White Paper: Using the Unified Modeling Language for Enterprise Architecture

Some believe, incorrectly, that the Unified Modeling Language (UML) is not a useful tool to perform enterprise analysis for the purpose of an enterprise architecture (EA) practice. Although I agree that, the UML is not particularly easy, once mastered, it offers many scalable analysis methods to communicate information to others and recognize strategy, which is an import aspect of a good EA. Through this paper, I'm going to explain some of the important processes an architect might use to achieve a successful UML EA practice.

Before I get started, let me explain my perspective on EA. Largely, I believe EA is mostly about aligning our information systems to accomplish the business goals of the enterprise. Many argue that this is not an important principle of EA. After all, this is the historic reason of the EA practice (i.e., the reason that EA came into existence). Whether you believe this premise or not, this is the viewpoint of this paper. When I say align, I mean synergize or optimize the people, process and systems to accomplish the business needs (goals) of the enterprise to accomplish the desired effects or important outcomes. I characterize outcomes in the form of [capabilities](http://austinea.org/definition/default.html#capability) – sometimes referred to as business [services](http://austinea.org/definition/default.html#service). By the way, throughout this paper I hyperlink you to various definitions to enhance our mutual understanding on the topic.

To me, capabilities are the highest level of the architecture, the purpose of the architecture and the foundational outcome the architecture is trying to improve. Most organizations have many, many capabilities and the architecture staff is unable to accomplish everything all at once; therefore, through some form of governance, the organization needs a way to prioritize architecture engagements is the most effective way.

|  |
| --- |
| C:\Users\byrdr.COACD\Desktop\EssentialCapabilities.jpg  Capability Area Architecture |

I recommend that the architecture employ a UML profile extension for enterprise architecture. These extensions, which exist for most popular UML tools, are available based on the foundational work by the Department of Defense (DoD). If you're a civilian or commercial organization, don’t let the DoD source concern you, because EA is EA no matter how you slice it or dice it. The DoD has exactly the same challenges as industry – aligning their information systems to the desired outcomes of the enterprise. In their situation, exercising military power throughout the globe.

The DoD wisely recognized that their DoD Architecture Framework (DoDAF) provided great guidance and definition of a good EA practice, but lacked a definitive method of execution. Not only a method, but also a standard method to allow EA shared information among dissimilar tools. To accomplish this endeavor, the DoD commissioned the Object Management Group (OMG) to accomplish a groundbreaking feet – standardize a profile to ensure consistency of architectures throughout the DoD including the Ministry of Defence (MOD) a significant warfighting partner. As a result, the OMG, in collaboration with various tool vendors, established the [Unified Profile for DoDAF and MODAF](http://www.omg.org/updm/index.htm) (UPDM). The profile combines the UML, System Modeling Language (SysML) and the Service Oriented Modeling Language (SOAML) into a unified approach to EA service delivery. Meaning, "one" framework can accomplish the enterprise analysis while delivering comprehensive business needs to develop effective solution architectures. How cool is that!

This paper is not going to discuss the entire end-to-end solution, because this would require a book or even multiple books. Therefore, I'm going to focus my attention on the enterprise architecture piece, which employs only the UML portion of the framework. This paper is not about frameworks; rather, this paper discusses "how" one uses the UML to create a successful enterprise architecture practice to implement your framework. I suspect this approach would work for TOGAF as well – I'd be surprised if it didn't. By the way, I recognize that EA is more than your UML models. There are other elements of a successful EA practice such as a Technical Reference Model (TRM), Application Portfolio, etc. that requires different models or tools for which the UML is not well suited. I'm only going to address the UML portion of enterprise modeling for the purpose of analysis and design.

The UML is an object-oriented language and employs similar principles used by software developers; however, UPDM abstracts these principles to the business layer of the enterprise. These models largely capture the business needs of the enterprise in the form of UML use case specifications – the foundational element of the UML. By the way, there's a good book – one that I recommend. It's titled [UML 2 For Dummies](http://www.amazon.com/UML-Dummies-Michael-Jesse-Chonoles/dp/0764526146) and it's really not for dummies! However, the author explains the UML language using understandable terms and uses good examples the non-UML person can understand. I recommend this book to all my mentee architects and it has helped immensely. So let's get started.

Where does the architect begin? I usually begin by building what I call [Capability Area Architectures](http://austinea.org/definition/default.html#CAA) (see figure above). Capability Area Architectures (CAAs) provide the architect with an understanding of what the enterprise is all about, the important outcomes the enterprise is trying to accomplish, and the goals that organizations of the enterprise are trying to achieve. The UPDM includes vision and mission stereotypes as well, which I find useful in the architecture-scoping phase. The architect facilitates this conversation and the collaboration activity is enlightening, informative and valuable in providing the architect a comprehensive understanding the business stakeholder challenges and needs. The exercise of creating these models provides the architect a good picture of the overall enterprise challenges. You don’t immediately build the CAA as shown, which provides a comprehensive enterprise picture; rather, build individual models for each organization and assemble them when you have collected representative CAAs for each business entity with a specific mission.

|  |
| --- |
| C:\Users\byrdr.COACD\Documents\austinea.org\arch\realEstate\realEstateVision.jpg  City Real Estate Office |

Click on the diagram to the left to enlarge for a more readable view - I'll use this diagram for further discussion. This is the highest level of the model that the architect uses to understand the important outcomes the business domain you are modeling. By the way, when I say highest, I mean the organization's viewpoint and the goods and service they're trying to accomplish. This becomes a building block for a more comprehensive understanding that I'll explain later leading to the CAA. A trick I sometimes use to facilitate the collaboration is to start with a "black box" view of the organization to facilitate the stakeholder understanding of their customer environment. I'll have business experts explain who their business stakeholders are (i.e., business entities whom they deliver services) and begin to identify important outcomes the business is trying to accomplish.

|  |
| --- |
| **Select image to read the full report…**  C:\Users\byrdr.COACD\Documents\austinea.org\arch\ctm\eim\customerFacing.jpg  Value Proposition Viewpoint |

I'm sorry, I don’t have an example of the Real Estate Office to tie these ideas together, because a skilled architect will not always use this technique – for example, I jumped strait to the CAA for real estate because they were able to understand the capability idea up front. This is an important skill the architect must achieve – the ability to know when certain EA methods facilitate your particular audience. The example on the right is an example of one I used to help our Enterprise Infrastructure Management group define their "value proposition" to improve their customer service relationship. Again, this is not the highest level of the architecture; however, it's the highest level for Enterprise Infrastructure Management. I call these diagrams the Value Proposition Viewpoint because they emphasize the organization's value to the stakeholders they service. The idea is to use this customer-focused viewpoint to realign the organization's service delivery model. To learn more about this concept, follow the link to the [Value Proposition Viewpoint](file:///C:\Users\byrdr.COACD\Documents\austinea.org\arch\ctm\eim\eIMUpdate131107.html). Next, I'll explain a similar approach at a more enterprise level.

Most organizations need more architecture than the available skilled architecture staff can provide; therefore, the organization must choose their architecture engagements wisely. CAAs are no-brainers - these you must do. Depending on the complexity and size of your organization, CAAs can take a very long time – mostly because it's difficult to get all the business users together to perform the collaboration and the architect usually only get a couple hours of their time each week. Depending on the kinds of goods and services your organization is responsible for, you don’t require a complete (total) set of CAAs to begin to recognize the goal "patterns" the enterprise is trying to accomplish. The CAA example above only provides CAAs for seven departments and one from the CIO Council, which I show as an example to align IT goals to the enterprise business needs. The view, although not complete, is enough to observe the patterns of goals of the enterprise. What I'm referring to are shared [goals](http://austinea.org/definition/default.html#goal) – goals that are common or similar in the enterprise whether they're information technology related or not. I tend to emphasize information technology (IT) related goals and let the departments concern themselves with non-IT goals once identified. Through analysis using the "human computer" called the brain, the architect applies their information technology knowledge, skills and abilities to abstract a set of Common Capabilities to accomplish the goals of the enterprise.

|  |
| --- |
| C:\Users\byrdr.COACD\Desktop\EssentialCapabilities.jpg |

This is a skill you're looking for in a good enterprise architect – the ability to see "the forest through the trees." According to Wikipedia, important skills and knowledge include:

* Systems thinking - the ability to see how parts interact with the whole (big picture thinking)
* Knowledge of the business for which the enterprise architecture is being developed
* Interpersonal and leadership skills - servant leadership, collaboration, facilitation, and negotiation skills
* Communication skills, both written and spoken
* Ability to explain complex technical issues in a way that non-technical people may understand
* Knowledge of IT governance and operations
* Comprehensive knowledge of systems engineering

After more than 14 years of doing EA, I'm in complete agreement with Wikipedia's skills assessment, because your enterprise architect needs to be a strategic and abstract thinker. In fact, that's why good enterprise architects are hard to find. Most technically skilled individuals are not typically people who pursue quality management, process improvement, and strategy development activities. Because most IT folks specialize in the solution space, they rarely move away from this comfort zone – except to possibly become a CIO, which I believe is a leadership role more than a technical role and EA is a technical role. This is why most organizations align their enterprise architects with the CIO. I like Wiki's reference to [servant leadership](http://en.wikipedia.org/wiki/Servant_leadership), because and enterprise architect rarely has any official authority; rather, they are influencers, they lead by example, and become sought after for the value of their service. This means the enterprise architect needs good people skills. Not to mention the fact that your Chief Enterprise Architect needs great marketing and advocacy skills, otherwise there is no hope for your EA! I'm often reminded of Robin Williams in his role as the android Andrew in the film [Bicentennial Man](http://en.wikipedia.org/wiki/Bicentennial_Man_(film)), who frequently states, "One is glad to be of service."

If you study the CAA diagram above (select to enlarge), under the Transportation department capability model, you'll notice that the goal *Greater ability perform analysis* has a large concentration of arrows (UML dependencies) on it. This indicates that this goal is extremely important to Transportation. Again, if you look closely, you'll observe that this goal "depends on" the goal to *Obtain an electronic review solution*. This goal contributes to Transportation's capability to strategically plan, regulate right of way, and regulate special events. Further study of the model reveals that this same goal is required by Planning and Development Review and Public Works – later we've learned that other departments share this goal as well. Therefore, this is a significant and shared goal! Even though this goal doesn’t have a large concentration on it, the goal is important to provide the meta-data necessary for *Greater ability to perform analysis* because it provides the meta-data needed for their research to find relevant plans for future analysis. Currently, building and land use plans (blueprints) are coordinated with the city on paper, marked up by hand, and manually entered into our information systems for Submittal Applicant feedback regarding regulatory compliance for resubmittal. Using the CAA model, it became clear to city leadership that we need to fix this problem and pursue the *Obtain an electronic review solution* goal.

If you further inspect the CAA, you'll observe that the *Obtain an electronic review solution* goal maps to the Manage Case Files common capability. It's not clear in the diagram (too many traces), however, our UML tool provides layering to allow the architect to turn off other common capability layers to focus on the trace under study. Typically, the *Manage Case Files* service manager is responsible to manage a solution. This is where traditional roles become a bit blurred – in a typical organization the requirements collection is the responsibility of the service manager. However, if you have a good EA program, your EA staff uses UML use cases to develop the requirements.

|  |
| --- |
| http://austinea.org/arch/essentialcapabilities1.jpg  UML Use Case Model |

The UML Use Case Model is the heavy lifting part of any good EA program. These models document the people, the process, the business rules, the information flow, and the required dialog between roles and systems to achieve the use case result of value. In fact, when I train new architects this is where I begin – I teach them to develop quality use case models. As they gain experience, later in their career I introduce strategic viewpoints such as the CAA. As an enterprise architect, I don’t teach them the traditional approach in use case development, rather I've modified the technique based on my understanding of the UPDM method. Fundamentally, the approach is the same – i.e., results of value (ROV) are still important. What's different about an EA use case model is that we don’t show dialog between system and software components using the traditional UML approach from which the UML was derived. We abstract our use case models up a notch and use the capability as the "subject" system.

If you select to enlarge the diagram on the left – *UML Use Case Model*, you'll see the use case in the upper left corner. We create our uses case models from the business user viewpoint without dictating a solution. In fact, I sometimes refer to these use case views as the user experience viewpoint. Based on our identified goal to *Obtain electronic review solution*, further analysis of the CAA revealed that the primary capability to exercise the process is the *Perform Regulatory Review* capability – also shared by other city departments. As a knit point, let me deviate my thought a bit and explore an interesting topic.

Typically, capabilities are class representations of process. They act as process containers (people, process and systems) and represent a "type" of process – meaning that capabilities typically have more than one process that exercises the capability. In my example, this capability has just one process – i.e., *Perform Regulatory Review*. So technically, the capability is not actually a capability at all; rather it's a process – this happens all the time because the architect is not always familiar with the business domain being modeled, but that's okay. Later as your model matures, you explore other maturing processes and begin to map your use case models to a new, redefined capability – the new capability notionally might change to *Obtain Certificate of Occupancy*, because the purpose of the processes lead to the Submittal Applicant value item to legally occupy the building. This is the iterative nature of the process (i.e., you architect a little, learn a lot, architect some more) leading to a well-defined understanding of your enterprise. This is another distinct value of object-oriented modeling. Changes are easy – even big changes! Change it once and it changes everywhere – you gota' like that! I do… The process is quite revealing to both the architect and the business user. I've had business customer's thank me for helping them think through their value proposition – let alone the value of the use case itself.

The use case diagram's primary purpose is as a scoping mechanism (shown in upper left hand corner of the *UML Use Case Model*). Using this model, you collaborate to understand the various roles, systems requiring some form of interface and develop the Use Case Scope. Again, the process is iterative. You might construct an incomplete use case diagram and begin developing your activity view and your sequence diagram to explore the required interfaces to document the process. This may lead to newly discovered roles and a better-defined scope. The activity model captures the business rules in the form of an interaction overview. Notice that the activity model (interaction overview - right corner of diagram) is a little different from the traditional activity diagrams typically developed by business systems analysts. First, they include objects (information elements shown in green) and show their flow through the process. Additionally, they capture the high level business rules – notice that the activity model does not use the traditional (yes/no) at decision points; rather, they explicitly identify the condition or state as objects move through the process. You'll also note that the activities on the interaction overview expand to expose the sequence diagram view (i.e., each activity has its own sequence diagram – hence the term interaction overview, because they convey the interaction of sequence diagrams). In the UML tool, the architect clicks on the activity and the sequence diagram automatically opens. This approach is using an information-hiding concept in that each "layer" reveals more and more detail until you begin to describe the user experience as they use the capability as a subject – in this case, *Perform Regulatory Review*.

Some feel strongly that this is a system or solution architecture – it is not! It's the business user viewpoint. I believe this confusion comes from the fact that we identify system interface needs - in our example you'll see an interface to our electronic document imaging and management system (EDIMS). This is simply the business user saying that they wish to store the relevant document (object) in EDIMS, but it doesn't say how! The how should be accomplished by the solution provider. We accomplish sequence diagrams by examining the capability as a "black Box" – the subject system. If you further inspect the diagram, you also notice that each message (lines with arrows and objects) on the sequence diagram has a description in a table. In the model, this description is contained in the documentation of the message (i.e., the operation of the class Perform Regulatory Review – stereotyped as capability). This is where the business user explains how they wish to use the system. We have our business users explain the use as the "system" instead of the capability name. Because of the iterative nature of the process, you may later change the name of your capability as I described earlier. If you used the capability name in the specifications, you would need to edit all of your capability descriptions if it changes. So it's best to just say the user uses "the system to…bla, bla" in your specifications. Look at the *performIntake* message on the *1. Perform Intake* sequence diagram. There you'll see a description (i.e., functional need) on how the business user expects the wishes the dialog between the role (i.e., Submittal Applicant) and the subject system (i.e., capability) to occur.

In the beginning, you'll discover that the business user is a little confused on how the process works, because most business users have never experienced this process. However, trust me, they catch on quite quickly and you'll observe them in open collaboration! The communication line opens and they debate back and forth – I sometimes just sit back and enjoy the ride until they settle on a conclusion and they always do. When you have been doing this a long time, it's very rewarding to watch the business user reach this state, because it means they took ownership of the process (and they should).