A UML-driven Enterprise Architecture Case Study

Dr. Frank Armour, ArmourIT, LLC.
Dr. Stephen Kaisler,U.S. Senate
Jim Getter, Doug Pippin,U.S. Capitol Police
(farmour@att.net; skaisler1@comcast.net; jim_getter, doug_pippin@cap-police.senate.gov)

Abstract

The U.S. Capitol Police (USCP) are responsible for protecting the Capitol Hill complex encompassing the Senate and House office buildings and the Capitol building. Currently, the USCP operate numerous legacy systems, many of which exist as stovepipes. They are modernizing their suite of administrative and law enforcement systems. The USCP recognized that developing an enterprise architecture was an essential first step to ensure that the proper solution was selected, that key integration points between the individual systems were understood, and that a prioritized set of initiatives to remediate, renovate, or replace information systems could be planned and executed. We describe how to use standard UML models to capture architectural view information and some specific extensions that address areas in which UML is limited.

Introduction

An Enterprise Information Technology Architecture (EITA) framework is a conceptual model for describing a set of information system architectures that support an organization's business operations. It is a blueprint for creating enterprise-wide information systems. Generically, an *architecture* is a description of a set of components and the relationships between them. The components of an EITA are information systems. Creating and maintaining an EITA is a very complex undertaking, because it represents multiple views of an enterprise including: its business processes, organizational structure, functional processes, and information requirements as well as the underlying technical infrastructure. Maintaining an accurate and consistent representation for each of these views as well as across the views is critical for organizations that need to deploy an enterprise solution in a rapidly changing environment.

The Unified Modeling Language (UML), provides a set of modeling tools that have primarily been utilized in software development. We have chosen the Unified Modeling Language (UML) as our descriptive language because it provides a common set of scaleable, integrated set of diagrams that cover most of the system architecting and software architecting processes. It has been accepted by the Object Management Group [10] as a mechanism for providing a common notation across multiple object-oriented analysis and design methodologies which promotes comparison of different methodologies' results. And, unlike many other description languages, it is both customizable and extensible for specific domains. In this research we have begun to extend the UML, through the use of stereotypes and tagged fields, to encompass the representation of multiple EITA views. This extension of UML provides the enterprise modeler with a toolkit of entities and relationships for each architectural view. It also helps support intra- and inter-view consistency, validation and impact analysis. In this paper, we present a quick overview of our EITA framework, which we have developed and used in multiple organizations. We describe the challenges in creating and maintaining architectural representations of the enterprise and describe a case study based on our work at the U.S. Capitol Police.

Large, complex enterprise-wide information systems are rarely built anew, but tend to evolve from a base of existing legacy systems. Emerging internet-based applications must also be integrated with existing systems. Such systems of systems are usually heterogeneous - in hardware, software, design, and implementation - and, are often geographically distributed. As the enterprise's business evolves, so must the information systems. Ensuring a consistent, coherent vision of the system architecture in order to guide the evolution of the enterprise's information systems is difficult without strategic thought. Enterprise system architecting is the process by which the vision is conceived, planned, executed and monitored. When performing enterprise architecting utilzing UML we have also found a number of concepts from the Agile development school to be very useful [3]. Agile concepts, such as short implementation cycles, high bandwidth communication between the various stakeholders and aggressive testing approaches are very useful in pragmatically delivering the functionality defined in the enterprise

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

architecture. Given the broad, deep, and diverse set of functions defined in an enterprise architecture, a formal modeling approach such as UML is critical in representing these complex aspects. However, an agile approach to its deployment is also important to deliver business results quickly.

Architecture Views

An EITA needs to capture a very complex and multidimensional picture of the enterprise. Just as we look at a city plan in different ways, so stakeholders have different perspectives or views of an enterprise. A data administrator will be concerned with data and information entities. A business analyst will be focused on the business processes and functions. A network engineer will be interested in the network technical infrastructure and topology. Capturing these multiple perspectives in separate views helps manage complexity, separates concerns, and addresses the different life spans of the architecture's elements.

An *architectural view* is a well-defined vantage point that describes a specific set of concerns. Each view is modeled by a set of *elements, relationships*, and *constraints*. These views provide a picture of how individual information systems fit into the larger structure that supports the organization's total business operations.

An *element* of an architectural view represents a physical or logical entity in the actual or proposed enterprise system architecture. An element may be an organization unit, a database, an application, a particularly skilled worker, or any of many other types of entities. A *relationship* specifies the physical or logical connection between two or more elements of an architectural view. For example, an organization unit is located in one (or more) physical location(s). A *constraint* specifies a physical or logical limitation on an element or a relationship between elements. For example, constraints may specify the bandwidth between network nodes, or the protocol to be used between two software components, or that French is the medium of conversation between organizational units. We are developing an initial set of elements, relationships, and constraints for modeling business enterprises as part of our investigation into an enterprise architecting ontology.

Architectural views will differ from framework to framework, but normally include views focusing on data or information, functions or processes, organizational or work structures, and the technical infrastructure. Other views that various frameworks may define include a security view, an applications view, etc. [16] [11] [9].

Enterprise IT Views

We use five views to develop enterprise system architectures as depicted in figure 1. [1] Our methodology uses a decomposition process to describe the major components of the architecture and their relationships and an aggregation and composition process to describe the characteristics of individual components. [2] [3] We begin by understanding the business mission and processes of the organization (Business View) [14] [13]. We factor out the information – elements and relationships -- about who performs and where they perform the business operations (Work View), the business information systems and functions (Functional View), and the data and information required to support the business areas (Information View). We also identify the hardware, software, and telecommunications components that enable automated systems (Technical View) [15]. Then, we describe the characteristics of each of the elements and relationships and the constraints upon them.

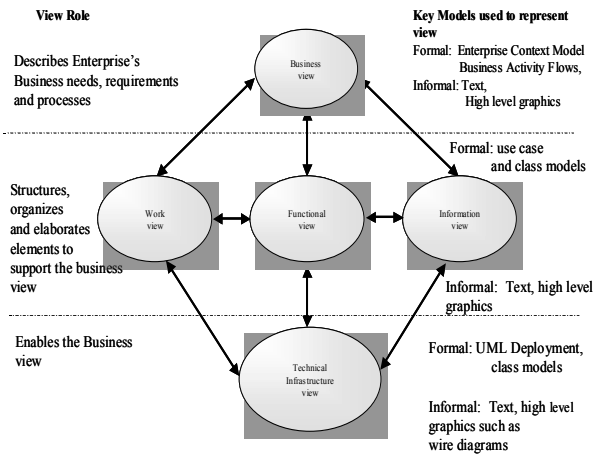


Figure 1. Enterprise IT Architecture Framework

Challenges in Modeling EITAs

Given the broad, complex and multi-dimensional nature of an EITA, the modeling challenges are many. These challenges include, but are not limited to:

1. The development and representation of a complex set of enterprise IT information systems requires multiple views.
2. How to specify and represent the information captured in each view in a

IEEE COMPUTER SOCIETY

clear, concise and understandable fashion.

3. How to present and validate this information to a diverse set of stakeholders with varied backgrounds, and technical and non-technical skill sets and interests, including:
   - CIO
   - High-level business customers
   - End users
   - IT developers
   - Systems analysts
   - Systems architects

4. There is then the challenge of integrating the views so that explicit traceability and decomposition and the nature of the "flow" and dependency from elements in one view to the elements in another view can be maintained.

While there are a number of architecture description languages available, they typically focus on software architectures, principally for single information systems. And, their syntax often closely resembles a programming language, which does not match the abstraction level of a system architecture.

Extending UML to Support an Enterprise System IT Architecture

To address the challenge of rigorously defining the models and concepts, we utilize the Unified Modeling Language (UML). UML was created primarily as a modeling language for the representation of a single software system. It does not contain all the elements we feel are needed to provide robust, understandable models to various stakeholders, to provide integrated representations across multiple "systems", and to provide traceability and integration across the views. However, the creators of UML have provided several extension mechanisms in UML, such as stereotypes, tagged values and constraints that allow the language to be extended so that it can be tailored for specific domains and environments. [12] [5] [10] For example, a stereotype is a way to alternatively classify elements in the model. It is used to signify that the element is different in some way from other elements of its type. Stereotypes are displayed as text surrounded by guillemets (<< >>). We will be utilizing some of these mechanisms to develop the information representations for an enterprise architecture domain. Stereotypes will be used to distinguish between types of elements that are being represented within the different views.

Overview of the Case Study

The United States Capitol Police (USCP) is responsible for the safety and security of the Members and staff of the House of Representatives and United States Senate in the U.S. Capitol and the Capitol Hill complex. The USCP currently operates a set of administrative systems including budget, personnel, training, time and attendance and inventory control. These systems are a heterogeneous collection of legacy mainframe software applications and commercial off-the-shelf software application products. The current systems are costly to operate and difficult to maintain. The infrastructure has been managed and developed on an as-needed basis and often tied to specific applications.

The current set of applications, written in the early 1980s in COBOL and Computer Associates' User Formatted On-Line (UFO) language, has failed to keep pace with the business processes of the USCP, and are difficult and costly to maintain. Consequently, the USCP initiated a business systems modernization process in the Fall of 2001 to remediate, renovate, or replace existing administrative and law enforcement information systems.

The USCP has mandated that the modernized administrative information systems will operate as a set of web-enabled, integrated applications running on one or more enterprise servers. Some of these applications interface with external systems through telecommunication links which will also be modernized.

The Capital Police modernization effort project is large and enterprise-wide in scope. As such, there will be multiple, phased releases, with multiple subprojects that are coupled together. Users have both functional and non-functional requirements that cross across the enterprise IT architecture, requiring the close coordination and management of multiple teams and locations. Our methodology helps to uncover these requirements and partition them across teams and architectural elements.

Given the extensive nature of this effort, the known dependencies and interrelationships between these systems, and the critical nature of the systems, the USCP determined that the creation of an EITA was critical to the successful understanding, planning and deployment of the systems over a multi-year time period. There will be key data, functional, and schedule dependencies that will need to be managed between efforts. We describe some of our extensions to UML in the following paragraphs that have facilitated communication between the architects, the stakeholders, and the system developers. We primarily

IEEE COMPUTER SOCIETY

used structured interviews with system owners, users and current IT personnel as well as interviews with other government agencies with similar missions to capture the architectural requirements.

Architectural Extensions to UML

Table 2 below outlines some of the specific challenges we have encountered. In this paper we focus on the following challenges specific to the business and functional views.

View	Challenges
Business	Capture "horizontal" processes or workflows that can cross multiple use cases and Business Application Packages Present the information in as "user" friendly form as possible, but still define the problem in enough rigor to highlight routes, roles and the information that is acted on. Clearly identifying what is in, and what is outside, the enterprise
Functional	Capture the interfaces across multiple applications Identify and define various interfaces to both external and internal actors
Information	Given the large amount of information that can be modeled related to an enterprise is to determine what data is relevant to the overall enterprise, model it at the right level of abstraction.
Work	Capturing the key aspects about departments, locations and roles
Technical Infrastructure	Robustly documenting the key technical components such as server platforms, client platforms, legacy systems, networks, middleware, etc. Determining the key "paths" and their characteristics, based on usage, through the technical components.

Table 2: Key Challenges when Modeling an Enterprise

Modeling the Business View

The business view is the *why* of information systems, describing the business applications and business processes that support the business mission and operations. We use this view to understand how the organization operates and how to align information systems to support business operations. The business view is often divided along major organizational applications, such as finance, human resources, manufacturing, and research and development. In addition, there are normally horizontal business processes that need to be captured. If you fail to consider the business's goals, objectives, and processes, you may develop a great architecture for the wrong business.

Selected modeling techniques that we use to the model the business view include:

- Enterprise Business Context Diagrams and
- Business Activity Diagrams

Enterprise Business Context Diagram

We wish to capture the overall context of the enterprise, its key Business operations and major external entities with which it interacts. We refer to this model as a Enterprise Business Context Diagram. In this diagram, we utilize UML packages to represent these operations. A collection of related business operations can be represented by a UML package. However, packages are just organizational mechanisms and do not capture the semantics of the relationships and interactions between the various business operations. We define a BusinessApplication as a functional area or major grouping of business operations. Thus, we extend the package concept through stereotypes to represent this semantic information. For brevity, in the following examples, we will only describe a few key elements. We stereotype packages into what we refer to as Business Application Packages (BAP) (represented as light gray on the original colored diagram). We include within the BAP's specification information on organization ownership, existing and proposed systems, etc. Each Business Application package is a logical construct, and does not reflect a concrete design decision. (Although we find that design does tend to break down in that format).

External Actor, representing entities external to the entire enterprise, is a stereotype of UML Actors. External Actors include NFC and NCIC, which are depicted outside the boundary of the Enterprise Business Context Diagram for the U.S. Capital Police, are represented in Figure 2.

IEEE COMPUTER SOCIETY

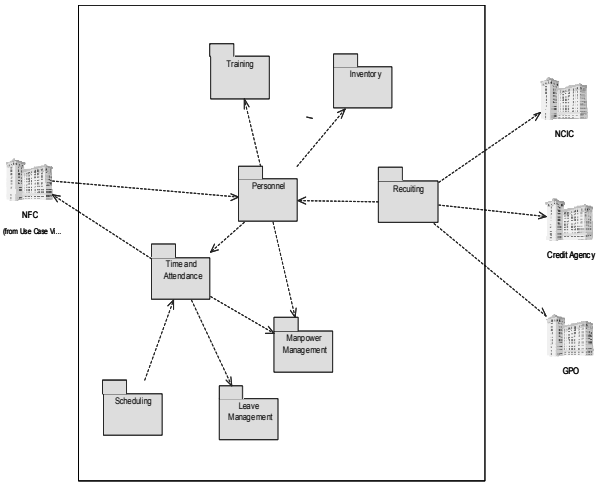


Figure 2: USCP Enterprise Context Diagram

Each BAP depicted in figure 2 has a detailed description associated with it. For example, the T&A application's partial description appears as follows:

Element	Business Operation
Represented By	Business Application Package
Name	Time and Attendance Applications
Short Name	T&A
Use Case List	See T&A use case model
Interfaces	Basic Personnel, Training
Interfaces With	Personnel, Scheduling, Leave Management, Training
Implemented By	To be Determined – Some Software Package
Hosted On	<<server name and type>>

Element	Automatic Activity
Represented By	Business Activity Diagram
Name	Perform Check In
Implements	Perform Check In
Uses	Daily Schedule
Produces	Daily T&A start record
Performed By	Sworn Officer

Element	Manual Activity
Represented By	Business Activity Diagram
Name	Verify T&A Certification Report
Uses	Daily T&A Data
Produces	T&A Certification Report
Performed By	Sworn Officer, Supervisor

Business Activity Diagrams

A UML Activity Diagram can also be used to represent the business activities that support a particular business process. This provides visibility into each organizational unit's role in the business operation. There are several limitations to the use of standard UML Activity diagrams, including the ability to:

- Distinguish between automated and manual processes,
- Represent a drill down of an automated activity into a subprocess
- Represent users and other entities participating in an activity.
- Represent documents and other artifacts created or consumed by activities

To address these issues we have extended activity diagrams in the following manner:

Stereotype Automated and Manual Activity States

We implemented stereotypes of Activity Diagrams to represent automated and manual activities. The automated activity state is depicted with a standard white background and a manual activity is depicted with a red background on the original colored diagram, as shown in figure 3 below.

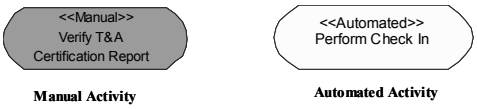


Figure 3: Stereotyped Activity Representations

A set of stereotyped classes represent elements consumed and produced by a business process. These elements include documents created and routed and datastores, which depict logical groups of information captured and stored, as shown below.

Figure 4: Stereotyped Classes for Logical Documents and Data groups

We also utilize various actor stereotypes to reflect the roles involved within a business process. These concepts will be more fully explained in the next section which deals with the use case model in the functional view.

We have found that it is important to visually show the traceability from a business activity to the use case(s) in the functional view that provide the functional details related to the activity. The stakeholders validating the Business Activity diagrams like to see this concept highlighted, rather then buried in traceability tables or matrixes. We provide explicit traceability from an activity(s) within a business view to the use case(s) in the functional view that realize the activities in more detail, using UML note elements, stereotyped as Activity-use case trace notes. These

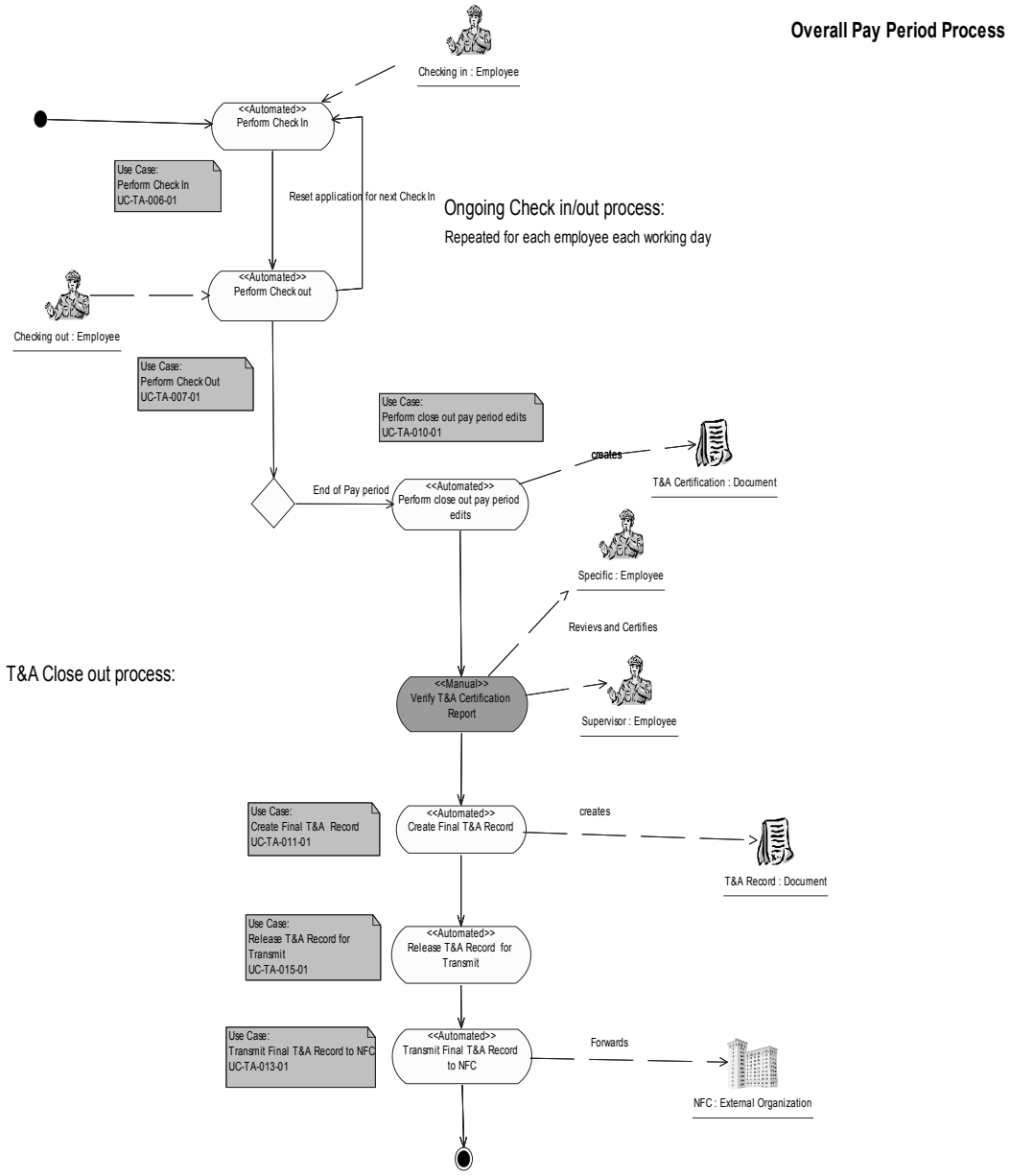
IEEE COMPUTER SOCIETY

have a light gray background (on the original colored diagram). Normally, there is at least one use case to one activity. However, a "many use cases to one activity" relationship can exist.

Figure 5: Activity-Use Case Trace Note

Figure 6 below is the Business Activity diagram that represents the overall pay period process for the USCP, please note the following UML extensions within this:

Figure 6 USCP T&A Pay Period Process diagram



IEEE COMPUTER SOCIETY

- The Use Case Traceability notes
- The Role icons such as employee and NFC (an external organization) as well as the T&A record (document created)

Functional View

The functional view describes the business information systems, functions, or activities that capture, manipulate, and manage the business information in support of the business operations. It also describes the logical dependencies and relationships among a set of applications and their functions. The identification of common functions also provides for the creation of common information systems and applications that can be shared across the organization and can be developed and maintained centrally.

The primary approach that we use to represent the functional view is *use case modeling*. Use cases, which traditionally describe interactions between an actor and a single system [7], can be extended to model enterprise business activities [8]. We refer to a set of use cases and accompanying documentation as the use case model. We normally represent the detailed descriptions of the use cases in text. However, organizations that are more visually oriented may supplement or replace use cases with UML activity diagrams [4].

For each Business Application Package defined in the Business View, we create a separate Use Case Model. Our rule is that each use case occurs in one and only one Business Application Package.

When modeling use cases within a BAP, we have developed several extensions to standard use case modeling that have proven to be very useful. First, several unique categories of external entities may be referenced within a use case. Each entity type reflects an entity outside the current "BAP" context, but represents a different form of Actor. During requirements analysis, in a single software development effort it is normally sufficient to represent rules as standard UML actors. However, in an EITA, there are number of different types of external entities each with a different perspective. To represent them, we have stereotyped UML Actors into *several categories as shown in figure 7:

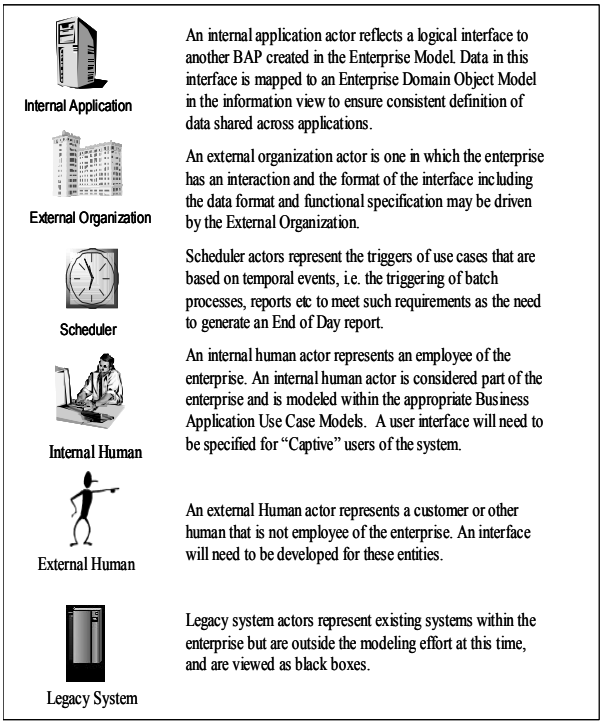


Figure 7: Stereotyped Actors

In internal applications, when an actor is defined to reflect an interface between multiple BAPs, there will be a use case in one BAP generating data that will be consumed by a use case in another BAP. This interaction may be part of a larger process(es), perhaps reflected in a business activity diagram in the business view. Since the deployment and implementation of an EITA will normally involve multiple projects and, perhaps, multiple vendors, it is very important that each interface be clearly highlighted to represent any dependency between projects. The data definition in the information view should be used to ensure that interfaces are defined in a consistent manner across the enterprise. We represent this logical data interface as a UML note tag with a blue background with white text and refer to as a *data interface tag*. The following three figures - 8, and 9 - represent use cases for the Personnel and T&A BAPs of the U.S. Capitol Police.

Figure 8 is a model of the Personnel BAP. Note the use of the data tags from the Recruiting BAP and the T&A BAP. Actor stereotypes include human, scheduler, internal application and external organization.

In Figure 9, we present the T&A BAP. Again note that data interface tags from the personnel BAP and to the Leave Management BAP..

IEEE COMPUTER SOCIETY

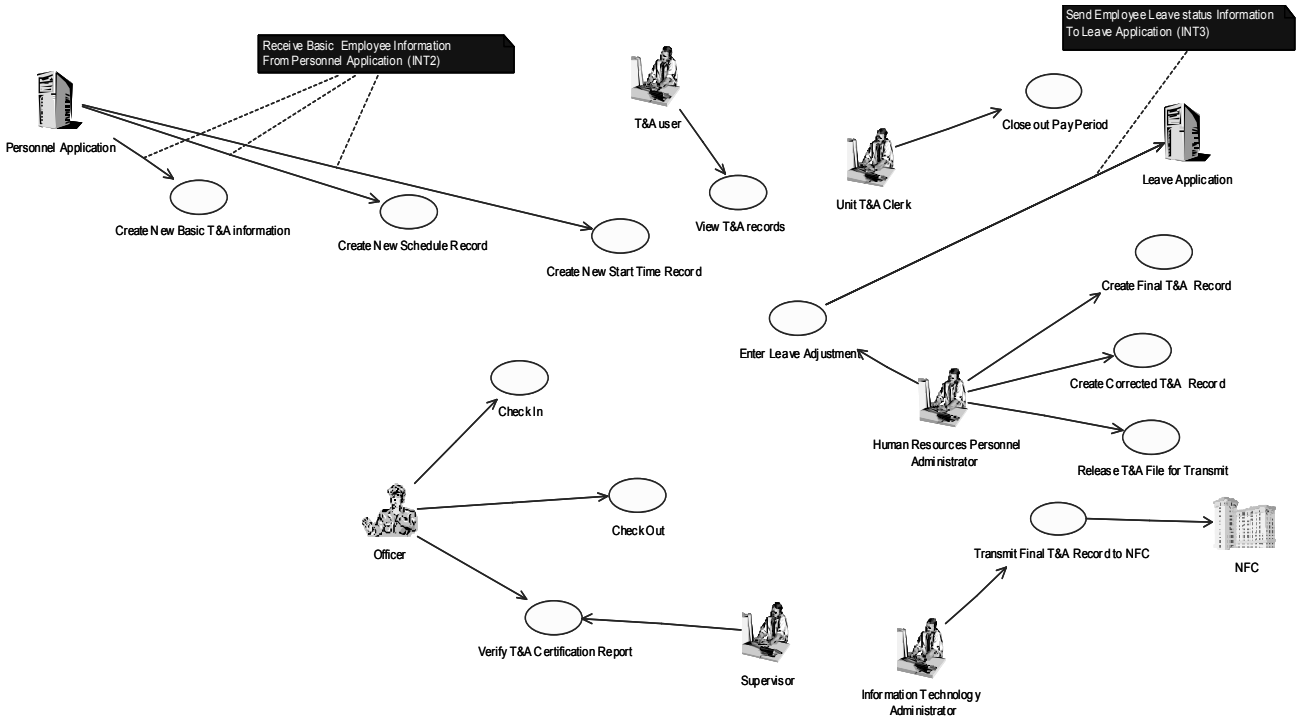


Figure 8: Partial Personnel BAP Use Case Diagram

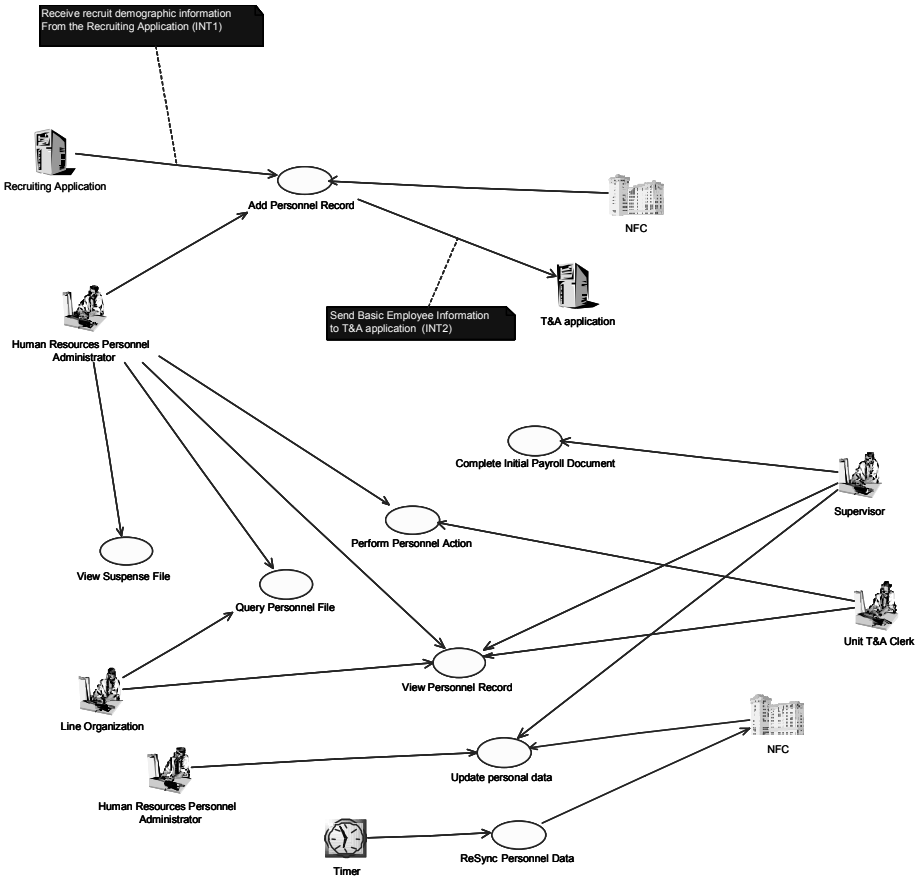


Figure 9: Partial T&A BAP Use Case Diagram

IEEE COMPUTER SOCIETY

We next briefly touch on the modeling approach for the other three enterprise IT architecture views

The Other Views

The information view models the data used by BAPS, utilized in the logical interfaces between BAPs, or is transferred between BAPs. Overall, we refer to this as the Enterprise Domain Object Model (EDOM), which consists of information sets and documents. Two stereotypes of UML classes are used to represent these elements. Each BAP will have its own Domain Object Model, a subset of the EDOM.

The work view describes the overall organizational structure, including, as necessary, any hierarchical relationships. The work view identifies the stakeholders, organizational units, the distribution of these work elements to business locations, and the communication and coordination between these work elements and between the business locations. These communications will help to drive data allocation in the information view, in functional view, and the physical data flow in the infrastructure view. It also describes the major operations performed by each business area and its divisions and may describe the types of work performed by the workers. The work view contains diagrams of the overall organizational hierarchy.

The business, work, functional, and information views describe the enterprise from four different points of view: business, work, functional, and information. The *technical infrastructure view* continues the description of the enterprise by focusing on the technology, including the hardware, software, and telecommunication components that are deployed to meet the needs of the organization. The infrastructure view helps to "ground" the other four views by validating that the technology exists to be able to implement them. In addition, the infrastructure view usually depicts the evolution of the technical architecture. We are developing extensions to capture the interplay between architectural elements, to include:

- Architectural Scenarios, utilizing UML state charts and Activity diagrams to capture the interplay between architectural components. These are based directly off the use cases and business activity diagrams. They represent "paths" through the architecture and will be used to allocate, test, and validate the feasibility of the enterprise-wide functional as well as nonfunctional requirements such as overall performance, capacity and security.

- An expanded UML deployment view with stereotyped classes to represent architectural elements such as systems, subsystems, nodes, platforms, databases and networks

Conclusion

Using UML with our extensions, we are developing a comprehensive set of context, process and use cases models for the overall USCP enterprise as well as each of the USCP administrative BAPs and several of the law enforcement systems. We used Rational Rose as our primary modeling tool. These models were used to validate the requirements, interrelationships, and constraints of each information system with the stakeholders. We are providing these models and detailed requirements (in Rational Requisite Pro) to contractors and developers to ensure traceability. We created an enterprise system architecture [6] that is driving the remediation, renovation, and replacement of USCP information systems [13, 14].

There are several benefits to this approach:

- First, having specific architectural representations in UML leads to consideration of architectural principles and views. Architectural views are explicitly described with their elements and relationships are explicitly described. The user of the product does not have to make mental translations from standard UML constructs to the architectural views. This will avoid loss of information that might occur during the translation process.

- Second, we can ensure that business needs and processes are directly reflected in the architectural views. We map business needs and processes down through the architectural views to make sure that they are fully represented and supported by the appropriate architectural elements. And, we map architectural elements up through the architectural views to ensure that every entity supports some business need or process.

- Third, we now have a means within our methodology for enforcing consideration of cross-view consistency and description through explicit modeling mechanisms. Using the new cross-checking mechanisms, we are able to ensure that descriptive elements are

IEEE COMPUTER SOCIETY

not overlooked in developing an enterprise architecture.

- Fourth, we address the cognitive aspects of validating a complex set of views with explicit traceability and interface definition. The complex set of roles are addressed by actor stereotypes of both internal and external entities.

We are continuing to develop our enterprise architecture ontology with additional elements using UML 1.4 stereotyping and profiling mechanisms. We are developing extensions to UML for the work, Information and Technical Infrastructure views well as the business and functional views discussed in this paper.

UML currently does not provide a mechanism for modeling the content of interactions between actors. We are investigating agent markup languages (specifically DAML [17]), which are themselves extensions of XML, as a mechanism for modeling interaction content. DAML provides a descriptive mechanism for syntactic and semantic information. This will also enable us to develop tools in the future that analyze properties of systems described through DAML. We have also begun to investigate a better mechanism for modeling the dynamic properties of business systems through further extensions to UML.

References

[1] Armour, F., Kaisler, S., and S. Liu. 1999. A Big Picture Look at Enterprise Architectures. IEEE IT Pro 1(1): 35-42

[2] Armour, F., Kaisler, S., and S. Liu. 1999. Building an Enterprise Architecture Step-by-Step. IEEE IT Pro 1(3):49-57

[3] Armour, F. and Kaisler, S. 2001. Building an Enterprise Architecture Step by Step, Part 2: Agile Transition and Implementation. IEEE IT Pro, Nov/Dec 2001.

[4] Armour, F. and G. Miller. 2001. *Advanced Use Case Modeling: Software Systems*, Addison-Wesley, Reading, MA

[5] Booch, Grady, James Rumbaugh, and Ivar Jacobson. 1999. *The Unified* Modeling Language User Guide, Addison-Wesley, Reading, MA

[6] Business Systems Modernization Office. 2002. USCP Enterprise Systems Architecture (Draft), U.S. Capitol Police, Washington, DC

[7] Jacobson, I., M. Christerson, P. Jonsson, and G. Overgaard. 1992. *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, Reading, MA.

[8] Jacobson, I., M. Ericsson and A. Jacobson 1995. *The Object Advantage: Business Process Reengineeering With Object Technology,* Addison-Wesley, Reading, MA.

[9] Open Group Consortium. 1996. X/Open Architectural Framework. http://www.opengroup.org/public/arch

[10] Booch, G., J. Rumbaugh, and I. Jacobson. 1998. *The Unified Modeling Language Reference Manual*, Addison-Wesley, Reading, MA.

[11] Sowa, J. and Zachman, J. 1992. "Extending and Formalizing the Framework for Information Systems Architecture," IBM Systems Journal, 31(3):590-616.

[12] OMG Unified Modeling Language Specification, version 1.3, June 1999http://www.omg.org

[13] US Capitol Police. 1999. Strategic Plan, Washington, DC

[14] US Capitol Police. 2000. IT Strategic Plan, Washington, DC

[15] US Capitol Police. 2000. Technology Assessment, Washington, DC

[16] Zachman, J. 1987. "A Framework for Information Systems Architecture." IBM Systems Journal, Vol. 26, no. 3, IBM Publication G321-5298.

[17] DAML website, http://www.daml.org

Acknowledgements

*We would like to acknowledge the work performed on the OPSCO project, HSBC London, UK 2001 in which several of the use case stereotyped actor concepts and icons represented in this paper were created. We specifically would like to thank: Maz Al-Shaikh-Ali, Charles Edwards, Giorgio L Mazza, and Luca Vetti Tagliati.

IEEE COMPUTER SOCIETY