Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2010 Proceedings

Americas Conference on Information Systems (AMCIS)

8-1-2010

Role of Testers in Selecting an Enterprise Architecture Solution: An Exploratory Study

Rashid Koja
University of Memphis, rkoja@memphis.edu

Robin Poston
University of Memphis, rposton@memphis.edu

Follow this and additional works at: http://aisel.aisnet.org/amcis2010

Recommended Citation
Koja, Rashid and Poston, Robin, "Role of Testers in Selecting an Enterprise Architecture Solution: An Exploratory Study" (2010). *AMCIS 2010 Proceedings*. Paper 299.
http://aisel.aisnet.org/amcis2010/299

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2010 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Role of Testers in Selecting an Enterprise Architecture Solution: An Exploratory Study

Robin Poston
University of Memphis
rposton@memphis.edu

Jasbir Dhaliwal
University of Memphis
jdhaliwl@memphis.edu

Rashid Koja
University of Memphis
rkoja@memphis.edu

ABSTRACT

Software testing groups are playing an increasingly prominent role in both the software development lifecycle (SDLC) and in the long-term planning of technology architectures that support large-scale organizational information systems. The advent of integrated enterprise architectures (EA) provides new opportunities for testing groups to play a proactive role in building consistent and testable guidelines for improving enterprise-wide software quality. Given that testing groups historically have not been invited to participate in EA decisions, there is little academic literature or industry best practices on approaches that testers might use to guide their participation. This article draws lessons from the experience of a Fortune 100 corporation whose testing group used theoretical notions of "testability" to guide its involvement in an EA acquisition process. It describes how it operationalized testability criteria, incorporating controllability, observability, and simplicity, into various stages of the process and illustrates the benefits and challenges of taking such an approach.

Keywords

Enterprise Architectures; Testing; Development Lifecycles; Exploratory study

INTRODUCTION

Today's large scale organizations are globally dispersed and highly complex entities which interact frequently and often in real-time with other enterprises that are also dispersed in many different locations. As organizations continue to grow and become more collaborative and networked, integrated enterprise architectures (EA) are becoming more popular and necessary (Camarinha-Motos and Afsarmanesh, 2008). A well designed EA will apply systematic, holistic and rational methods to the design of an organization's data and technology infrastructures in order for information technology (IT) to more effectively and efficiently pursue its purposeful business activities. EA is a master plan of the organization covering business planning with goals, visions, strategies, and governance principles; business operations with business terms, organization structures, processes, and data; automation with information systems and databases; and enabling technologies with computers, operating systems, and networks. The challenge however is how to secure an EA that provides standardization benefits while resolving localization challenges created by complex global business operations involving diverse products, markets, and technologies.

The role of the testing group in software development has focused on how to test complex integrated systems where testing activities were relegated to one of the later stages in structured development approaches (i.e., waterfall model). As a result, many software products are conceived, initiated, and designed with little thought as to how they will be tested or even be testable in the later stages of development. Consigning testing to the later stages of lifecycle creates a multitude of problems including: too much work in too little time, how to fix problem code and design flaws, and software modules that are more difficult to test than needed (Gelperin and Hetzel, 1988). Companies are therefore moving testing activities to the beginning stages of the software development lifecycle (SDLC). The hope is that by applying testing earlier in the lifecycle, software teams will be able to catch bugs when the fix is cheaper (Zhu, Hall, and May, 1997). However, this movement to early testing is challenged by incompatible architectures, tools, and platforms that make it difficult. The advent of prototyping and newer more agile approaches to software development (i.e., SCRUM, Adaptive Software Development, Crystal Methods, and eXtreme Programing) has also not made the situation better because of similar reasons. The recent popularity of integrated EA that strive to provide common technological platforms for both IT development and operational purposes, therefore holds promise for having testers engage earlier in the SDLC. A key requirement for this is that testing groups must engage even beyond the start of the SDLC. They must be actively engaged from the very start in the conceptualization, design, and acquisition of the EA itself.

Activities within the SDLC, like testing, are directly influenced by the manner in which the EA is set up. How software will be developed and tested will be based on how the integrated EA is designed. Thus, the testing group must have input into the strategic discussions about the overall EA that will serve as the basis of software development and testing down the road. This article explores the following research question: *How can a software testing group in a large global organization play a proactive role in the definition, design, and vendor selection process for acquiring an EA that will serve as the focal point for future IT activity?*

LITERATURE REVIEW

Enterprise Architecture

An EA description gives a holistic, systematic description of an enterprise. It encompasses business functions, business process, people, organization, business data, software applications, and computer systems with their relationships to enterprise goals. Enterprise model methodologies and tools allow the user to represent, visualize, understand, communicate, redesign, and improve the operations of the enterprise (Chapurlat and Braesch, 2008). EAs use enterprise models to focus on timeliness, cost, quality, and speed-to-market. An enterprise model represents the environment, systems, and entity in the physical, social, and logical world of a company (Camarinha-Motos and Afsarmanesh, 2008). The model organizes the activities of constituents, roles for participants, and rules of governance over data, processes, and technology. An enterprise model is a tool for decision-makers, and it is especially useful for those designing and maintaining software systems that support enterprise operations (McGinnis, 2007).

Testability

In order to achieve testing goals, the testability of components of software code is critical. The IEEE Standard Glossary defines testability as, "the level at which a component facilitates the formation of test conditions and at the determination through whether those criteria have been fulfilled" (Freedman, 1991). Other definitions of testability include: the probability of uncovering a bug if it is present in the software (introsoftwaretesting.com); greater controllability, observability, and simplicity lead to easier testing (Payne, Alexander, and Hutchinson, 1997); and the likelihood of the code failing if something in the code is incorrect (Voas and Miller, 1995).

Testability consists of several notions such as controllability and observability (Binder, 1994). Controllability reflects the testers' ability to control inputs to the software code being tested and its internal environment (i.e., how the code within it works). Observability reflects the testers' inspection of the output of the software code being tested (i.e., how well the code performs its job). If the testers cannot control the input or observe the output, they cannot be sure how a given input was processed by the code. Lacking controllability means identical tests of the same code may produce different results. Lacking observability means the testers may think the output of the code is correct because they cannot see that it is not correct (Payne et al., 1997). But when input is controlled and output is observable, testers can determine whether the code is working properly (Bach, 2003; Binder, 1994).

Test automation also impacts testability. According to Fodeh (2003), "Automated testability is the degree to which the application under test facilitates the implementation, execution and maintenance of automated testing." Testers should consider the:

- Visibility of the output, errors, system interactions, etc. (i.e., applying a glass-box approach);
- Control of the ability to enter input, trigger events, and invoke methods associated with the capability to exercise command over system parts;
- Persistence which is the frequency of change of the software being tested;
- Consistency in the level of coherence in the look, operation, and performance of the software being tested;
- Reliability which pertains to the probability of the system functioning properly; and
- Documentation where good specification of system functionality and interface is required for adequate automated testing (Fodeh, 2003).

Software testability is not a characteristic of source code artifacts alone. From the domain of distributed real-time systems, companies are beginning to understand that the architecture (i.e., high-level design) of a software system can be a main factor of testability. Based on this premise, we extend this view and refer to testability as a characteristic of a software artifact (a system, component, or document) independent of the current abstraction level (Jungmayr, 1999).

The same views can be applied to a company's strategic software product development infrastructure that handles all aspects of the SDLC. In this environment the software product development infrastructure serves as the single source for the enterprise's product data and rules while enabling current product aware systems to leverage a common set of product

information. Adoption of an enterprise product development infrastructure across the enterprise enables integrated configuration, testing, and deployment methodologies for reducing time-to-market for new software products and their enhancements.

Engineering Testability into the SDLC

Design for testability has been proposed as an approach to increase the quality of software development outcomes (Pettichord, 2002). "Testability takes cooperation, appreciation, and a team commitment to reliability" (Pettichord, 2002, p. 25) and focuses on the more human aspects. Other researchers consider testability as the extent to which a software artifact enables testing in a given test context, with the goal being to reduce the overall testing effort (Jungmayr, 1999). If code lacks testability, similar to other design problems, it is costly to fix when detected late in the SDLC. As a result, testability should be addressed during earlier reviews of development work. This research describes different aspects of testability, offers heuristics to evaluate testability, and illustrates how reviews based on checklists help encourage testability throughout the SDLC (Jungmayr, 1999). Our approach illustrates the managerial processes need to gain buy-in for these endeavors.

Prior research has also addressed testability regarding how to address it in object-oriented development. Testability has been illustrated to be lower in object-oriented software than procedural implementations (Payne et al., 1997). To address these concerns, software design-for-testability is becoming more important. One paper has offered methods to measure testability based on the fault revealing ability of a class-components based on data flow analysis and considering definition and use locations (Kansomkeat, Offutt, and Rivepiboon, 2005). However, this technique is based on the program's implementation, and it fails to highlight testability issues prior to code development. Our approach compliments this method by urging development teams to consider ways to infuse testability into all the stages of the SDLC, including infusing it into the overall EA.

RESEARCH METHODOLOGY AND EXPLORATORY STUDY

We use a qualitative approach in gathering investigative exploratory study data on how testability is being engineered into an EA implementation process. This approach focuses on gathering interview data in order to provide descriptive and explanatory insights into the testability activities engineered throughout the EA acquisition process. This approach allows us to observe and analyze managerial data. This approach facilitates better understanding and interpretation of the contextual complexities of the environment in the outcomes. Finally, this approach encourages an understanding of the holistic systematic view of the issues and circumstances of our situation--that of engineering testability into the entire EA acquisition process from different development perspectives.

We selected a large Fortune 100 organization, herein called LogiCo, known to have successful software development and testing activities. The organization has approximately 5,500 IT employees, with 3 high-level managers, 14 middle-managers, and 149 full-time employees in the testing organization. The software development team (including the testing organization) has 4 major local and global roll-outs of hundreds of new and updated software versions a year. We were able to capture the composition of the development teams' relationships, the history and background of software development efforts, and the nature of practices used to foster progress in their endeavor to engineer testability into an integrated EA. LogiCo is adopting an EA solution using a product lifecycle management (PLM) methodology. The following topics delineate the important facets of testability when attempting to engineer testability into the EA.

Engineering Testability into the EA Definition and Vendor Selection Process

LogiCo offers integrated logistic services, which are highly automated, to the global marketplace. As a result the services, which are viewed as software "products" as they are referred to herein, are based on software code that encapsulates business data and rules. LogiCo is looking at increasing its software product development speed-to-market through a centralized software product management technology platform that can be integrated into the overall LogiCo IT infrastructure. This will enable holistic new product development processes and leverage a standard modular design for all global logistics products and related features of their services which are supported by the IT infrastructure.

Prior to pursuing an integrated EA, LogiCo began with a diffused and decentralized IT architecture platform, which reflected the high growth that the company experienced over the prior two decades, which did not permit central architecture planning, decision making, or IT maintenance and support (e.g., there was no central data dictionary that existed for the myriad operating companies that LogiCo owned). This led to a proliferation of programming languages, architecture, and interfaces, etc. As a way to overcome this complexity, an initial corporate initiative was established to explore technological consolidation and integration. An integration and standardization program or corporate initiative to consolidate technology tools and enterprise solutions was pursued. These efforts impacted the testing group who argued for a common architecture and tool set for both development and testing. After three years, these efforts morphed into a higher level EA initiative. An

EA team was established with key representatives from all pertinent business and IT sub-units with top management oversight and support. The EA definition and selection process used is illustrated in Figure 1.

Initial meetings of the EA team explored EA through sandboxing and adopted a PLM approach given that services could strategically be viewed as structured software products. An EA "Product" was defined as a base logistic service (generally delineated by geography, transit, and delivery time) supported by software modules with associated options and product support attributes that define a sellable logistics service. A Technical Quality Advisor (herein named Testing Lead) was selected by the Vice President of Software Quality Assurance to participate in the EA definition and selection process, as a representative of the testing group with continuous supported from the groups' knowledgably input. As the Testing Lead considered PLM and related approaches, he pondered questions regarding what does testing have to do with EA and how best could testing engage on the project. He engaged organizational experts, did research, called academics, and consulted with his testing leadership team for guidance. He decided that to facilitate quality assurance and reliability any new EA must ensure that all software products developed must meet basic testability criteria. It was made imperative that theoretical notions of testability be derived for practical application and injected into EA requirements white-paper documents, request-for-proposals, scoring criteria, and proof-of-concept stages of the EA acquisition process.

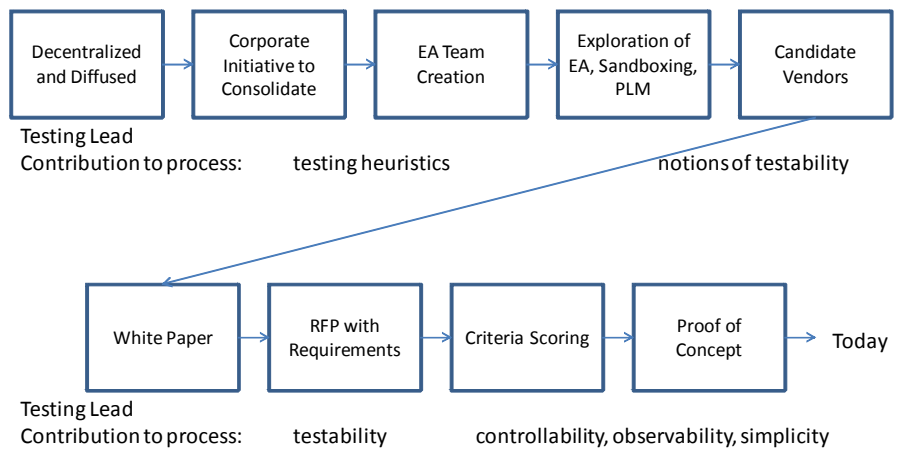

Figure 1: LogiCo's EA Acquisition Process

In a decision that was approved by strategic management, the EA team decided LogiCo needed to purchase a solution in the marketplace that will fulfill the Enterprise Product Manager (EPM) or configurator role. The capabilities needed spanned Product Data Management and Product Lifecycle Management (PLM) including product configuration with a focus on the intersection between product data and lifecycle management to support their EA vision. Currently, software product information was decentralized throughout multiple functional systems. The company needed the future EPM to be a single source of valid and verified software product information for all of the company's logistics products across the enterprise.

Figure 2 illustrates the key components of LogiCo's EA strategy that seeks to increase speed-to-market through three complementary dimensions adopted across the enterprise:

- **Modular Product Design**—Redesigning the company's products in a way that is modular and works with the LogiCo's IT infrastructure,
- **Technology Platform**—Building an EA software solution that allows business users to effectively and efficiently configure and manage company products, and
- **IT Enablement Services**—Retooling IT to accommodate the outputs of the EA solution.

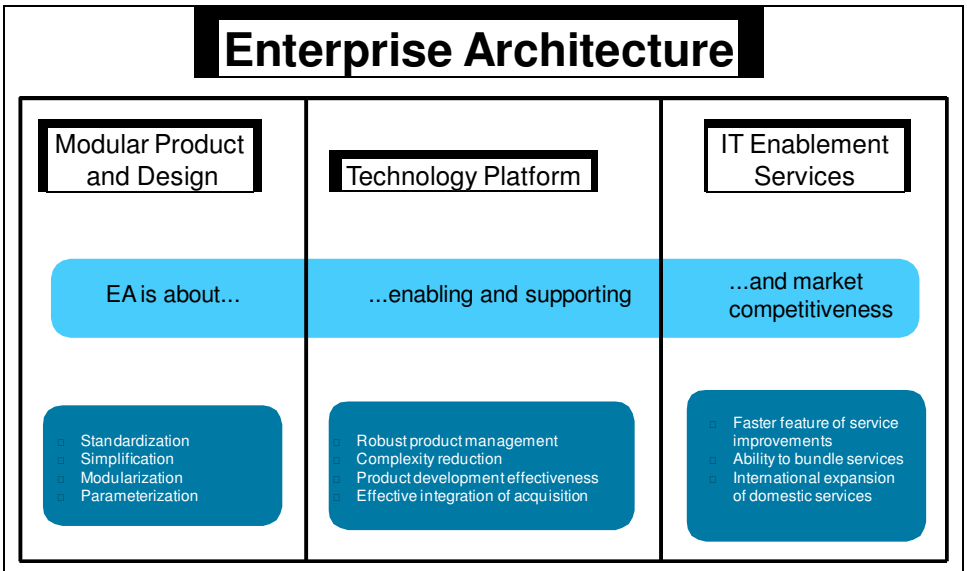

Figure 2: Components of LogiCo's EA Strategy

Modular product design of LogiCo's product framework was necessary so that business professionals could design products using building blocks (or tested coded modules) as opposed to defining a product in a silo approach. This allows for a more uniform option-driven approach to product development rather than a "start from scratch" approach for every new product or enhancement which is supported in the EA. Once established, product modularization offers flexibility to manage features within a product and across a product portfolio (i.e., effective up-feature or de-feature). In addition, modularization offers flexibility in defining base price and/or fee structures (i.e., what features are included in the base rate and what features are offered for a fee) of a new or existing product. Figure 3 illustrates how LogiCo is moving from an IT platform that supported software products that needed considerable integration testing efforts (i.e., built using a silo approach) to one that is more flexible based on a modular design.

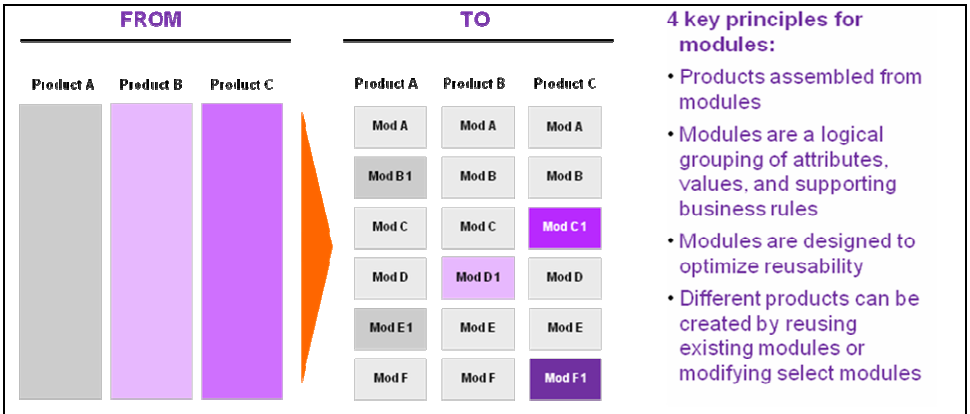

Figure 3: Description of LogiCo's EA Modular Design

The EPM approach for their EA uses for the logistics software products decomposes them into modules that are made up of attributes, values, and business rules. This hierarchy is used multiple times throughout the EA business requirements document and describes the various "items" managed within the EA product data structure. Rules can apply to any level in the product architecture and vary in purpose. For example, portfolios may be defined by geography, which could set a series of rules (e.g., Mexico doesn't allow dangerous goods). While another rule could apply at the attribute level and may state a link to another attribute for example (e.g. alcohol requires an adult signature).

The new EA product structure and module design will be configured and maintained within the EA software platform. An illustration of how these components will interact is shown in Figure 4. The EA solution includes several key requirements, including data and rule management comprising:

- **EPM Catalog**-data management for the complete list of modules, attributes, and values;
- **EPM Library**-data management for the complete list of products grouped into portfolios and available for sell in production systems; and
- **Product Rule Management**-rules management applied throughout the product and module design and configuration process and maintained in the EPM Catalog, EPM Library and Configuration workflow.

In addition, the EA solution will include components that involve workflow, visualization, analysis and reporting, comprising:

- **Product Lifecycle Management (PLM)**-visibility to product definition information across portfolios to enable effective product lifecycle management feature analysis with the ability to enact efficient changes to product definitions at the appropriate life-cycle phase;
- **Product Configuration**-product definition design, project management, and change governance workflow; and
- **Release Management**-publication management of product and module introduction, variation and change to LogiCo production systems, including quality assurance procedures, through varying techniques (e.g., versions or effective dates).

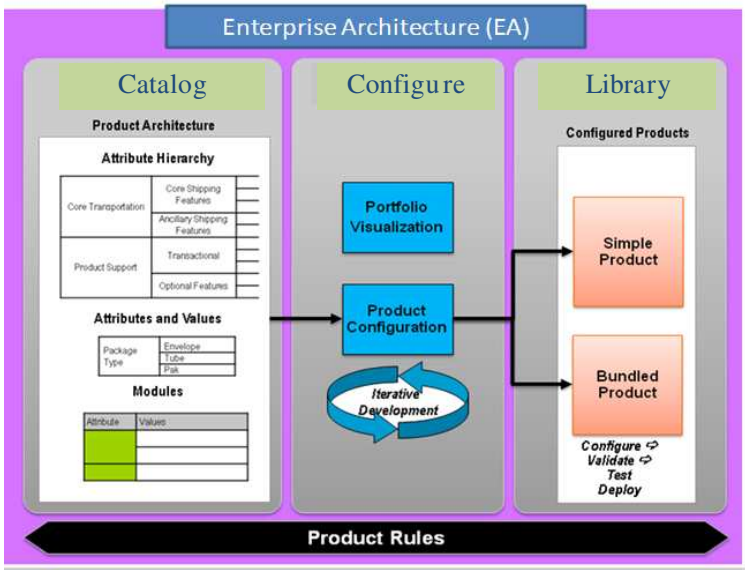

Figure 4: Description of LogiCo's EA Components

The goals of testability for the EA solution will be bound by certain key principles. At LogiCo, the overall goal of testing goals in the EA definition was to inject quality early in the concept and definition phases. This was done by articulating the following set of key principles to the EA team and winning acceptance for their inclusion in the EA acquisition process:

- Testability is comprised of controllability, observability, and simplicity heuristics;
- EA will be used to inject testability principles for building "quality in" earlier in the SDLC by its inclusion in the request for proposals from EA vendors and used as a criterion in their selection;
- Incorporating testability principles in requirements, architecture, interfaces, design, and coding activities will lead to higher quality software products created early in the SDLC; and
- Decoupling is a very important aspect of testability that must be factored into the overall architecture design.

A key achievement was that LogiCo's enterprise architects accepted these arguments and have adopted testability principles in the EA proof of concept and white-paper writing activities.

In order to accomplish the objectives of making the EA initiative a reality, the Testing Lead was assigned to work with the business, IT core, and extended EA teams. These teams were asked to understand the current state of PLM in the industry and participate in benchmarking sessions with leading providers and their customers that adopted this technology platform. At the beginning of this process, the sessions focused on vendor presentations of EA products as well as stakeholder interviews with key LogiCo business and IT representatives to discuss product feature sets and understand technology platforms. The key role of the Testing Lead in these sessions was to listen, learn, and understand the industry PLM capabilities for product development, while defining and espousing the role of "testability" for the new EA by reporting back to the testing group to obtain their input. At the end of a three-month period, the team narrowed down the list to ten providers. The team then conducted a stringent "request for proposal (RFP) process" where testability assessments were designed, conducted, evaluated, and recommended for the top two vendors who were selected to execute a proof-of-concept (POC) with LogiCo.

Engineering Testability into Vendor Selection

This section discusses how testability was assessed in vendor selection processes. The EA team started by looking at industry reports examining how well different vendors supported a PLM environment, configuration capabilities, and an integrated suite of software services. When examining the industry reports, none included measures of testing and testability. The Testing Lead defined and crafted testability requirements in the RFP documentation that was sent out to selected vendors who were asked to respond to the RFP and provide product demo evaluations. Figure 5 shows the key accomplishments as part of this evaluation process. Testability was built into the EA definition through the RFP requirements stating the need for integrated validation testing of products, modules, attributes, and rules within the EA itself and the need for enabling testing of a software product once it was configured. Testability was built into vendor selection though explicitly including scoring of testability requirements which were weighted 6.25% of the overall 30% assigned to the IT criteria as well as by selecting the top two vendors that met or exceeded the testability requirements.

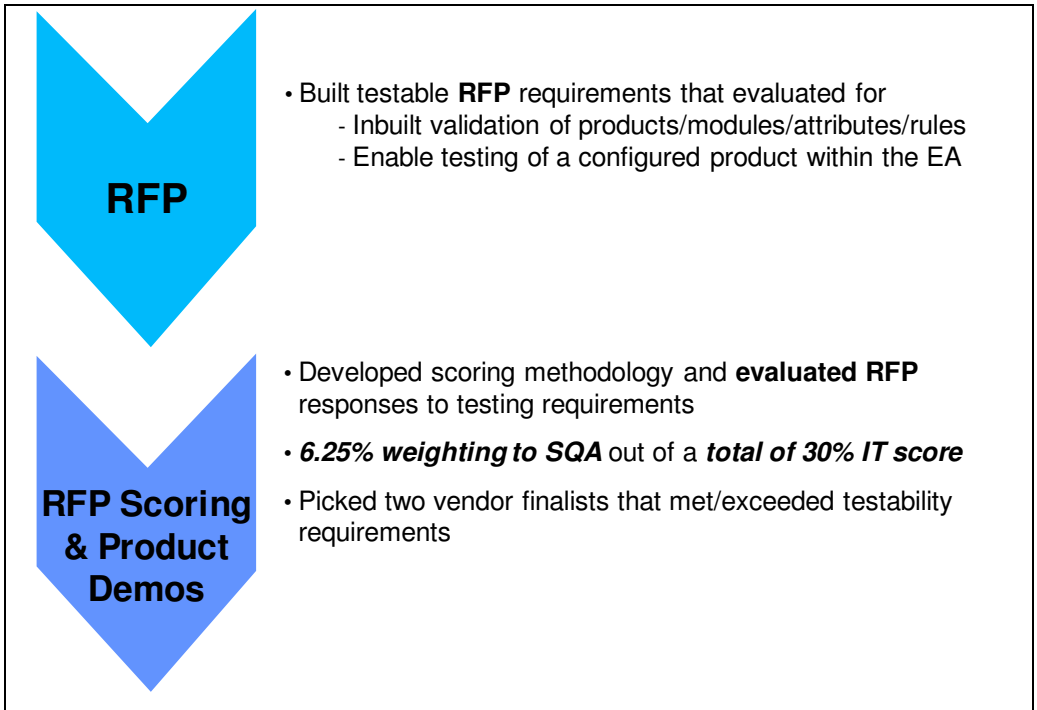

Figure 5: RFP Process and Testability Weighting in Vendor Selection

Using definitions of testability from the literature (Binder, 1994; Payne et al., 1997), the Testing Lead incorporated notions of controllability, observability, and simplicity into the RFP and EA product demonstrations by modifying and applying their definitions specifically to the LogiCo proposed EA environment. The application of controllability, observability, and

simplicity to the LogiCo environment can be found in Figure 6, which illustrates the key testability criteria that were designed into the RFP and vendor EA product demonstration activities.

Controllability (C)	Observability (O)	Simplicity (S)
The EA consists of a **rules development interface** that can be designed to create, export, import, query, examine and test rule sets and their associated attributes and values	The EA consists of **automated business logic triggers** to assess and check the viability of a rule, flag incompatible rules and/or modules	Enable simple creation of modules and its rules/attributes thereby promoting **functionality simplicity**
Dedicated product test environment to configure a product and **enable iterations** of the product development process and product validation.	Validate module for its rules and attributes by **inducing error logging and correction mechanisms** that enable the user to debug error messages	Evaluate products for **structural simplicity** to easily assemble new products via access to a EA catalog and library
Define, configure, test a **controlle**d suite of Enterprise Product use case scenarios and product data definitions Out of the box API's for standard integration to business rule management systems (ILOG, JRules) Testing tool sets (API's for QTP Load Runner Scripts and Automated Regression Testing)	Observe EA XML product data outputs and validate actual vs. expected for consumption to back-end product aware systems (FLAGS, EPIC)	

Figure 6: Testability Criteria within RFPs and Demos

The main attributes of testability considered important in selecting an EA were: controllability, observability, and simplicity. The next section considers each of these attributes and explains what makes them important to consider in the EA and PLM environment and how they were incorporated into the RFP process.

Testability in the EA RFP Processes

As illustrated in the left columns of Figure 7, key testability principles for controllability, observability, and simplicity were included in the RFP requirements. A key RFP requirement involved ensuring there is a mechanism provided for validating the configured EA (EPM to LogiCo) software module via a dedicated product test environment. This also ensures that there is intrinsic validation capabilities engineered within the EA architecture, which allows the product developer tool to validate the EA modules and their attributes defined by their business rules. The right columns of Figure 7 depict the actual feature sets that were observed for the corresponding RFP requirement that were evaluated in the EA vendor demos.

Controllability (Architecture)

The better we can control it, the more the testing can be automated and optimized

RFP Requirement

Testability Principle 1:
Software modules, objects or functional layers can be tested independently

•RFP Reqmt.: Modular architecture that architects "in-built modules, attributes and products"

•RFP Reqmt.: Capability to test configured products in a "sandbox" type environment

Testability Principle 2:
A scriptable interface or test harness is available

•RFP Reqmt:. Application should have the ability to provide common standard API's which can be linked with 3rd party products

Evaluated in the EA

Validate module for its rules and attributes by **inducing error logging and correction mechanisms** that enable the user to debug error messages

Dedicated product test environment to configure a product and enable iterations of the product development process and product validation

Out of the box API's for standard integration to business rule management systems (ILOG, JRules), testing tool sets (API's for QTP Load Runner Scripts and Automated Regression Testing) and XML type data for consumption to back-end product aware IT systems such as LSSI (FLAGS), EPIC, etc.

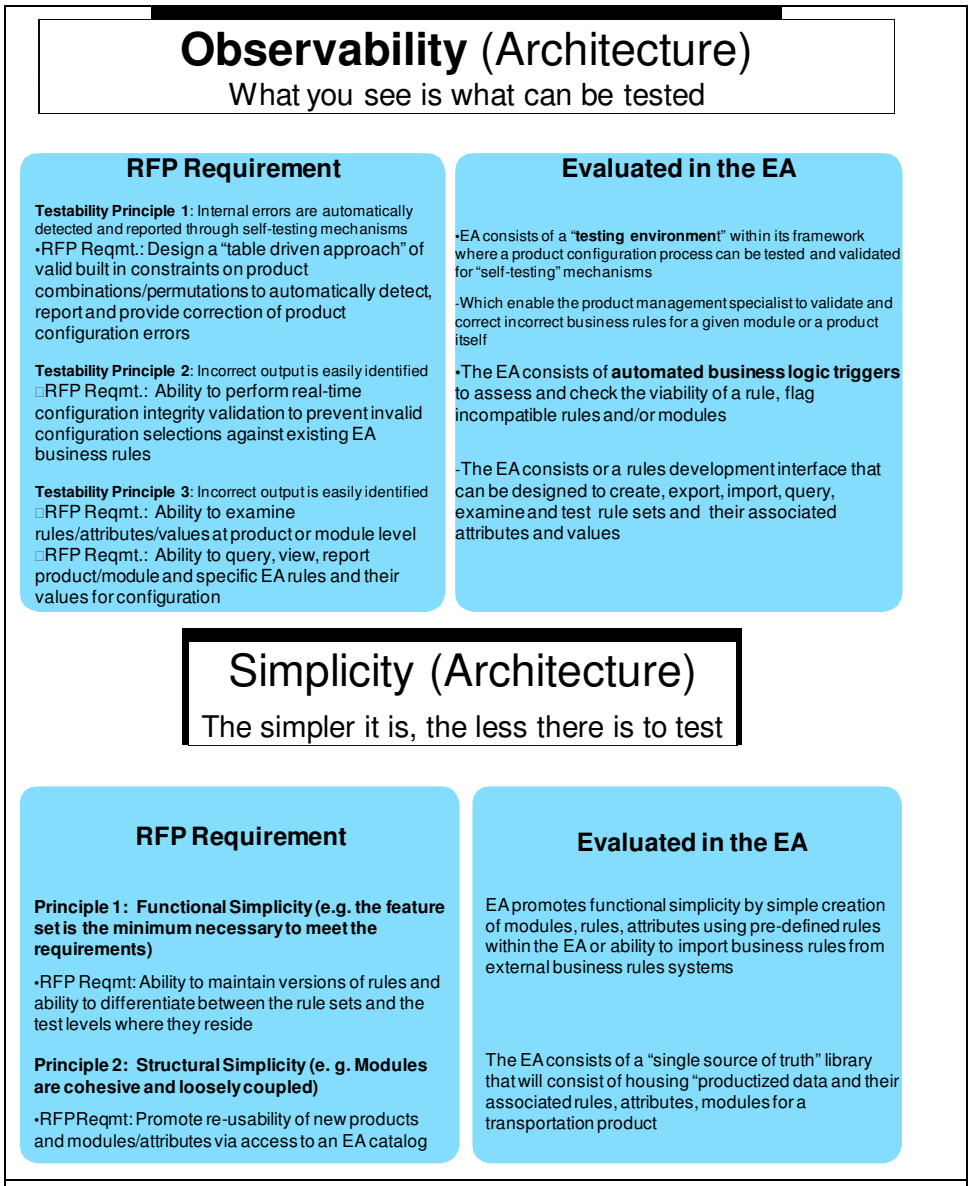

Figure 7: Testability in RFP

Testability Heuristics for Decoupling

LociCo's ultimate goal was a modularized EA with tight cohesion and loose coupling of its components. Testability can also be directly included in IT tight cohesion and loose coupling using the testability attributes. This in essence facilitates on-going integration and systems testing even in situations where particular modules do not meet specified criteria and thus must be decoupled from other modules. An integrated EA facilitates this if controllability, observability, and simplicity are built into the architecture. Figure 8 depicts how these were approached in relation to the EA acquisition processes at LogiCo.

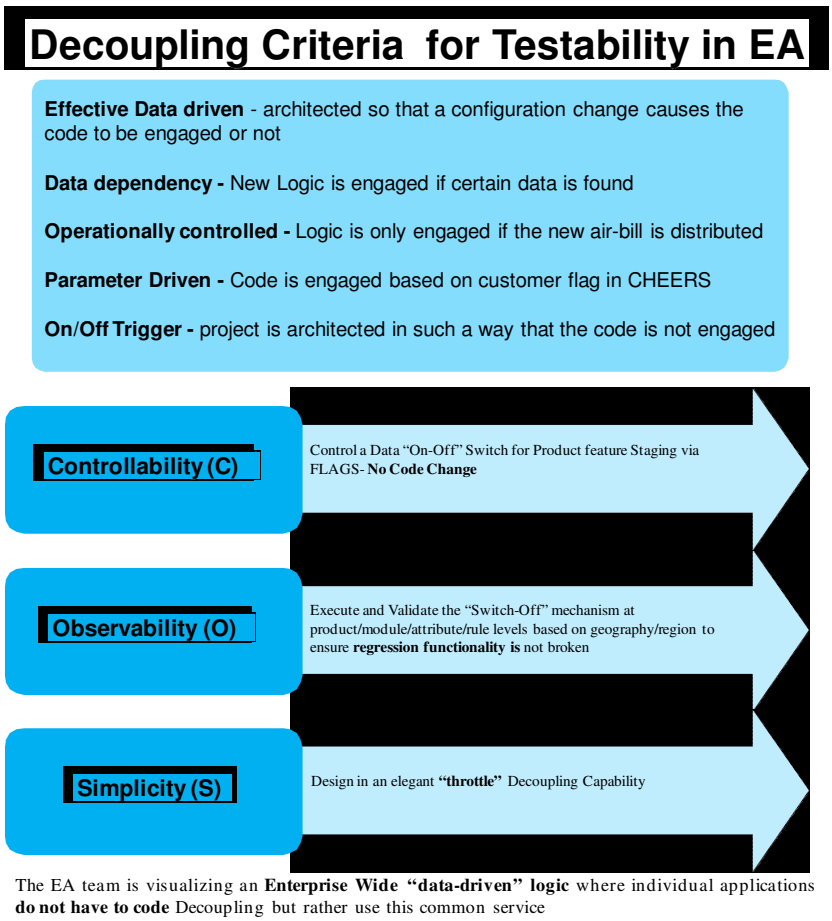

Figure 8: Decoupling Criteria and Heuristics for Testability

Ideal IT decoupling plans can be implemented as follows:

- Without code being backed out,
- Are testable at all test levels (integration/systems/functional testing) and testing cycles,
- Allow flexibility for business and operational constraints,
- Are clear and concise –reads like a disaster recovery plan, and
- Have flows depicting the 'decouple' points.

IT decoupling strategies are being conceived for the EA. In the context of the EA, let's say a new software product is defined as composed of A, B, and C. Some number of days prior to launch, it comes to light that a portion of B doesn't work and a decouple effort must be initiated. Can the product now be described as A, B1, and C (where B1 is some working version of the code)? Impact assessments can be initiated to ensure B1 doesn't negatively impact the product launch. Likewise, B1 may mean that the product launch is delayed and must be "switched off" for launch. The capability to "switch off" could be a mainstream capability for product and sub-product definitions. The goal is to have the flexibility to have certain products, certain countries, or certain features switched on or off.

There should be capabilities to decouple EA products by product type, feature, attribute, module, and rule sets. There should be decoupling capabilities provided in the product testing environment. One way to decouple is to rollback to the current version of that specific module, attribute, and rule and publish that back real-time into the EA while re-configuring the stated product in the EA and republishing the output of the decoupled re-configured product back into the product testing environment. Once this is done, regression and new functionality test cases must be executed and validated on the re-configured/decoupled product/feature/attribute/rule as the case applies to ensure it is not broken. Testing strategies will need

to be further defined to explore various decoupling testing scenarios. Given the requirements, two EA vendors have been selected for POCs.

IMPLICATIONS AND CONCLUSIONS

The EA acquisition process is still an ongoing activity at LogiCo. Testability elements continue to be tied to the integrated EA definition and vendor selection for a solution. The testing group is positioning the testability notions in the EA architecture whitepaper as a key driver for the planning phase deliverables. Next we define several lessons learned by the testing group based on their pursuit of infecting testability into the EA selection process.

Lesson 1: How to Involve Testers in Selecting an Enterprise Architecture Solution

The key outcomes of the EA acquisition process was the driving force to building quality in earlier through the testing groups' roles as a business technology leader, testing architect, and testing advisor. Building quality in early is the goal for injecting testability into the earliest stages in the SDLC. The Testing Lead not only focused on testability but took the time to understand the business landscape of the EA and the associated technologies and processes. While the testing group is not always included in the definition/development/launch phases of an SDLC, this case study illustrates how one company included a representative from the testing group in an EA decision from its inception. This case illustrates how software quality as promoted by the testing group can become the driving force for enterprise business and IT deliverables that are developed, managed, and used in high performance software and product development organizations.

On the EA team were Enterprise IT Architects, Enterprise Data Architects, Business Leads, IT Leads, Project Managers and extended team members from IT and Product Development from Marketing. The EA team members supported the testability principles and practices that were researched and proposed by the Testing Lead. Based on the experiences of this case example, the testability materials put forward to the EA team were not met with concerns but were embraced and ensuing discussions were related to how testability could best be applied to promote software quality earlier into the EA processes. There was cooperation by the team members once the testability concept was understood. In fact, the EA team encouraged and assisted the Testing Lead to create testability best practices. As a result, EA project team meetings could be leveraged as a training ground to help others understand the principles and help evangelize the inclusion of testability into the appropriate EA vendor deliverables (e.g., RFP, BRS, Scoring Models, Vendor demos, POC sessions, Whitepaper).

Lesson 2: How to Establish an Enterprise Test Architect Role in Large Organizations

One unique outcome of the Testing Lead's involvement with the EA team was his influence on the corporate enterprise strategic initiative for the selection, design, and adoption of the EA suites, platforms, and architectures. Through this process, the Testing Lead was able to coin the term "Enterprise Test Architect (ETA)", much in line as a counterpart to an Enterprise IT Architect. The ETA role was created as a senior position in the organization and treated on par with equivalent management positions throughout the company in terms of rewards, recognition, visibility, and influence. The future expectation is for the ETA to influence, mentor, coach, and provide direction to members of the testing group in EA activities. Key responsibilities expected from this new ETA role involve:

1. **Strategy** -- provide technical leadership and strategic direction to the testing group.
2. **Quality** – assist as the foremost technical authority with responsibility for the overall quality and testability of deliverables across all parameters, both functional and non-functional including performance, security, usability, etc.
3. **The Big Picture** -- maintain a "big and complete" picture of the software product, its dependencies, organizational goals, technology arena, etc. and helps guide and direct the functioning of the testing group appropriately.
4. **Influencer** -- influence the business' future direction, strategy, and planning as it relates to software products.
5. **Collaborator** -- collaborate effectively and on an on-going basis with all constituents involved in software product development and release activity including development, testing, technical publications, marketing, program management, and other entities to ensure execution timing and deliverables requirements are met.
6. **Testability** – support as a business technologist the understanding of business requirements and work with the development architects to translate requirements into solution architecture designs and incorporate testability as part of overall designs. Also, review requirements and seek clarity as required, participate in product design reviews and work with the development architect and development team to make any design improvements and refinement as needed.

Lesson 3: How to Define Testability for an Enterprise Architecture Solution

Through this case study, we illustrate that design for testability was a central theme and the company expects it will result in a better EA adoption as long as the selected EA framework can follow the four defined software product development principles:

1. **Comprehensive**: covering all applicable EA products
2. **Rigorously accurate**: all aspects of software product definitions are fully accurate
3. **Standardized**: all definitions are in standard form, nomenclature and style
4. **Fully described**: Each software product is individually fully described

In order for the EA to be testable at the architecture and design layers, the company included the following heuristics in order to achieve an EA that is effective, efficient, and adoptable:

1. "In-built validation" of the software modules and their attributes that design in mechanisms to promote functional validation before promoting them to 'end to end' or 'black box testing' techniques.
2. "Self-error checking" or exception handling mechanisms to promote "crisp" error messages that enable testers/users/business analysts to debug business rules and enable them to correct and validate.
3. "Sand-box" type environment to enable "Self-Testing" of the actual product configuration/development and validation aspects via iterations before it can be released to a software runtime environment.

In summary, this paper presented a testability approach that was used by a Fortune 100 corporation to ensure that the EA it was adopting would support testing efforts aimed at assuring high standards of software quality. An exploratory study approach was used to demonstrate its viability including details about how testability criteria were infused at multiple stages of the EA acquisition process. A key implication of the approach espoused in this paper is that testers have a pivotal role to play in the design phases of holistic EAs that serve as the integrated platforms for enterprise-wide software development and IT operations. While the traditional loose coupling and tight cohesion principles of modular design remain important, testers have a responsibility to ensure that the new integrated platforms also facilitate controllability, observability, and simplicity in software development as a means of ensuring testability of applications that are developed. Importantly, this paper demonstrates how testing leaders can play a valuable role in shaping the EA of their organizations.

REFERENCES

1. Bach, J. 2003. "Heuristics of Software Testability," (http://www.satisfice.com).
2. Binder, R. V. 1994. "Design for Testability in Object-Oriented Systems," *Communications of the ACM* (37:09), pp. p87-101.
3. Camarinha-Matos, L. M., and Afsarmaneshz, H. 2008. "On Reference Models for Collaborative Networked Organizations," *International Journal of Production Research* (46:9), pp.2453-2469.
4. Chapurlat, V., and Braesch, C. 2008. "Verification, Validation, Qualification, and Certification of Enterprise Models: Statements and Opportunities," *Computers in Industry* (59), pp. 711-721.
5. Fodeh, J. 2003. "Automated Testability: The Missing Link in Test Automation," http://www.stickyminds.com/sitewide.asp?ObjectId=6858&Function=DETAILBROWSE&ObjectType=ART.
6. Ford, S. 2005."What are Test Hooks?" http://blogs.msdn.com/saraford/archive/2005/03/16/396891.aspx.
7. Freedman, R. S. 1991. "Testability of Software Components," *IEEE Transactions on Software Engineering* (17:6), pp. 553-564.
8. Gelperin, D. and Hetzel, B. 1988. "The Growth of Software Testing," *Communications of the ACM* (31:6), pp. 687 – 695.
9. Jungmayr, S. 1999. "Reviewing Software Artifacts for Testability," presented at *EuroSTAR99* (November 8-12), Barcelona, Spain.
10. Kansomkeat, S., Offutt, J. and Rivepiboon, W. 2005. "Increasing Class-Component Testability," presented at *IASTED Conference on Software Engineering*, pp. 156-161.
11. McGinnis, L. F. 2007. "Enterprise Modeling and Enterprise Transformation," *Information Knowledge Systems Management* (6) pp. 123-143.
12. Payne, J. E., Alexander, R. T. and Hutchinson, C. D. 1997. "Design for Testability for Object-Oriented Software" *Object Magazine*, pp. 34-43.

13. Pettichord, B. 2002. “Design for Testability,” Presented at the *Pacific Northwest Software Quality Conference*, Portland, Oregon, October 2002.
14. Voas, J. M., and Miller, K. W. 1995. "Software Testability: The New Verification," *IEEE Software* (12:3), pp. 17-28.
15. Zhu, H., Hall, P.A.V., and May, J.H.R.. 1997. “Software Unit Test Coverage and Adequacy,” *ACM Computing Surveys* (29:4), pp. 366-427.