOA", *Software Tech News 11*, 1, 6-9.

Gregoire, J., K. Buyens, B.D. Win, R. Scandariato, and W. Joosen (2007), "On the Secure Software Development Process: CLASP and SDL Compared," In *Proceedings of the 3rd International Workshop on Software Engineering For Secure Systems, International Conference on Software Engineering*, IEEE Computer Society, Washington, DC, pp. 46.

Gundersen, D.E., D.L. Davis, and D.F. Davis (1995) "Can DSS Technology Improve Group Decision Performance for End Users? An Experimental Study," *Journal of End User Computing 7, 2*, 3-10.

Heyman, T., K. Yskout, R. Scandariato, and W. Joosen (2007), "An Analysis of the Security Patterns Landscape," In *Proceedings of the 3rd International Workshop on Software Engineering for Secure Systems*, IEEE Computer Society, Washington, DC.

Hinton, H., M. Hondo, and B. Hutchison (2005), "Security Patterns within Service-Oriented Architecture," Technical Report, IBM.

IASA (2010), "IT Architect Certification," International Association of Software Architects, http://www.iasaglobal.org/iasa/Certification.asp?SnID=468592411.

IBM (2002), "Security in a Web Services World: A Proposed Architecture and Roadmap", Joint White Paper with Microsoft Version 1.0, http://www-128.ibm.com/developerworks/library/specification/ws-secmap.

IBM (2005), IBM Systems Journal, Volume 44, Number 4, special issue on Service-Oriented Architecture.

IBM (2006), "Web Services Policy Framework (WSPolicy)", Joint White Paper with BEA Systems, Microsoft, SAP AG, Sonic Software, and VeriSign, March 2006, Version 1.2, http://www-128.ibm.com/developerworks/library/specification/ws-polfram/.

IBM (2008a), “Rational Software Architect,” International Business Machines Corporation, http://www-306.ibm.com/software/awdtools/architect/swarchitect/.

IBM (2008b), “Web Service Architecture Overview,” International Business Machines Corporation, http://www.ibm.com/developerworks/library/w-ovr/#h2.

IBM (2009), "Federated ESB,” International Business Machines Corporation, http://www-306.ibm.com/software/websphere/landings/esb.jsp.

IBM (2010), “The IBM Professional Certification Program,” International Business Machines Corporation, http://www-03.ibm.com/certify/.

ICS (2010), “The Incident Command System,” Federal Emergency Management Agency (FEMA), http://www.fema.gov/emergency/nims/IncidentCommandSystem.shtm.

IEEE (2007), “Recommended Practice for Architectural Description of Software-Intensive Systems, ANSI/IEEE Std 1471-ISO/IEC 42010,” Institute of Electrical and Electronics Engineers, http://www.iso-architecture.org/ieee-1471.

Imamura, T., M. Tatsubori, Y. Nakamura, and C. Giblin (2005), “Web Services Security Configuration in a Service-Oriented Architecture,” In *Special interest Tracks and Posters of the 14th International Conference on World Wide Web*, ACM, New York, NY, pp. 1120-1121.

InfoQ (2007), “Web Services Standards Overview,” InfoQ Website, http://www.infoq.com/news/2007/03/innoq-ws-standards-poster.

Jain, S. and C. McLean (2003), “A Framework for Modeling and Simulation for Emergency Response,” In *Proceedings of the 2003 Winter Simulation Conference*, New Orleans, LA, pp. 1068-1076.

James, G.A., R.A. Handler, A. Lapkin, and N. Gall (2005), “Gartner Enterprise Architecture Framework,” http://www.gartner.com/it/products/research/asset_129493_2395.jsp.

Jason, L.H. (2003), “System Architecture for Wireless Sensor Networks,” Ph.D. Dissertation, Department of Computer Science, University of California, Berkeley, CA.

John, U. and D.A. Comerford (2002), “A Review of Situation Awareness Literature Relevant to Pilot Surveillance Functions,” DOT/FAA/AM-02/3, Department of Psychology, Kansas State University, Manhattan, KS 66056.

Juric, M.B. and K. Pant (2008), *Business Process Driven SOA using BPMN and BPEL*, Packt Publishing, Birmingham, U.K.

Kaler, C. and A. Nadalin (2007), “Web Services Federation Language,” http://www-106.ibm.com/developerworks/webservices/library/ws-fed/.

Kazman, R., L. Bass, M. Webb, and G. Abowd (1994), “SAAM: A Method for Analyzing the Properties of Software Architectures,” In *Proceedings of the 16th International Conference on Software Engineering*, Sorrento, Italy, pp. 81-90.

Kazman, R., M. Klein, and R. Nord (2003),”Tailorable Architecture Methods,” In *Proceedings of 28th Annual NASA Goddard Software Engineering Workshop*, IEEE Computer Society, pp. 152-156.

Keen, M. A. Acharya, S. Bishop, A. Hopkins, S. Milinski, C. Nott, R. Robinson, J. Adams, and P. Verschueren (2004), "Patterns: Implementing an SOA Using an Enterprise Service Bus", IBM Redbook, http://www.redbooks.ibm.com/abstracts/sg246346.html.

Keen, M., O. Adinolfi, S. Hemmings, A. Humpherys, H. Kanthi, and A. Nottingham (2005), "Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6", IBM Redbook, http://www.redbooks.ibm.com/abstracts/sg246494.html.

Kim, J.K., R. Sharman, H.R. Rao, and S. Upadhyaya (2007), "Efficiency of Critical Incident Management Systems: Instrument Development and Validation," *Decision Support Systems 44*, 235-250.

Kossiakoff, A. and W.N. Sweet (2003), *Systems Engineering: Principles and Practice*¸ John Wiley & Sons, Hoboken, NJ.

Krafzig, D., K. Banke, and D. Slama (2004), *Enterprise SOA: Service-Oriented Architecture Best Practices*, Prentice Hall, Boston, MA.

Krishnamurthy, L. (2006), "Comparative Assessment of Network-Centric Software Architectures", M.S. Thesis, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Kruchten, P. (1995), "Architectural Blueprints—The 4+1 View Model of Software Architecture," *IEEE Software 12, 6*, 42-50.

Lassing, N. (2002), "Architecture-Level Modifiability Analysis," Ph.D. Dissertation, Free University Amsterdam, Amsterdam, Netherlands.

Lee, J.Y., S.B. Han, H. Kim, and S.B. Park (1999), "Network-Centric Feature-based Modeling," In *Proceedings of the 7th Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, Washington, DC, pp. 280-291.

Lewis, F.L (2004), "Wireless Sensor Networks," *Smart Environments: Technologies, Protocols, and Applications*, New York.

Lewis, G., E. Morris, D.B. Smith, S. Simanta, and L. Wrage (2007), "Common Misconceptions about Service-Oriented Architecture," *The Journal of Defense Software Engineering: CrossTalk 20 , 11*, 27-30.

Linthicum, D. (2005), "SOA IT Strategy Guide," *InfoWorld*, San Francisco, CA.

Lorinez, K., D.J. Malan, T.R.F. Fulford-Jones, A. Nawoj, A. Clavel, V. Schnayder, G. Mainland, M. Welch, and S. Moulton (2004), "Sensor Networks for Emergency Response: Challenges and Opportunities," *IEEE Pervasive Computing 3, 4*, 16-23.

Luddy, J. (2005), "The Promise and the Challenge of Network-Centric Warfare," Technical Report, Lexington Institute, Arlington, VA.

Maier, M.W. (1998), "Architecting Principles for Systems-of-Systems," *Systems Engineering 1, 4*, 267–284.

March, S. and G.F. Smith (1995), "Design and Natural Science Research on Information Technology," *Decision Support Systems 15*, 251-266.

Margulius, D.L. (2005a), "Needs-based Architectures," *InfoWorld 11*, 49-50.

Margulius, D.L. (2005b), "Pervasive Architectures," *InfoWorld 11*, 46-48.

Margulius, D.L. (2005c), "Process-Driven Architectures," *InfoWorld 11*, 40-42.

Marks, E.A. and M. Bell (2006), *Service-Oriented Architecture (SOA): A Planning and Implementation Guide for Business and Technology*, John Wiley & Sons, Hoboken, NJ.

Matheus, C.J., M.M. Kokar, and K. Baclawski (2003), "A Core Ontology for Situation Awareness," In *Proceedings of Sixth International Conference on Information Fusion*, Cairns, Australia, pp.545-552.

McAllister, N. (2005), "Lightweight and Open Architectures," *InfoWorld 11*, 44-46.

McGrath, D., A. Hunt, and M. Bates (2005), "A Simple Distributed Simulation Architecture for Emergency Response Exercises," In *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, IEEE Computer Society, Washington, DC, pp. 221-228.

McGraw, G. (2005), "How Does Security Fit With Engineering?" Network Computing, http://www.networkcomputing.com.

Medvidovic, N., D.S. Rosenblum, D.F. Redmiles, and J.E. Robbins (2002), "Modeling Software Architectures in the Unified Modeling Language," *ACM Transactions on Software Engineering and Methodology 11, 1*, 2-57.

Microsoft (1996), "The Distributed Component Object Model Technical Overview," Microsoft Corporation, http://msdn.microsoft.com/en-us/library/Aa286561.

Microsoft (2002), "Security in a Web Services World: A Proposed Architecture and Roadmap," Technical Article, Microsoft Corporation, http://msdn.microsoft.com/en-us/library/ms977312.aspx.

Microsoft (2005), "Security Development Lifecycle," Microsoft Corporation, http://msdn2.microsoft.com/en-us/library/ms995349.aspx.

Microsoft (2007a), "COM: Component Object Model Technologies", Microsoft Corporation, http://www.microsoft.com/com/default.mspx.

Microsoft (2007b), "Web Services Security Specifications," Microsoft Corporation, http://msdn.microsoft.com/en-us/library/ms951273.aspx.

Microsoft (2010a), "Microsoft Certified Architect Program," Microsoft Corporation, http://www.microsoft.com/learning/en/us/certification/architect.aspx.

Microsoft (2010b), "Microsoft .NET Framework," Microsoft Corporation, http://www.microsoft.com/net/.

Monroe, R.T., D. Kompanek, R. Melton, and D. Garlan (1997), "Stylized Architecture, Design Patterns, and Objects," *IEEE Software 14, 1*, 43-52.

Naedele, M. (2003), "Standards for XML and Web Services Security," *Computer 36, 4*, 96-98.

NAF (2007), "NAF: NATO Architecture Framework version 3.0," NATO Consultation, Command and Control Board, http://www.nhqc3s.nato.int/ARCHITECTURE/_docs/NAF_v3/ANNEX1.pdf.

NCOIC (2008), "Accelerating Network Centric Systems through Industry Collaboration," Network Centric Operations Industry Consortium, https://www.ncoic.org.

Neal, D.M. and B.D. Phillips (1995), "Effective Emergency Management: Reconsidering the Bureaucratic Approach," *Disasters 19, 4,* 327-337.

NIST (2000), "Federal Information Technology Security Assessment Framework," Report, National Institute of Standards and Technology, Computer Security Division, Gaithersburg, MD.

Nord, R., M.R. Barbacci, P. Clements, R. Kazman, M. Klein, L. O'Brien, and J.E. Tomayko (2003), "Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM)," Technical Report CMU/SEI-2003-TN-038, Software Engineering Institute, Pittsburgh, PA.

O'Brien, L., P. Merson, and L. Bass (2007), "Quality Attributes for Service-Oriented Architectures," In *Proceedings of the International Workshop on Systems Development in SOA Environments*, IEEE Computer Society, Washington, DC.

OASIS (2002), "Security Assertion Markup Language (SAML)," Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/specs/#samlv2.0.

OASIS (2004a), "UDDI Version 3.0.2", Organization for the Advancement of Structured Information Standards, http://uddi.org/pubs/uddi_v3.htm.

OASIS (2004b), "Web Services Security: SOAP Message Security 1.1," Organization for the Advancement of Structured Information Standards, http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

OASIS (2006), "Reference Model for Service-Oriented Architecture 1.0," Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf.

OASIS (2007a), "WS-Federation," Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/committees/wsfed.

OASIS (2007b), "WS-SecureConversation 1.3", Organization for the Advancement of Structured Information Standards, http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html

OASIS (2007c), "Business Process Execution Language 2.0," Organization for the Advancement of Structured Information Standards, http://www.oasis-open.org/specs/#wsbpelv2.0.

OASIS (2008), "Web Services Reliable Messaging Policy Assertion (WS-RM Policy)," Organization for the Advancement of Structured Information Standards, http://docs.oasis-open.org/ws-rx/wsrmp/200702.

OMG (2010), "CORBA: The Common Object Request Broker Architecture," The Object Management Group, http://www.corba.org/.

Oracle (2004a), "Building a Network-Centric Warfare Architecture," White Paper, Oracle Corporation, Redwood Shores, CA.

Oracle (2004b), "Orchastrating Web Services; A Case for a BPEL Server," White Paper, Oracle Corporation, Redwood Shores, CA.

Oracle (2010), "Java Enterprise Edition 5 Technologies," Oracle Corporation, http://java.sun.com/javaee/technologies/javaee5.jsp.

Orca (2002), "Evaluation Environment," Orca Computer Inc., http://www.orcacomputer.com/eeHelp/eeHelp.htm.

Oreizy, P., N. Medvidovic, and R.N. Taylor (1998), "Architecture-based Runtime Software Evolution," In *Proceedings of the 20th International Conference on Software Engineering, International Conference on Software Engineering*, IEEE Computer Society, Washington, DC, pp. 177-186.

OWASP (2007), "The Ten Most Critical Web Application Security Vulnerabilities," The Open Web Application Security Project, http://www.owasp.org/images/e/e8/OWASP_Top_10_2007.pdf.

Owens, W.A. (1996), "The Emerging U.S. System of Systems," Strategic Forum 63, Institute for National Strategic Studies, National Defense University, Washington, DC.

Papastergiou, S., G. Valvis, and D. Polemi (2008), "A Holistic Anonymity Framework for Web Services," In *Proceedings of the 1st International Conference on Pervasive Technologies Related To Assistive Environments*, ACM, New York, NY, pp. 1-8.

Parnas, D. L., P.C. Clements, , and D.M. Weiss (1984), "The Modular Structure of Complex Systems," In *Proceedings of the 7th International Conference on Software Engineering, International Conference on Software Engineering*, IEEE Press, Piscataway, NJ, pp. 408-417.

Parnas, D.L. (1978), "Designing Software for Ease of Extension and Contraction," In *Proceedings of the 3rd International Conference on Software Engineering*, International Conference on Software Engineering, IEEE Press, Piscataway, NJ, pp. 264-277.

Paton, D. and R. Flin (1999), "Disaster Stress: An Emergency Management Perspective," *Disaster Management and Prevention 8, 4*, 261-267.

Perry, D.E. and A.L. Wolf (1992), "Foundations for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes 17, 4*, 40-52.

Perry, R.W. and M.K. Lindell (2003), "Preparedness for Emergency Response: Guidelines for the Emergency Planning Process," *Disasters 27, 4*, 336-350.

Pressman, R.S. (2010), *Software Engineering: A Practitioner's Approach*, McGraw-Hill, New York, NY.

Professional Underwriters (2006), "College Campus Security," White Paper, http://www.professionalunderwriters.com/.

Rahman, M.A., A. Schaad, and M. Rits (2006), "Towards Secure SOAP Message Exchange in a SOA," In *Proceedings of the 3rd ACM Workshop on Secure Web Services*, ACM, New York, NY, pp. 77-84.

Raytheon (2008), "Network Centric Systems," Raytheon Company, http://www.raytheon.com/businesses/rncs/.

Raytheon (2010), "RCAP: Raytheon Certified Architect Program," Raytheon Company, http://www.raytheon.com/technology_today/2009_i1/eye_on_tech_arch.html.

Reilly, W.S., S.L. Guarino, and B. Kellihan (2007), "Model-based Measurement of Situation Awareness," In *Proceedings of the 2007 Conference on Winter Simulation Conference*, IEEE Press, Piscataway, NJ, pp. 1353-1360.

Rosen, M., B. Lublinsky, K. T. Smith, M. J. Balcer (2008), *Applied SOA*, Wiley Publishing Inc., Indianapolis, IN.

Saaty, T.L. (1994), *Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process*, RWS Publications, Pittsburgh, PA.

Salmon, P.M., G.H Walker, D. Ladva, N.A. Stanton, D.P. Jenkins, and L. Rafferty (2007), "Measuring Situation Awareness in Command and Control: Comparison of Methods Study," In *Proceedings of the 14th European Conference on Cognitive Ergonomics*, ACM, New York, NY, pp. 27-34.

Sanders, D.T, J.A. Hamilton, and R.A. McDonald (2008), "Supporting a Service-Oriented Architecture," In *Proceedings of the 2008 Spring Simulation Multi-Conference*, ACM, New York, NY, pp. 325-234.

Sarter, N.B. and D.D. Woods (1991), "Situation Awareness: A Critical But Ill-Defined Phenomenon," *The International Journal of Aviation Psychology 1, 1*, 45-57.

Schafer, J.A. and B.H. Levin (2007), "Policing and Mass Casualty Events: Volume 3 of the Proceedings of the Futures Working Group," Technical Report, Futures Working Group, Behavioral Science Unit, FBI Academy, Quantico, VA.

Schmidt, D.C., R.E. Schantz, M.W. Masters, J.K. Cross, D.C. Sharp, and L.P. DiPalma (2001), "Towards Adaptive and Reflective Middleware for Network-Centric Combat Systems," *CrossTalk, 14, 11*, 10-16.

Schmidt, M., B. Hutchison, P. Lambros, and R. Phippen (2005), "The Enterprise Service Bus: Making Service-Oriented Architecture Aeal," *IBM Systems Journal 44, 4*, 781-797.

Schumacher, M., E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad (2006), Security Patterns, Wiley, NJ.

Security On Campus (2008), "A Non-Profit Organization for Safer Campus for Students," Security On Campus, http://www.securityoncampus.org/.

SEI (2001), "People Capability Maturity Model (P-CMM) Version 2.0," Software Engineering Institute, http://www.sei.cmu.edu/cmmi/tools/peoplecmm/.

SEI (2007), "Architecture Expert: ArchE," Software Engineering Institute, http://www.sei.cmu.edu/architecture/arche.html.

SEI (2008a), "Integration of Software-Intensive Systems (ISIS) Initiative: Addressing System of Systems Interoperability," Software Engineering Institute, http://www.sei.cmu.edu/sos/index.cfm.

SEI (2008b), "Ultra-Large-Scale Systems," Software Engineering Institute, http://www.sei.cmu.edu/uls/.

SEI (2010a), "Capability Maturity Model Integration (CMMI) for Development, Version 1.3," Software Engineering Institute, http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm.

SEI (2010b), "Software Architecture Certificate Programs," Software Engineering Institute, http://www.sei.cmu.edu/training/certificates/architecture/.

Shaw, M. (1998), "Architectural Requirements for Computing with Coalitions of Resources," Position Paper, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

Shaw, M. and P. Clements (2006), "The Golden Age of Software Architecture," *IEEE Software 23, 2*, 31-39.

Simon, H.A. (1981), *The Sciences of the Artificial (2nd Edition)*, MIT Press, Cambridge, MA.

Software Engineering Standards Committee (1999), "IEEE Standard for Information Technology—Software Life Cycle Processes—Reuse Processes," IEEE Computer Society, New York, NY.

Sommerville, I. (2007), *Software Engineering*, Addison-Wesley, Boston, MA.

Soni, B. (2005), "A New Service-Oriented Architecture Maturity Model," White Paper, BearingPoint Inc., http://www.bearingpoint.com.

Sonic Progress (2008), "Enterprise Service Bus," http://www.sonicsoftware.com/solutions/service_oriented_architecture/enterprise_service_bus/.

Steiner, D. (2006), "Oracle SOA Suite: Quick Start Guide," Technical Report B28938-01, Oracle Corporation, Redwood Shores, CA.

Steven, J. (2006), "Adopting an Enterprise Software Security Framework," *IEEE Privacy and Security 4, 2*, 84-87.

Talbert, M.L. (1995), "A Methodology for the Measurement and Evaluation of Complex System Designs," Ph.D. Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Telelogic (2008),"Telelogic System Architect," IBM, http://www.telelogic.com/products/systemarchitect/systemarchitect/overview.cfm.

Termini, T. (2008), *The Zen of SOA: An Executive Blueprint to Web-Enable Your Organization With Service-Oriented Architecture*, Blauer Hund Press, Washington, DC / Dublin, Ireland.

Than, D.V. and I. Jorstad (2005), "A Service-Oriented Architecture Framework for Mobile Services," In *Proceedings of the Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference*, IEEE, Piscataway, NJ, pp. 65-70.

Thomas, I. M. Menzel, and C. Meinel (2008), "Using Quantified Trust Levels to Describe Authentication Requirements in Federated Identity Management," In *Proceedings of the 2008 ACM Workshop on Secure Web Services*, ACM, New York, NY, pp. 71-80.

TOG (2009), "TOGAF: The Open Group Architecture Framework 9.0," The Open Group, http://www.opengroup.org/architecture/togaf9-doc/arch/.

TOG (2010), "ITAC: IT Architect Certification Program," The Open Group, http://www.opengroup.org/itac/.

Troy, D.A., A. Carson, J. Vanderbeek, and A. Hutton (2008), "Enhancing Community-based Disaster Preparedness with Information Technology," *Disasters 32, 1,* 149-165.

Tsipenyuk, K., B. Chess, G. McGraw (2005), "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors." *IEEE Privacy and Security 3, 6,* 81-84.

Turban, E., J.E. Aaronson, and T.P. Liang (1997), *Decision Support Systems and Intelligent Systems (5th Edition),* Prentice Hall PTR, Upper Saddle River, NJ.

Turoff, M. (2002), "Past and Future Emergency Response Information Systems," *Communications of the ACM 45, 4,* 29-32.

MoDAF (2008), "MoDAF: Ministry of Defense Architecture Framework 1.2," U.K. Ministry of Defense, http://www.modaf.org.uk/.

Udell, J. (2005), "Service-Oriented Architectures," *InfoWorld 11,* 48-49.

Urbaczewski, L. and S. Mrdalj (2006), "A Comparison of Enterprise Architecture Frameworks", *Issues in Information Systems 7, 2,* 18-23.

USNI (2007), "Proceedings Archive of the U.S. Naval Institute," U.S. Naval Institute, http://www.usni.org/magazines/proceedings/archive/.

VaCAS (2008), "Coordinated Sensing and Control for Surveillance and Tracking by Heterogeneous Autonomous Vehicles Teams," Virginia Center for Autonomous Systems, http://www.unmanned.vt.edu/discovery/coordinated.html.

Valerdi, R., E. Axelband, T. Baehren, B. Boehm, D. Dorenbos, S. Jackson, A. Madni, G. Nadler, P. Robitaille, and S. Settles (2008), "A Research Agenda for Systems of Systems Architecting," *International Journal of System of Systems Engineering 1,1-2,* 171-188.

Van Wyk, K.R. and G. McGraw (2005), "Bridging the Gap between Software Development and Information Security," *IEEE Privacy and Security 3, 5,* 75-79.

Vega, J. and J. Epstein (2006), "Why Applying Standards to Web Services Is Not Enough," *IEEE Privacy and Security 4, 4,* 25-31.

Vinoski, S. (2008), "Convenience Over Correctness," *IEEE Internet Computing 4, 12,* 89-92

Vogels, W. (2003), "Web Services Are Not Distributed Objects," *IEEE Internet Computing 7, 6,* 59-66.

VT (2007), "The We Remember Website," Virginia Tech, http://www.weremember.vt.edu/.

VT Review Panel (2007), "The Virginia Tech Review Panel Report – August 2007," The Virginia Tech Review Panel, http://www.vtreviewpanel.org/report/.

W3 Schools (2008), "eXtensible Markup Language Tutorial," http://www.w3schools.com/xml/.

W3C (2001a), "Notes from Workshop on Web Services," http://www.w3.org/2001/03/WSWS-popa/paper08.

W3C (2001b), "Web Service Description Language (WSDL) 1.1", http://www.w3.org/TR/wsdl.

W3C (2002), "XML Encryption Syntax and Processing," http://www.w3.org/TR/xmlenc-core/.

W3C (2003), "SOAP Version 1.2 Specification", http://www.w3.org/TR/soap12/.

W3C (2004), "Web Services Architecture," http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

W3C (2006a), "SOAP version 1.2," http://www.w3.org/TR/soap12.

W3C (2006b), "XML Signature," http://www.w3.org/Signature/.

W3C (2007a), "Web Services Policy 1.5-Attachment," http://www.w3.org/TR/ws-policy-attach/.

W3C (2007b), "Web Services Policy 1.5-Framework," http://www.w3.org/TR/ws-policy/.

Waldo, J. (1999), "The JINI Architecture for Network Centric Computing," *Communications of the ACM 42, 7*, 76-82.

WSO2 (2008a), "Enterprise Service Bus (ESB)," http://wso2.org/projects/esb/java.

WSO2 (2008b), "WSO2 ESB Documentation Index," http://wso2.org/project/esb/java/1.7/docs/docs_index.html.

Xiaochun, X., Guanghui, X. and Yongsen, X. (2002), "Architectural Issues in Network-Centric Computing," *ACM SIGSOFT Software Engineering Notes 27, 1*, 53-57.

Yoshioka, N., H. Washizaki, and K. Maruyama (2008), "A Survey on Security Patterns," *Progress of Informatics - Special Issue 5*, 35-47.

Zachman, A.J. (2009), "Zachman Framework for Enterprise Architecture," http://zachmaninternational.com/2/Zachman_Framework.asp.

Zhang, L. and Z. Wang (2006), "Integration of RFID into Wireless Sensor Networks: Architectures, Opportunities and Challenging Problems," In *Proceeding of the 5th International Conference on Grid and Cooperative Computing Workshops*, IEEE Computer Society, Hunan, China, pp. 463-469.

Zhang, S., and S. Goddard (2007), "A Software Architecture and Framework for Web-based Distributed Decision Support Systems," *Decision Support Systems 43*, 1133-1150.

Zlatanova, S. (2005), "A Proposed System Architecture for Emergency Response in Urban Areas," Directions Magazine, http://www.directionsmag.com/article.php?article_id=1989&trv=1.

APPENDIX A: INITIAL SET OF INTERVIEW QUESTIONS

List of initial set of interview questions

Question 1	Do you have any questions for me before we start the interview?
Question 2	Have you read the project description that I attached to my e-mail? Do you have any questions about that?
Question 3	What is your or your office's role in providing security to Virginia Tech's campus?
Question 4	What is your or your office's role in providing situational awareness to Virginia Tech's campus?
Question 5	What is your or your office's role in providing emergency response management to Virginia Tech's campus?
Question 6	My plan is to develop an architecture for a system that provides situational awareness and security control, and enables emergency response management. What kind of features you would like to see this system has?
Question 7	What kinds of quality characteristics you would like to see this system has (e.g., performance, usability, security, etc.)?
Question 8	From your experience, do you have any suggestions about capabilities you wished you had to respond effectively to emergencies you were involved with in the past?
Question 9	Do have any additional comments or advise you would like to share with me with regards to this research?
Question 10	Can I contact you via e-mail to ask you some follow up questions, if I have any?

APPENDIX B: IRB APPROVAL LETTER

VirginiaTech

Office of Research Compliance
Institutional Review Board
2000 Kraft Drive, Suite 2000 (0497)
Blacksburg, Virginia 24061
540/231-4991 Fax 540/231-0959
e-mail moored@vt.edu
www.irb.vt.edu

FWA00000572(expires 1/20/2010)
IRB # is IRB00000667

DATE: July 10, 2009

MEMORANDUM

TO: Osman Balci
Amine Chigani

Approval date: 7/10/2009
Continuing Review Due Date:6/25/2010
Expiration Date: 7/9/2010

FROM: David M. Moore

SUBJECT: **IRB Expedited Approval:** "Requirements Elicitation for a Network-Centric Architecture for SA, SC, and ERM in Open Campus Environment" , IRB # 09-592

This memo is regarding the above-mentioned protocol. The proposed research is eligible for expedited review according to the specifications authorized by 45 CFR 46.110 and 21 CFR 56.110. As Chair of the Virginia Tech Institutional Review Board, I have granted approval to the study for a period of 12 months, effective July 10, 2009.

As an investigator of human subjects, your responsibilities include the following:

1. Report promptly proposed changes in previously approved human subject research activities to the IRB, including changes to your study forms, procedures and investigators, regardless of how minor. The proposed changes must not be initiated without IRB review and approval, except where necessary to eliminate apparent immediate hazards to the subjects.
2. Report promptly to the IRB any injuries or other unanticipated or adverse events involving risks or harms to human research subjects or others.
3. Report promptly to the IRB of the study's closing (i.e., data collecting and data analysis complete at Virginia Tech). If the study is to continue past the expiration date (listed above), investigators must submit a request for continuing review prior to the continuing review due date (listed above). It is the researcher's responsibility to obtain re-approval from the IRB before the study's expiration date.
4. If re-approval is not obtained (unless the study has been reported to the IRB as closed) prior to the expiration date, all activities involving human subjects and data analysis must cease immediately, except where necessary to eliminate apparent immediate hazards to the subjects.

Important:
If you are conducting **federally funded non-exempt research**, please send the applicable OSP/grant proposal to the IRB office, once available. OSP funds may not be released until the IRB has compared and found consistent the proposal and related IRB applicaton.

cc: File

Invent the Future

VIRGINIA POLYTECHNIC INSTITUTE UNIVERSITY AND STATE UNIVERSITY

An equal opportunity, affirmative action institution