Enterprise Architecture for Architecture Driven Planning

Abstract

In today's rapidly changing, technology-centric world, companies are seeking to become more agile with their development cycles. However, developing applications faster is only one of the challenges facing IT organizations. Companies whose development effort is spread across multiple sub domains with teams that rarely speak to each other must have an in depth understanding of their systems in order to make agility possible. Even more important is leveraging this body of knowledge for tactical guidance of project development efforts and for strategic planning purposes.

No matter what architecture guidelines you apply or development methodology you practice, without a systematic, well-thought out approach to capturing the information needed about the business and IT systems, you are doomed to, at best, unsatisfactory results and, at worst, failure. Unfortunately, it seems the odds are stacked against you for numerous reasons.

In this paper, I will discuss the concepts of Enterprise Architecture and how it can be used for what I call Architecture Driven Planning. The different architecture views representing the various stakeholders will be explored along with a means to capture this information in a digestible form. This non-proprietary approach, which is the culmination of applying concepts of several architecture standards over numerous projects, is based on UML that can be adapted to your organization's needs rather than forcing a particular method. Any modeling tool that complies with UML 2.0, provides UML's extension mechanisms, and can structure all of your models into one seamless view can be used. Examples of some of the concepts will be shown using Sparx System's Enterprise Architect, a UML modeling tool that fits all of these requirements.

Introduction

In order to make the right decision, you must have the right information. This is very true for IT organizations. Without the right information you are doomed to rebuild rather than reuse; to have varying approaches to development, causing confusion; to miss opportunities because the connection between the business and IT is not well understood. For example,

- As a developer, do you know what technologies have been adopted and which are being eliminated within your company?
- Do you know what business services can be used for your project and what constraints are placed on using them?
- Do you know how to use the integration products to reach out to other domains?
- As an IT manager, do you know which applications are at risk due to lack of vendor support or to dependence on obsolete technologies?
- Do you know what impact changing one component will have on other components dependent upon it?
- As a portfolio manager, do you know how many applications are performing the same business functions, resulting in unnecessary licensing fees?
- As a DBA, do you know how many databases could be combined into a single instance, also reducing licensing fees?
- As a CIO, do you know the cost benefit and eventual ROI, if any, of replacing a current technology base with a promising new one?

- Do you understand the technical implications of providing a new service or product to your customers?
- As a CEO, do you understand how new technologies could improve your business processes or offer new ones?
- As an employee, do you know what the guiding principles of your company are so you can ensure you apply them in everything you do?

In today's environment, business and IT are completely interwoven and the information necessary to drive both is critical to a company's survival. Capturing that information is only half the problem. Putting it into digestible form and making it readily available is the other half. This is the role of Enterprise Architecture.

There are numerous definitions of architecture[1] in general and of Enterprise Architecture in particular. Practically every standards body and every tool vendor in the space have created their own definition. However, what is often neglected is the importance of architecture as an effective communication device. In the hands of the right people and the right tool, an architecture description can provide an Architecture Driven Planning approach to future development efforts. With it, you can create a project portfolio that will realize a return on your IT investment by addressing business needs and IT risks over a series of projects.

For an Architecture Description to successfully support Architecture Driven Planning, it must realize the following goals:

- *The Architecture Description must provide the guiding principles and objectives of the business and IT*

  Many organizations spend a lot of time defining what the guiding principles of the business are, the things that should be a conscience part of everything that is done every day. The same can be said for the principles that guide the effort of automating much of the businesses processes. But how many organizations can truly say that those principles are readily available to all employees who should be putting those principles into practice?
- *The Architecture Description must speak to all stakeholders*

  There is a diverse set of stakeholders for a development project. These include:
  - Senior business managers who develop the company vision which the IT systems are to support
  - Business people who use the systems to conduct daily work
  - Auditing and compliance personnel responsible for ensuring the systems obey all internal and external mandates
  - Senior IT managers who have the vision of how technology can support business needs
  - Senior architects responsible for realizing that vision and ensuring compliance to it

[1] When talking about architecture, architecture standards such as TOGAF formally differentiate between architecture (what you have or want to have) and architecture description (a formal description of what you have, usually including visual models and textual documentation). However, they informally blur the distinction, leaving it to context to determine which is being addressed. In this paper, I will do the same.

- Development teams who need to know the architecture standards to which they should design their solutions
- Information architects who need to know the structure, constraints, and distribution needs of the data
- Database architects and infrastructure personnel who need to determine capacity and throughput requirements
- Quality assurance personnel who need to ensure the systems meet functional and non-functional requirements
- Operations staff who need to monitor the deployed systems and ensure they stay up and running
- Security Analysts who need to ensure the information and processing is protected

In order to address the needs of each stakeholder, we must separate their concerns. Talking to business people about the benefits of .net vs. java is meaningless. Likewise, a network engineer is not interested in whether an application is developed in C++, C#, Java, or Assembler. Therefore, the concerns of each stakeholder must be captured in separate areas of the Enterprise Architecture model that are dedicated to the language of the stakeholders while linked to all of the other areas of the model. This provides the clarity and coordination that will lead to the type of information that can be used for tactical project level needs and strategic business and IT planning.

- *The Architecture Description must provide an inventory of what is currently available*

  There are numerous software development life cycle (SDLC) methodologies that have quite different approaches. Some, like RUP, are document-centric while others, like Agile, are people and task-centric, with several flavors in between. However, none of them can provide success without sufficient knowledge of what already exists to be leveraged. Otherwise, you are doomed to rebuild things that are already there. One of the problems in this area has been that different tools have been used for different parts of the SDLC. Business models are kept in one tool. Design artifacts are captured and maintained in another tool. An inventory of reusable components is kept in a third tool, and publishing the results of these tools lies with yet another tool. Being able to capture all of these in a single tool, along with links to additional information maintained in external files, provides an incredible boon to productivity.

- *The Architecture Description must show the preferred and the acceptable ways to build, integrate, and use software and to configure hardware*

  The Architecture Description should indicate what software products should be used for business and technical solutions. It should provide patterns of how to build things as well as real examples of how they were built in other projects. It should distinguish between the preferred methods and the acceptable methods, providing the reasons why one should be used over the other.

- *The Architecture Description must provide a road map to the future*

  The Architecture Description should show how to use new technology and approaches that are being introduced to the company. This should include patterns and real examples. The planning process should allow for the introduction of new methods and technologies over a series of projects in an iterative, incremental manner.

- *The Architecture Description must be readily available*

  None of the above bulleted items will provide any benefit unless they are readily available to all stakeholders and project members. I have worked with large organizations that are dedicated to improving the development process. However, even though they were all using the same modeling tool, that tool used disjoint model files as its repository. This made it extremely difficult to create a single library of models and even more difficult to peruse the models. Addressing everything within one tool with a single repository that can publish the results on a corporate intranet will turn what may have formerly been considered an expense into a valuable corporate asset.

Over the years, I have extended numerous modeling tools for the purpose of capturing my clients' architecture. In the beginning, the focus was on standardizing and improving development efforts across a large number of project teams. More recently, the purpose has been expanded to include a broader range of information necessary to address Architecture Driven Planning while providing a feedback loop between the broader aspects of Enterprise Architecture and the underlying necessities of project development.

Some of the tools I have used are good at business modeling, others are good at data modeling, others at low level design, and yet others at modeling the data center. However, it is crucial to have the relationships among the artifacts in the various architecture views within the same repository in order to glean strategic information from them.

Some tools are based on proprietary toolsets and approaches. This provides short-term gain when initiating an Enterprise Architecture effort by leading you down a prescribed path. Eventually, you may find that path to be restrictive and your dependence on the proprietary nature of the product problematic as you try to grow your user base with that proprietary knowledge.

Other tools are based on industry standards and are more general in approach. Within this group fall the various UML tools. I am often asked whether something as general as UML can be used effectively to capture something as diverse as Enterprise Architecture? In this report, I will explain how a modeling tool that supports UML version 2.0 can be used very effectively for this purpose.

In particular, I have found Sparx Systems' Enterprise Architect (herein referred to as Sparx EA) to be well suited for the task. Its adherence to industry standards make it immediately usable to a wide-range of architects and designers. Its implementation of UML's extension mechanisms allow it to be adapted to specific audiences when addressing the concerns of different stakeholders. These extensions can make the diagrams more comprehendible to the different stakeholders by speaking in their own language. They can also be used to make conforming to your architecture principles easier than not doing so. Being repository based, information can be re-used across the modeling effort and can be extracted into meaningful reports.

In the following sections I will discuss Enterprise Architecture, the architecture views needed for an enterprise with a distributed business information system, how to model those views with UML, and how to effectively share the information captured in your Enterprise Architecture.

Starting an Enterprise Architecture Modeling Effort

Many organizations start a program to model their Enterprise Architecture with the belief that it is the right thing to do. They make a big announcement, introduce the team, and make big predictions about

how the Enterprise Architecture model will fix all their ills. Then, with the spotlight on them and with no specific requirements to satisfy, the team wanders forward, meeting no specific deadlines and eventually losing favor and funding.

> When starting an effort to model your architecture, have specific goals and requirements that will yield business value, take an iterative, incremental approach, and do not start off with a lot of fanfare!

Modeling your Enterprise Architecture should not be done for its own sake. You should have specific goals and requirements for doing so and an iterative plan to develop the architecture. Progress should be measured against those goals and business value should be obtained along the way or, as mentioned, interest will wane. Starting a project to model the Enterprise Architecture should be the same as any other project. The business case should be made with well-defined, measureable requirements that can be prioritized and addressed in an iterative, incremental manner.

On one such project for an IT department that did more integration of third party applications than software development, we started with the following set of goals to reduce cost and risk.

1. Determine what applications were performing duplicate functionality in order to eliminate as many as possible and, therefore, reduce licensing fees.
2. Determine what applications were dependent on each other in order to reduce problems when replacing or updating a given application.
3. Determine what applications were at risk due to a lack of support. This meant more than just finding out what applications were no longer supported by their vendors. It also meant mapping the applications to the technologies they required and determining which technologies were no longer supported.
4. Determine the latest version of the DBMS each application required in order to migrate the applications to as few versions as possible, thus reducing licensing fees.
5. Determine how many databases running on separate instances of the DBMS could be combined to a single instance, again reducing licensing fees.
6. Determine what applications could be hosted on the same servers in order to reduce the number of servers in the data center and reduce the associated licensing fees.

For another client, the focus was development oriented. Here the initial goals focused on improving the development effort.

1. Provide guidance on and ensure conformance to the architecture guidelines and principles that were to lead the company forward into standardized approaches and new technologies.
2. Ensure that the design models and documentation produced by the application architects across the various domains of the company were produced consistently and addressed all the needs of the development team and its extended members. This included providing modeling frameworks, patterns and report templates to help automate the job of the architects.

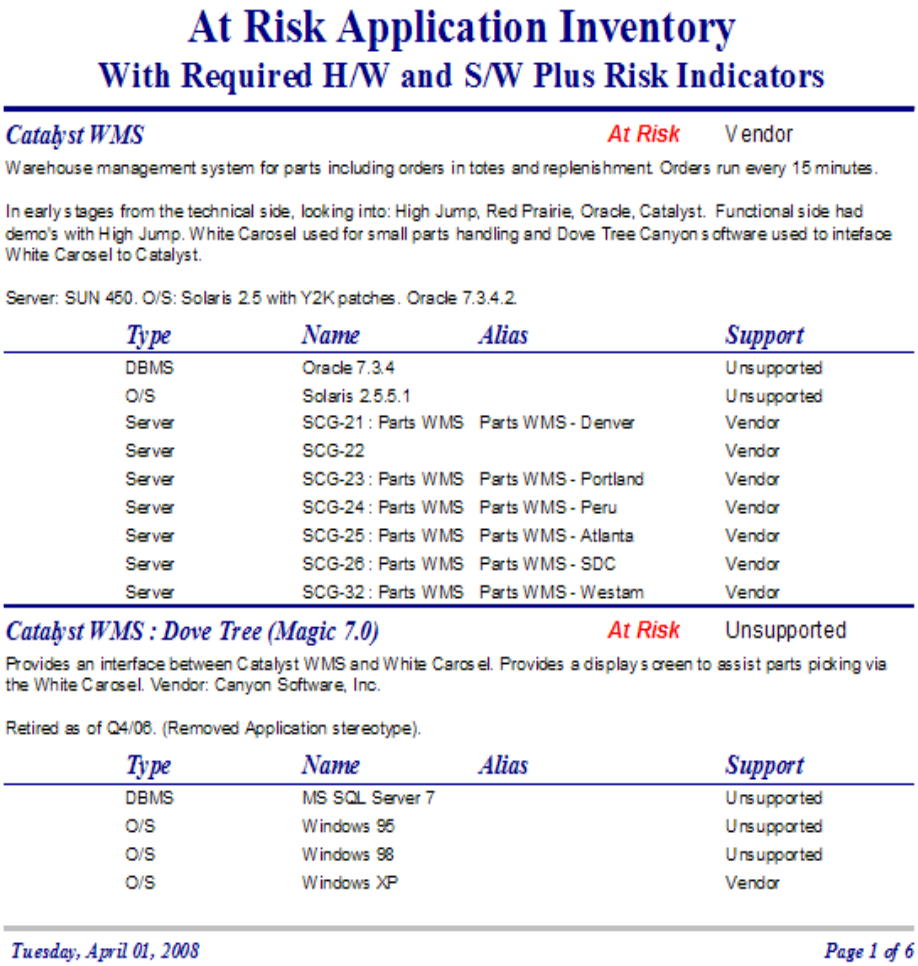
The main focus for yet another client was to:

1. Capture an inventory of IT assets and make them readily available to the development teams.
2. Reduce the number of assets to increase maintainability. For example, SAP was one of the main products used in the IT shop, but there were hundreds of small applications that had been

developed that communicated with SAP. Most of them had their own interface to SAP, even though they often got the same information as the other programs. By capturing all of the interfaces in the architecture model, the number of interfaces was eventually reduced 10-fold.

Each of these projects proved successful because they started small and, after their initial success, were able to expand the scope of their modeling efforts.

On the first project above, we concentrated on the Application Architecture, Technology Architecture, and Deployment Architecture Views (more on these later). The information this provided led to a multi-year plan to reach their goals of reduced cost and risk, which yielded a significant savings after the first year. A year later, when the company decided to change hosting companies for their servers, we were able to use the information to help plan the move. Because we knew which applications were dependent on each other and on which servers the applications were deployed, by extension we knew which servers were logically dependent on each other and had to be moved as group. Brief excerpts from some sample reports from this project are included below.



At Risk Application Inventory

With Required H/W and S/W Plus Risk Indicators

Catalyst WMS — *At Risk* — Vendor

Warehouse management system for parts including orders in totes and replenishment. Orders run every 15 minutes.

In early stages from the technical side, looking into: High Jump, Red Prairie, Oracle, Catalyst. Functional side had demo's with High Jump. White Carosel used for small parts handling and Dove Tree Canyon software used to inteface White Carosel to Catalyst.

Server: SUN 450. O/S: Solaris 2.5 with Y2K patches. Oracle 7.3.4.2.

Type	Name	Alias	Support
DBMS	Oracle 7.3.4		Unsupported
O/S	Solaris 2.5.5.1		Unsupported
Server	SCG-21 : Parts WMS	Parts WMS - Denver	Vendor
Server	SCG-22		Vendor
Server	SCG-23 : Parts WMS	Parts WMS - Portland	Vendor
Server	SCG-24 : Parts WMS	Parts WMS - Peru	Vendor
Server	SCG-25 : Parts WMS	Parts WMS - Atlanta	Vendor
Server	SCG-26 : Parts WMS	Parts WMS - SDC	Vendor
Server	SCG-32 : Parts WMS	Parts WMS - Westam	Vendor

Catalyst WMS : Dove Tree (Magic 7.0) — *At Risk* — Unsupported

Provides an interface between Catalyst WMS and White Carosel. Provides a display screen to assist parts picking via the White Carosel. Vendor: Canyon Software, Inc.

Retired as of Q4/06. (Removed Application stereotype).

Type	Name	Alias	Support
DBMS	MS SQL Server 7		Unsupported
O/S	Windows 95		Unsupported
O/S	Windows 98		Unsupported
O/S	Windows XP		Vendor

Tuesday, April 01, 2008 *Page 1 of 6*

Deployed Server Configuration Items

Server Name	*Item Type*	*Item Name*	*Item Support*
	Application	SQR Launcher	Internal

SCG-49 : Oracle Parts — *Site:* Managed Hosting Provider
Server Model: SUN E6800 — *Zone:* Internal — *Support:* Vendor

	Application	Business Management (BM)	Internal
	Application	EbAdmin	Internal
	Application	Fulfillment	Internal
	Application	Transportation Mgt System [TMS]	Unsupported
	Application	Visibility	Unsupported
	Database	CDSDSS : holos	Unknown
	Database	Holos_Security : holos	Unknown
	Database	iceprd (standby) : ice_station	Unknown
	Database	lmsprod : lms	Unknown
	Database	manprod : transport	Unknown
	Database	p2kprod : p2k	Unknown
	Database	tmprod : transport	Unknown
	Database	vcprod : transport	Unknown
	Database	VehicleDSS : holos_veh	Unknown
	Database	VehicleOwner : holos_veh	Unknown
	DBMS	Oracle 8.1	Unsupported
	O/S	Solaris 8	Vendor
	S/W Tool	PERL	Vendor

SCG-50 : NIS Plus / Home Dir / NTP/ FTP — *Site:* Managed Hosting Provider
Server Model: Sun V240 — *Zone:* Internal — *Support:* Vendor

	Application	Business Management (BM)	Internal
	DBMS	Oracle 8.1	Unsupported
	O/S	Solaris 2.6	Unsupported
	O/S	Solaris 2.6	Unsupported

SCG-51 : BKUP NIS Plus / Home Dir / NTP — *Site:* Managed Hosting Provider
Server Model: Sun V240 — *Zone:* Internal — *Support:* Vendor

	S/W Tool	Home Directory Server	Unknown
	S/W Tool	Netscape Directory Server (LDAP)	Unknown

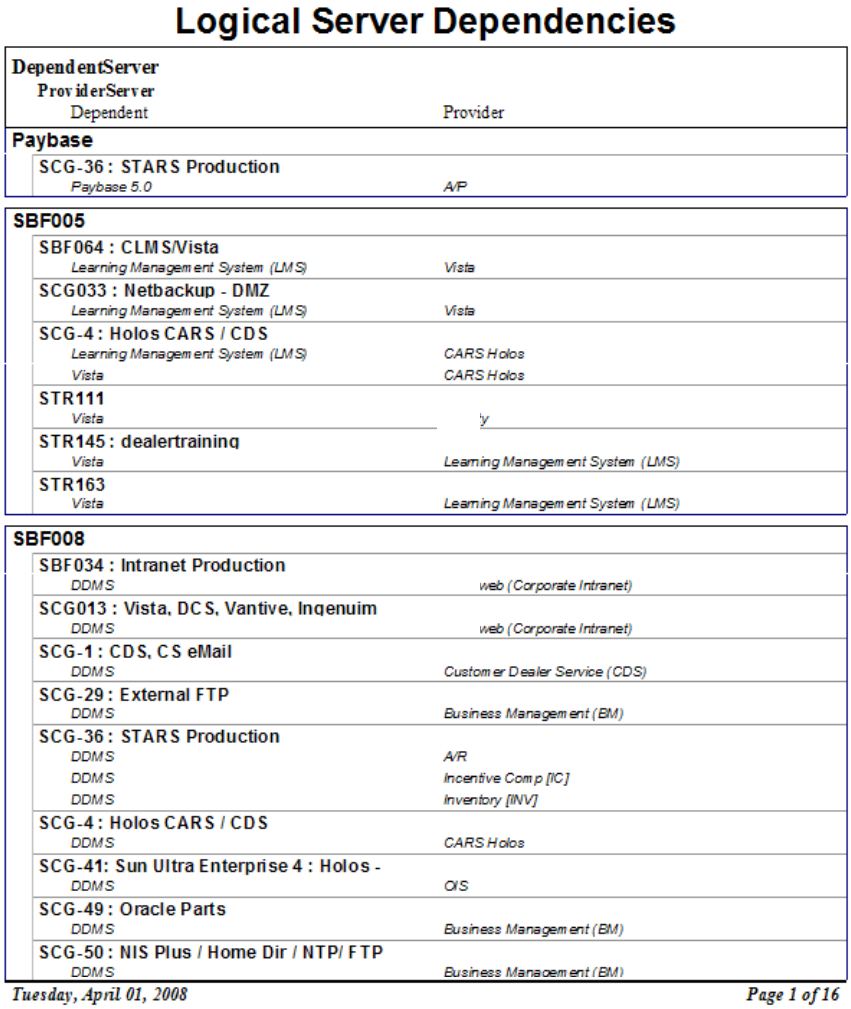
SCG-53 : Xelus Prod — *Site:* Managed Hosting Provider
Server Model: SUN E6800 — *Zone:* Internal — *Support:* Vendor

	Database	rpemprd : red_prairie	Unknown
	Database	rphubprd : red_prairie	Unknown
	Database	rplensp : red_prairie	Unknown
	Database	rplprd : red_prairie	Unknown
	Database	rpmdlprd : red_prairie	Unknown
	Database	rpwprd (standby) : red_prairie	Unknown
	Database	siedbpro : siebel_trans	Unknown
	Database	siedwpro : siebel_whse	Unknown
	Database	xelprod : xelus	Unknown

Logical Business Software Dependencies

Software			Support Status	Software Type	Deployment Status	
Mode	Related Software	Logical Interface	Interface Type	Related S/W Type	Deployment Status	
Barton & Cooney			**N/A**	**External**	**N/A**	
Provides to	Business Management (BM)	iDealer Financial Info	Data Flow	Application	Deployed	
Bell & Howell			**N/A**	**External**	**N/A**	
Provides to	Bell & Howell : Electronic Parts Catalog (EPC)	iParts Updates	Data Flow	Application	Deployed	
Bell & Howell : Electronic Parts Catalog (EPC)			**Vendor**	**Application**	**Deployed**	
Requires from	Bell & Howell	iParts Catalog Updates	Data Flow	External	N/A	
Business Management (BM)			**Internal**	**Application**	**Deployed**	
Provides to	EXAM	iDealer Financial Info	Data Flow	Application	Deployed	
Provides to	Holos : Parts Holos	iDealer Financial Info	Data Flow	Application	Deployed	
Requires from	DDMS	iDealer Info Feed	Data Flow	Application	Deployed	
Requires from	Dealer	iDealer Financial Statements	Data Flow	External	N/A	
Requires from	Holos : CARS Holos	iVehicle Sales, SAS Contracts & C	Data Flow	Application	Deployed	
Requires from	OIS	iSales/Owner Info	Data Flow	Application	To be Retired	2009-07-01
Requires from	SDC	iDealer Financial Statements	Data Flow	External	N/A	
Requires from	SNE	iDealer Financial	Data Flow	External	N/A	
Requires from	SNE	iDealer Financial Statements	Data Flow	External	N/A	
CC Group			**N/A**	**External**	**N/A**	
Provides to	IT Change Request System	iCR Notices & Reports	Data Flow	Application	Deployed	
Chase			**N/A**	**External**	**N/A**	
Provides to	OIS	iChase	Data Flow	Application	To be Retired	2009-07-01
Cobalt			**N/A**	**External**	**N/A**	
Provides to	Holos : CARS Holos	iVehicle & CPO Inventory, Accesori	Data Flow	Application	Deployed	
Creative Print Group			**N/A**	**External**	**N/A**	
Provides to	DDMS	iDealer Info Feed	Data Flow	Application	Deployed	
Customer Dealer Service (CDS)			**Internal**	**Application**	**Retired**	**2007-07-01**
Provides to	Holos : CDS Holos	iDealer, Claim, Warranty data	Data Flow	Application	Deployed	
Provides to	Oracle 11i	???	Data Flow	Software System	Deployed	
Provides to	Oracle 11i : A/P	iPayments	Data Flow	Application	Deployed	
Requires from	DDMS	iDealer Info Feed	Data Flow	Application	Deployed	
Requires from	Holos : CDS Holos	iClaim Summary	Data Flow	Application	Deployed	
Requires from	Holos : Logistics Holos	iTransportation Info	Data Flow	Application	Deployed	
Requires from	Oracle 11i : Install Base [IB]	iModel Info	Data Flow	Application	Deployed	

Tuesday, April 01, 2008 *All lifecycle statuses are included* *Page 1 of 15*



Logical Server Dependencies

DependentServer / ProviderServer	Dependent	Provider
Paybase		
SCG-36 : STARS Production	Paybase 5.0	A/P
SBF005		
SBF064 : CLMS/Vista	Learning Management System (LMS)	Vista
SCG033 : Netbackup - DMZ	Learning Management System (LMS)	Vista
SCG-4 : Holos CARS / CDS	Learning Management System (LMS)	CARS Holos
	Vista	CARS Holos
STR111	Vista	y
STR145 : dealertraining	Vista	Learning Management System (LMS)
STR163	Vista	Learning Management System (LMS)
SBF008		
SBF034 : Intranet Production	DDMS	web (Corporate Intranet)
SCG013 : Vista, DCS, Vantive, Ingenuim	DDMS	web (Corporate Intranet)
SCG-1 : CDS, CS eMail	DDMS	Customer Dealer Service (CDS)
SCG-29 : External FTP	DDMS	Business Management (BM)
SCG-36 : STARS Production	DDMS	A/R
	DDMS	Incentive Comp [IC]
	DDMS	Inventory [INV]
SCG-4 : Holos CARS / CDS	DDMS	CARS Holos
SCG-41: Sun Ultra Enterprise 4 : Holos -	DDMS	OIS
SCG-49 : Oracle Parts	DDMS	Business Management (BM)
SCG-50 : NIS Plus / Home Dir / NTP/ FTP	DDMS	Business Management (BM)

Tuesday, April 01, 2008 *Page 1 of 16*

The result of the work done on capturing the Enterprise Architecture changed how the company planned their projects. Instead of simply reacting to one-off requests from the business, they employed Architecture Driven Planning that allowed them to integrate both business and technology needs into a strategic program.

The success of this project was determined up front by having specific goals and little fanfare. Within a couple of months, we had enough information to start making a strategic difference, which ultimately led to additional funding in order to broaden the scope of the modeling effort to include other architecture views.

Another reason for the level of success was that we were able to capture all the information we required in one tool, namely Sparx EA. This made it easy to see the relationships among the various artifacts. Often, especially within large organizations, different groups use different tools. This can mean that the only way to relate artifacts from the different models is by visual inspection. Having them in the same repository, on the other hand, means these relationships can be explored. It also made all the information readily available via published reports and web pages on the corporate intranet.

Being able to capture all of the views of your Enterprise Architecture in one modeling tool allows you to explore the relationships among the elements of the views and glean strategic information from them.

Modeling Enterprise Architecture

Enterprise Architecture is a means to support the business through the use of IT. Enterprise Architecture modeling is a holistic approach for capturing not just all of the areas within IT, but for capturing the alignment of IT to the business as well. This will help to ensure that IT provides tangible benefit to and keeps pace with the business. Enterprise Architecture modeling provides information necessary to help make tactical and strategic business decisions.

As mentioned above, we must address the concerns of each stakeholder. There are several architecture standards today that prescribe various aspects of capturing Enterprise Architecture. Standards like the Open Group Architecture Framework (TOGAF), the International Standards Organization's Reference Model for Open Distributed Processing (RM-ODP) and the Rational Unified Process (RUP) prescribe a process for capturing your architecture. Each also provides a set of architecture views, but leaves it up to the architecture implementation team to fully determine what they should be.

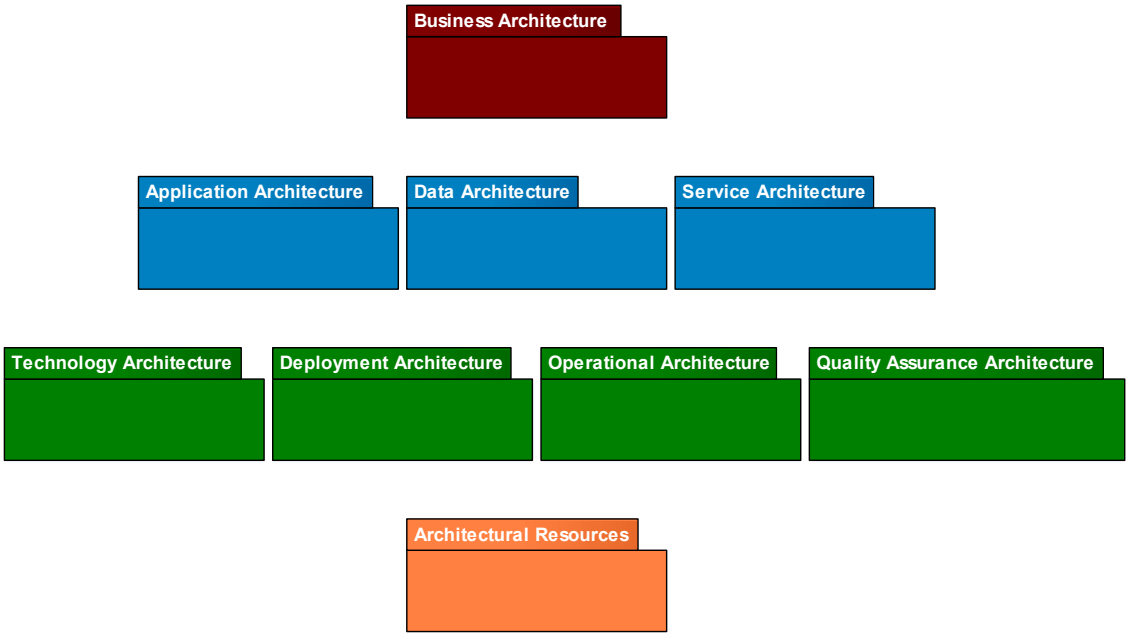
Excerpt from TOGAF – The Book, Section 2.4 Using TOGAF with Other Frameworks:

Because TOGAF is a generic framework, as mentioned above, and intended to be used in a wide variety of environments, it does not prescribe a specific set of deliverables; rather it talks in general terms about the types of deliverable that need to be produced, and focuses instead on the methods by which these should be developed.[i]

Another standard, Model Driven Architecture (MDA) by the Object Management Group (OMG), also provides a separation of concerns, in this case into three levels; the Computational Independent Model that describes the business environment in completely technology free terms, the Platform Independent Model (PIM) that provides a design solution that is not tied to a particular technology base, and a Platform Specific Model (PSM) that puts the solution in terms of the chosen technology base.

After extensive work with numerous large companies in the insurance, financial, pharmaceutical, publishing, and automobile distributorship sectors in adopting the concepts of the above standards, I have found that the following architecture views provide a good representation of the needs of the various stakeholders. They provide a good starting point for my clients that we adapt for their specific needs. These views are for companies that have distributed business information systems. A company with other development modes, e.g., real-time applications, will have a much different structure to their architecture.

Figure 1: Enterprise Architecture Views



Business Architecture View

This is where it should all begin, although it often does not. A requirements-only driven approach eliminates the possibility of confirming whether the IT solutions are really aligned with the current and future state of the business. The following sections comprise the Business Architecture:

Business Principles

Business principles affect all business operations. Everything that is done in the business and IT spaces must adhere to the business principles. The benefit of placing these principles in a location where everyone involved in the business and IT can readily see them can not be understated. Many of the things prescribed here can be found somewhere within the company, but often only with great difficulty. Placing them in the Enterprise Architecture model and then deploying that model on the intranet makes them readily available to all.

Business Objectives

Business objectives provide big picture guidance as to where the business wants to go. They speak to new business initiatives such as new products and services, or new delivery mechanisms, such as an entry into web-based sales.

Business Context

Business context provides a picture of where the enterprise fits in with its world environment. It displays the customers, business partners, regulators, and other external entities that do business with or control some aspect of the business of the enterprise. Business Actors represent people or other business systems that interact with the business. Business Use Cases explain what benefit these external actors expect to get by interacting with the enterprise.

Business Organization

Business Organization shows the departmental structure of the enterprise and the business workers (roles) who accomplish the tasks necessary to fulfill the enterprise's purpose for being in business. The business workers perform the business processes with or without the assistance of automated systems.

Business Processes

A business process is a set of coordinated activities, conducted by people and/or systems that accomplish a specific organizational goal. Business processes are the actions the enterprise must do to conduct its business.

Business Functions

Business Functions represent a decomposition of departmental responsibilities down to the activities that are performed by business workers in the domain. The business workers' activities are strung together along with business events, inputs/outputs, and process execution decision making to perform the functionality of business processes.

Business Information

Business information consists of Business Entities, which represent key informational concepts with which the business people work to perform their jobs, and the relationships among the Business Entities.

Business Policies

Business Policies are sets of related business rules that govern the behavior of the Business Processes and the Business Entities and their relationships. Business systems can be thought of as Business Information that is acted upon by Business Processes which are both controlled by Business Policies. The structure of Business Information is the most stable of the three. Business Processes change more often than does the structure of Business Information. Business Policies are the most volatile of the three. Therefore, it is important to isolate these and to trace them to the IT components that enforce them.

Business Requirements

The Business Requirements section of the Enterprise Architecture Model represents business requirements that are on a “to-do” list, that are assigned to projects for implementation, or have already been satisfied but are still being tracked for a complete picture of current and future states of the business architecture. There are three types of requirements to be captured…

- Business Needs, which capture the high level requirements of the business and are further described by Use Cases,
- Use Cases, which satisfy the business needs and address the automation of the business processes by scoping the responsibilities of the IT systems and enumerating the steps that must be taken by those systems.
- Quality Attributes, aka non-functional requirements, that capture performance, availability, and other such requirements

Understanding the Big Picture

The top level diagrams from each of these business architecture areas, along with their textual documentation, should be sufficient to understand what the nature of the company's business is all about. The Business Architecture should be mandatory reading for any one joining the company, regardless of whether they are in the business or IT sides of the house.

Application Architecture View

Application Architecture deals with how applications are built and interact with each other. Building a suite of applications that communicate across your intranet with a service oriented approach to distributing core business logic and information will utilize a different Application Architecture than a suite of applications that run on the same PC and share common functionality. In either case it makes sense to look at the Application Architecture from two perspectives, namely the functional and integration views. The level of detail you want in either view depends on the type of IT shop you have. If you develop applications in house, then you will want to explore the underlying components of your applications. If you are mostly an integration shop that purchases off-the-shelf software and provide some "integration glue" to allow them to communicate, then you may only need to deal with the applications as "black boxes."

The Application Architecture is composed of two sections, the design of the applications and the use case realizations that depict how projects have used the applications and their components to realize the functional requirements of the project's use cases. The realizations are archived and maintained in the Application Architecture so they are available to future projects that may need to modify the use cases or want to leverage what has been done on previous projects. Use case realizations for new projects reside in the projects area of the enterprise model until they have been completed. At that time, their results are moved to the Use Case Realization section and the various architecture views will be updated if the project has impacted them.

Application Architecture is unique in that it incorporates the other views as necessary to explain how an application is built and where it will be deployed.

Application Architecture Behavioral View

This view shows what the application or use case realization does. Various behavioral diagrams can be used, i.e. activity, state machine, communication, sequence, timing, and interaction overview diagrams.

Application Architecture Functional View

The Functional View of the Application Architecture depicts how the applications (or their components, depending in the level of detail you wish to obtain) interact with each other. Applications or their components communicate through well defined interfaces. Depending on your underlying integration architecture, these communications may be direct or indirect. Regardless, it is still valuable to understand with which applications a given application communicates in order to understand the risk involved when modifying a dependent application. The Functional View also shows the general mechanism by which the applications communicate, e.g. by file transfer, synchronous or asynchronous request, etc. With Sparx EA, these can be realized as Quick Links to guide the designer towards conformance to the architecture standards

you have created. See Figure 2 below for an example of the functional view and Figure 4 below for a Quick Link example.

Application Architecture Integration View

The Integration View of the Application Architecture shows how applications (or their components) actually interact. Whereas the Functional View may show that Application A communicates with Application B via a synchronous request, the Integration View may show that they communicate asynchronously via an enterprise service bus with adapters handing the protocol change from synchronicity to asynchronicity and back.

Application Architecture Data View

This is a subset of the Data Architecture (see below) that is directly related to the application or the use case realization. It includes database tables and local data utilized by the application or use case realization.

Application Architecture Technology View

This is a subset of the Technology View (see below) that is directly related to the application or the use case realization. It is required only if new components are being introduced by the project that need to be mapped to their technology requirements or if the technology requirements of existing components are changing. Otherwise, a link to the approach diagram in the Technology View is sufficient.

Application Architecture Deployment View

This is a subset of the Deployment View (see below) that is directly related to the application or the use case realization. It is required only if new components are being introduced by the project and need to be mapped to the computers on which they are to be deployed. Otherwise, a link to the appropriate diagram in the Deployment View is sufficient.

Application Architecture Operational View

One part of the Application Architecture Operational View depicts any products or processes that are necessary to ensure the health of the application in the data center. The other part of this view deals with the procedures that must be followed to deploy the application and to back it out if necessary.

> The Application Architecture view includes sub views of the other Architecture Views to show how applications are built and deployed.

Data Architecture View

Data Architecture depicts the various databases required to support the applications, their underlying table structures at the logical and physical level, the associations among the tables, and the policies that govern the attributes and associations of the tables. Sparx EA provides the capability to generate your data model directly from the Business Information Model, after which you can refine the data model for normalization/denormalization principles and technology driven requirements.

Service Architecture View

The service architecture depicts the business services available to the applications as well as IT services like logging, security, distribution, etc. The IT view of the Service Architecture incorporates all of the types of IT services required for an open distributed processing environment including coordination, integration, management, repository, and security services. The same sub views in the Application Architecture are used for the development of the services.

Technology Architecture View

Technology Architecture shows all of the technology components required to support the business applications. They are categorized into hardware versus software with each broken down into sub categories. Among the hardware components are nodes and devices like PCs, workstations, servers, switches, firewalls, disk arrays, etc. The software components include operating systems, integration & middleware software, application servers, DBMS's, and software tools like programming languages. These are linked to the applications and application components that require their presence to operate.

Deployment Architecture View

This view shows the physical nature of the enterprise. Within it is depicted:

- The sites where the enterprise's hardware is located.
- The technology partners that are used to support the extranet aspect of dealing with customers, trading partners, and other external entities.
- The deployment of applications, application components, business and IT services on PCs, servers and mainframe computers.
- The network topology that connects the nodes together, including PCs, servers, firewalls, switches, load balancers and other network devices. See Figure 3 below for an example deployment diagram.

Operational Architecture View

This view depicts the processes and products that are used by the support specialists in the data center to monitor the applications to keep them running.

Quality Assurance Architecture View

This view contains the testing products and processes that are used to perform the analysis required to ensure the quality of the applications being developed.

Architecture Resources

This area provides guidance to design teams by presenting concepts and providing modeling aids. It includes objectives of the technology direction to be taken, transitional goals to be accomplished over a series of projects, and architecture patterns to be employed. It also includes the meta-models, UML profiles and patterns that control your approach to modeling.

Modeling Enterprise Architecture with UML and Sparx EA

Now that we have laid out what Enterprise Architecture looks like, let's talk about what we can use, tool-wise, to…

- model it,
- make it readily available to all stakeholders and development teams,
- use it to provide guidance,
- make it easier to conform to architecture standards than to ignore them, and
- provide input to strategic business and IT planning.

As you can see from the above discussion, we have a lot of different kinds of things to capture; e.g. business processes, application components, operating systems, servers, and network devices. It is imperative that the modeling tool provide the guidance necessary, making the conforming path the easiest to travel. Using a tool as generic as UML does not inherently provide that assistance. Fortunately, the architects of UML provided a means to extend UML so that it can fit specific needs. These extensions are stereotypes and tagged values which can be organized into UML profiles.

> Stereotypes and tagged values can be used to refine your UML modeling to target specific domains.

Stereotypes

Stereotypes provide a mechanism to refine the meaning or context of a given model element. They are notated by putting guillemets around the name of the stereotype, e.g. «entity». UML defines three analysis stereotypes that help provide additional meaning to classes. A class with an «entity» stereotype is something that is concerned about data and the control of that data. A class stereotyped as «control» is concerned about process. A «boundary» class represents something at the edge of the system being modeled that interacts with things external to the system.

Stereotypes can also be created by the designer. The concept of a control mentioned above is pretty general. To refine it for an open distributed processing environment using an n-tier approach, the following stereotypes can be introduced, indicating the type of interaction provided by the control object:

- «Service» - a control that maintains the state of an interaction only within the context of a single invocation.
- «Session» - a control that maintains the state of an interaction through multiple invocations with the initiation and termination of the session controlled by the client.

To place a control into the proper tier of a distributed environment, the previous stereotypes can be further refined to:

- «Presentation Session» - a «Session» that controls the interaction with the user.
- «Work Session» - a «Session» that controls the sequencing of actions required to perform the work of a use case realization. A «Work Session» can interact with «Presentation Sessions», «Business Services», «IT Services», and other addressable components within its deployment space.

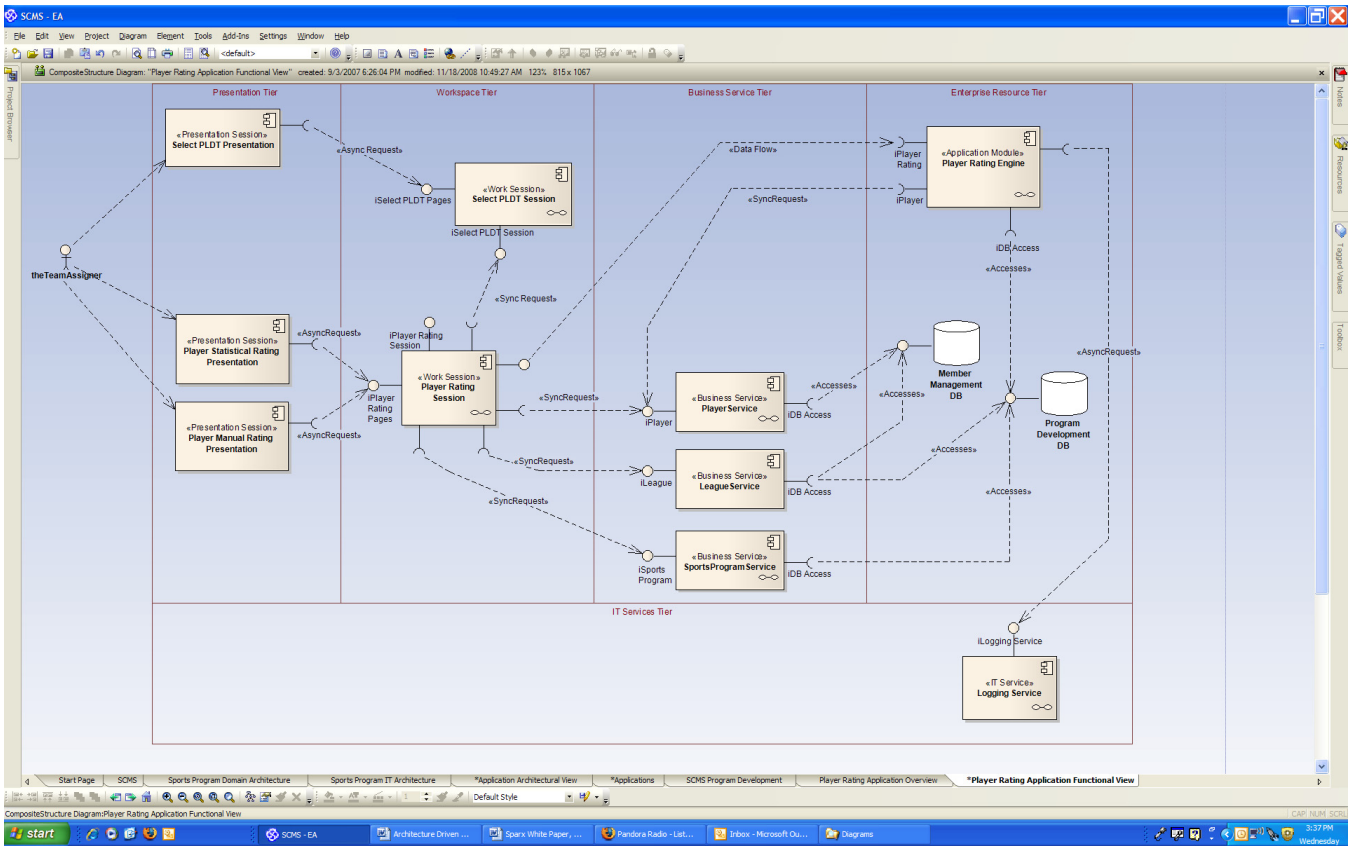
- «Business Service» - a «Service» that provides access to and control over reusable data and processes by interacting with enterprise resources such as databases and back end applications as well as extended enterprise resources, e.g. connections to trading partners.
- «IT Service» - a «Service» that provides aspect functionality such as logging, security, service discovery, transaction control, etc.

Now, instead of simply seeing a class with a name, you can get the full context of what type of processing the class should do as well as its location within the distribution tiers. Furthermore, you can decide what interactions among the stereotyped classes will be permitted. Ultimately, these platform independent stereotypes can be mapped to components stereotyped for the platform specific model to be created. A particular «Work Session» may be mapped to a «JSP» for one platform or to an «ASP» on another. Pattern generated transformations can automate this process to a large degree. In fact, there are a number of UML profiles that are available for different domains.

Stereotypes can be used for model elements other than classes. For example, you can add a stereotype to a dependency connection. In the Application Architecture Functional view, we have application components interacting with each other. These interactions demonstrate dependencies between the application components. We further refine the interactions (the dependencies) based on the conceptual type of interaction, namely «Data Flow», «Shared DB», «Sync Request», or «Async Request». These too can be transformed into the underlying technology structures of a platform specific model.

The following diagram shows the functional view of the Player Rating Application from the reference architecture model we use for training purposes and deliver with $(EA)^2$ (see below). In it, you can see the various stereotypes discussed above.

Figure 2: Application Architecture Functional View



Tagged Values

Tagged values allow you to capture information about a model element that UML does not provide. For example, you may want to know the beginning and termination service dates for a given «Application». You may want to capture the various service level requirements for a «Business Service», e.g. average and maximum execution times. Tagged values come in a name/value pair. You provide the name of the tag and then the value.

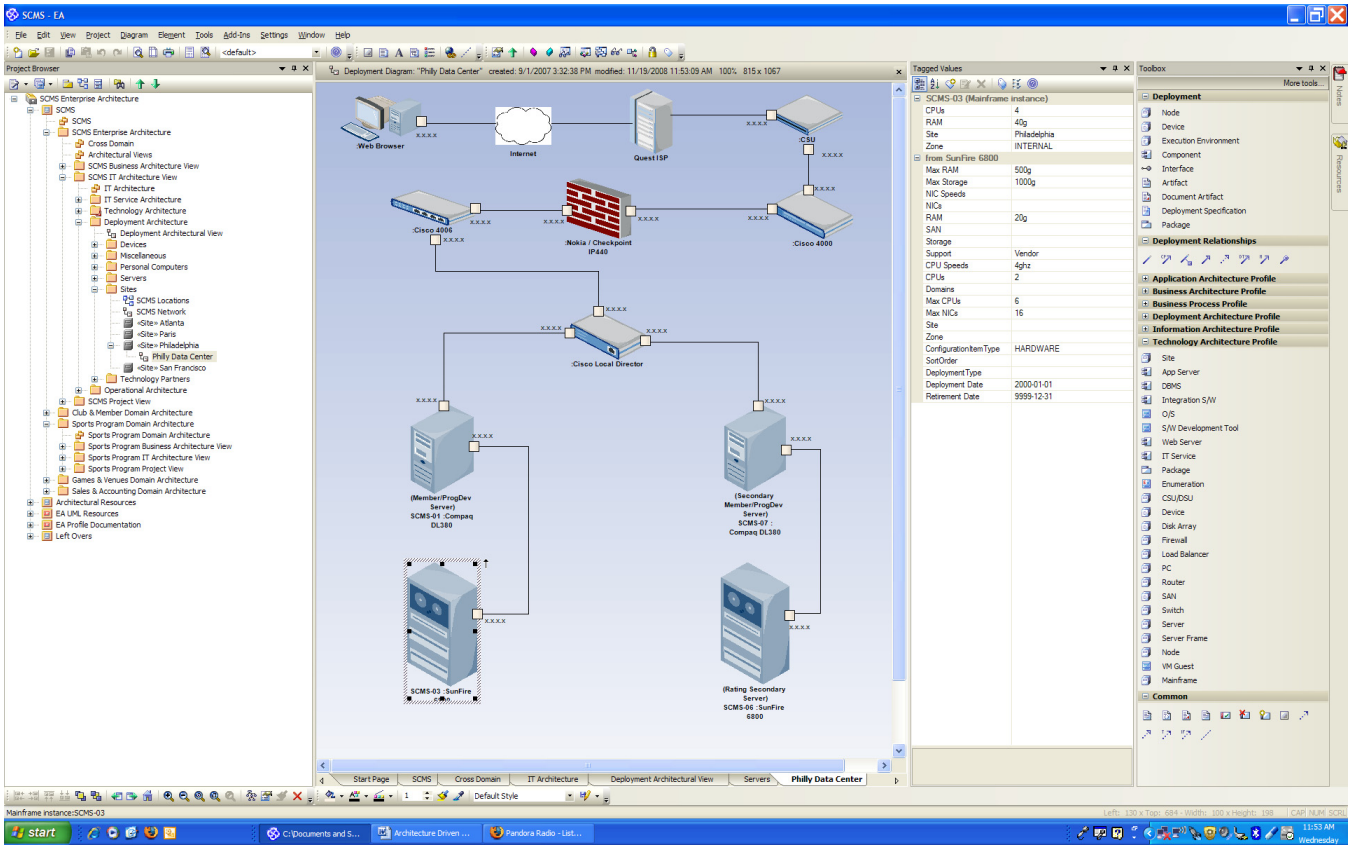
UML Profiles

UML allows you to add stereotypes and tagged values to a model element in a very ad hoc manner. This can lead to inconsistency and confusion across design teams. The clarity stereotypes can add to a model is quickly lost when those stereotypes are obtuse and undefined. A well-defined, well-documented set of stereotypes and their tagged values should be provided to all teams. Better yet, they should be used to create a meta-model which can be controlled by the modeling tool, with rights to manage the meta-model given to those responsible for it. With such a meta-model in place, the modeling tool can now provide guidance on the modeling standards embodied in the meta-model and make conforming to those standards easier than not doing so.

> Creating UML Profiles for each Architecture View puts modeling elements in the hands of the designer that speak directly to the view's intended audience and provides guidance to the designer.

By creating UML profiles from the extended meta-model, you can replace the generic constructs of UML with ones that are specific to the language of the view. For example, the following screenshot shows the network topology of the Philly Data Center from our reference architecture model. On the left is the browser, showing that we are in the Deployment Architecture View. The second section from the left is the diagram itself. Note the use of alternate images to give the "Visio look" that network engineers seem to like. The third section shows the tagged values for the mainframe at the bottom left of the diagram. Note that there are tagged values for the model (SunFire 6800) representing a standard configuration and overrides for the instance (SCMS-03) showing the values for that specific computer. The far right section shows the standard deployment toolbox above and the Technology Architecture toolbox created from the Technology Architecture profile below. As you can see, the profile shows model elements that are germane to the Technology Architecture view.

Figure 3: Deployment Architecture – Site Topology



UML Patterns

Once you have created your profiles, the next task is to automate some of the more intricate patterns. Sparx EA allows you to create patterns directly from a diagram. When creating a pattern you can decide whether a given pattern element should create a new model element or be associated with an existing one. This lets you quickly create new elements and reuse existing ones just by adding the pattern to a diagram.

Quick Link

As mentioned above, one of the goals of creating a modeling framework is to provide instant guidance to designers and to ensure conformance to your architecture principles. In fact, the ultimate goal should be to make the conforming path the easiest path (I can't emphasize this enough!). Sparx EA's Quick Link capability helps with that goal. The Quick Link is an arrow that appears to the top right of a modeling element when you click on the element. If you drag and drop the arrow onto another element, a context menu appears allowing you to choose the appropriate type of connector. Sparx EA allows you to modify the context menu, based on the types of elements involved. This gives you control over what type of connections can be made between the types defined in your meta-model.

> Quick Links can limit the types of connectors between modeling elements to those you have decided upon in your meta-model.

The screenshot below shows how the context menu has been changed to incorporate the stereotyped connections between a RequiredInterface and a ProvidedInterface.

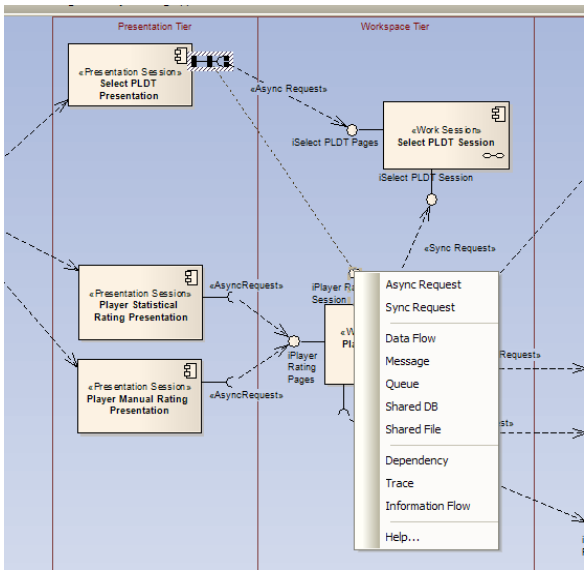


Figure 4: Quick Link Example

Sharing Your Enterprise Architecture

As mentioned earlier, having the best model of your Enterprise Architecture does not accomplish much if it is not readily available to all concerned. Sparx EA provides different ways of doing this.

Extended Teams Using Sparx EA

Sparx EA has a security feature that allows controlling access to packages within a model. Each user who has a login/password can be assigned individual access rights or they may be assigned to a group and inherit the group's rights. This allows users to review an entire model but to only make changes in the areas for which they have modification rights.

Read-only Access

Sparx Systems provides a free, read-only version of Enterprise Architect called EA Lite. This allows anyone with EA Lite to view all areas of the model in the same way that the people working in the model do, without the expense of purchasing the full application.

Web Site

Sparx EA can generate a web site for publishing on your intranet. This provides the widest possible audience, using technology with which most everyone is familiar. This method makes it very easy to share all those business principles and objectives and architecture guidelines on a very wide basis.

Published Reports

Reports can be produced by the built-in report writing capability of Sparx EA.

Distributing Non Sparx EA Files

Some things are just better captured in a spread sheet or a presentation or some other type of file than in a UML model. Links to these files can be embedded in the EA model, allowing the Enterprise Architecture model to be used as a distribution point for other forms of communication.

Summary

Enterprise Architecture is one of the hot buzzwords going around today and for large organizations in particular, it is very much a necessity for survival. However jumping into Enterprise Architecture modeling without a clear understanding of what you want to get out of it and how it can increase the bottom line may doom your effort to failure.

There are a number of tools available for modeling your architecture, but most come with a hefty price tag. Before jumping into the deep end with a modeling tool that will pretty much force its own approach, you may be better served prototyping your architecture with the open approach presented in this paper. Then, if you need additional capabilities, like modeling business simulations, you will be prepared to evaluate tools that provide those extra capabilities for your specific needs. On the other hand, you may find that the approach presented here not only satisfies your needs, but that since it is based on UML, an industry standard, you already have the knowledge and resources to jump right in.

About the Author

Terry Merriman has worked in IT development for 30 years as a developer, designer, development manager, and architect. He has consulted for fortune 500 companies in the insurance, finance, publication, automobile distribution, and pharmaceutical sectors, leading crucial production projects and setting strategic, architecture direction.

Mr. Merriman is CEO of OAD Consulting, Inc., which for the last 8 years, has concentrated on helping fortune 500 companies define their approach to Enterprise Architecture and modeling in general. He has developed frameworks that extend the capabilities of UML modeling tools to provide guidance to design teams, ensure consistency across their models, and make conforming to architecture principles easier than not doing so. He has worked with development and architecture teams on large projects to ensure successful delivery of the projects while mentoring the teams on the frameworks and the principles they embody.

Mr. Merriman is also a Senior Consultant for the Cutter Consortium where he has published several reports on architecture and modeling frameworks and participates in Cutter's Enterprise Architecture practice. A list of these reports and others can be found on OAD Consulting's website (www.OADConsulting.com).

OAD provides consulting and training services on

- Enterprise Architecture Modeling
- Business Modeling and Requirements Capture
- Analysis and Design
- Strategic Planning of Application Portfolios to Implement Business and IT Goals
- Meta-modeling client architecture and development environments with the goal of creating standardized approaches for all aspects of enterprise modeling at the architecture and project development levels

OAD Consulting, Inc. is a VAR for Sparx Systems.

About *(EA)²*

(EA)² is an MDG add-in for Sparx EA that provides

- A documented base UML model that includes an area for the Enterprise Architecture and its architecture views as well as an area for your development projects. Each major package is fully documented, providing support directly in the model as you build it. The base model can be used by organizations with one or many sub domains.
- UML Profiles for each architecture view, which help modelers speak to the different stakeholders in terminology specific to their area of expertise,
- UML patterns for automating the intricacies of modeling tasks and easing architecture conformance,
- Quick Links that guide the designer by providing the appropriate types of connectors for the elements being connected,
- Document and tabular based reports designed to feed strategic business planning as well as providing the blueprints for project development,
- A Reference Architecture Model that provides a complete example of all architecture views.

More information about OAD Consulting and *(EA)²* can be found at www.OADConsulting.com.

---

[i] TOGAF – The Book, available online at http://theopengroup.org