

MarkITS Technical Information Paper TIP

Technical Information Papers present strategy and tactics for Enterprise Architecture service enablement.
For more information contact one of our Executive Customer Care representatives using the [MarkITS contacts](#) page.



Topic: MarkITS Point of View – Architecture Tenets
Customer: Financial Services, E-Commerce, High Tech., Government
Product: Architecture Assessment Tenets
Date: 4/4/2009
Author: Mark S. Sternberger
Domain: Enterprise

Page 1

Executive Summary:

The DEADONS IT architecture framework defines a layered taxonomy to simplify complexity and establish key performance metrics (KPM's) to measure the value and ROI of business, IT, and product capabilities within and across the **D**evelopment, **E**nterprise, **A**pplication, **D**ata, **O**perations-business, **N**etwork, **S**ecurity, **I**nformation, and **T**echnical architecture domains with the goal of improved PARTS.

The improved PARTS (**P**ortability, **P**erformance, **A**daptability, **A**vailability, **R**euse, **R**eliability, **T**estability, **T**raceability, **S**upportability and **S**calability) tenets empowering people, process, and product development initiatives provides a foundation of KPM's for tactical implementation gating as well long term business analytics trending of longer term assessments of business-technology convergence and governance strategies.

We define a framework of tenets for improved PARTS which guide the data acquisition necessary to characterize quality and operational effectiveness within a subset of the DEADONS domains, specifically the **D**evelopment, **A**pplication and **D**ata architecture layers.

Portability:

The objective of the portability tenet is to abstract the application and data implementations from platform dependencies such as the operating system executive services and the physical hardware layers such as processor and communication bus architectures for example. The priority and magnitude of the portability requirements is determined based upon the expected life span of the application relative to the persona of target user (internal field agents vs. supported enterprise product), the number of target users, and the diversity of platforms targeted for deployment, future platform performance upgrades desired to be leveraged and timeframes in which they may come to market

Performance:

- **Application** designs if poorly conceived can exponentially bottleneck application performance as business rule complexity deepens and broadens. Design mechanisms regarding dynamic load balancing, levels of indirection, memory management, consistent usage of foundation methods, how executive service requests are wrapped, calling trees and class hierarchies, inter and intra process communications and external interface design and implementation issues are examples of performance metrics which if not properly applied can adversely effect the system throughput. Application visual modeling is an important tool for identifying otherwise unrealized optimizations. How object-relational mappings are implemented is important for streamlining performance if metadata relationships are not efficient.
- **Database** design consists numbers of tables, columns, reference integrity among them, the degree of normalization and the effective application of the 3rd party DBMS services used to implement the designs and maintain the content within the schemas. The % of “business entity” persisted data vs. supporting data for error handling, defects and ancillary data support embedded within the application DB is an important realization to occur. DB visual modeling is an important tool for identifying otherwise unrealized optimizations.



MarkITS Technical Information Paper TIP

Technical Information Papers present strategy and tactics for Enterprise Architecture service enablement.
For more information contact one of our Executive Customer Care representatives using the [MarkITS contacts](#) page.

Topic: MarkITS Point of View – Architecture Tenets
Customer: Financial Services, E-Commerce, High Tech., Government
Product: Architecture Assessment Tenets

Date: 4/4/2009
Author: Mark S. Sternberger
Domain: Enterprise

Page 2

Adaptability:

Modularity, internal subsystem and entity cohesion, low levels of coupling are general goals which promote lower Total Cost of Ownership, decrease time to market for future releases by maximizing reuse, generally extend the life expectancy of applications. Interpretive languages are great for prototyping and experimentation proof of concept, but for production systems generally do not provide the robust qualities necessary. Compiled applications on the other hand require a conscious effort to design adaptability by applying effective “chunking” skills to increase modularity, plug ‘n play effectiveness via API’s, wrappers, and minimal interfaces or otherwise “hidden dependencies” which can inhibit non-intrusive modification upgrades.

Availability:

Does the application and DB design provide high availability to users for all levels of concurrent transaction volume targeted? Where are the degradation thresholds for high availability and what are the acceptable levels of availability as defined by the business? When can the system be not available for archival purposes, if ever? This availability may decrease at faster than linear rate as the volume of users for an alternative business line or as complexity of the business increases over time.

Reusability:

Component vs. object approach and understanding the design mechanism differences and establishing consistency across the design team is important. Either approach is effective when properly adhered to. Goals “high degree of cohesion” internal to entities and “low degree of coupling” between entities are important design principles. This maximizes opportunities for future business lines, makes maintenance costs lower, and decreases the time to market for new applications or new releases and fixes.

Reliability:

Does the system provide consistent repeatable results under all the various operating modes, stresses and durations?

1. What is the mean time between content errors;
2. Number of outstanding content or functional defects open;
3. Average time to resolve a content or functional defect.

Testability:

Traceability is a key driver in assuring a system is testable. A clear and succinct response these types questions can be quite telling and also indicates how good the other PARTS areas will be for an application. Test documentation availability including Use Cases, Test Cases, Test Plans and Test Procedures and a traceability between these artifacts yield great insights into the overall evaluation.

Supportability:

- **Application** – Look for indicators of high support needs due to low levels of inheritance, poor component packaging or lack of packaging altogether, complex makefiles and build procedures, unreadable on non-self documenting source code, infrastructure functionality and rules which are not separate from business rules, many redundant copies of foundation rules which can potentially exist throughout the application. Be aware of strong technical leads which can seemingly speak to and mask these problems on the surface but yet has a team which otherwise has little knowledge transfer across the team which impedes an efficient support model. In this scenario, as the application matures with



MarkITS Technical Information Paper TIP

Technical Information Papers present strategy and tactics for Enterprise Architecture service enablement.
For more information contact one of our Executive Customer Care representatives using the [MarkITS contacts](#) page.

Topic: MarkITS Point of View – Architecture Tenets
Customer: Financial Services, E-Commerce, High Tech., Government
Product: Architecture Assessment Tenets

Date: 4/4/2009
Author: Mark S. Sternberger
Domain: Enterprise

Page 3

changing personnel, this is a risk is exacerbated. Visual modeling would vastly improve the supportability through effective knowledge transfer.

- **Database** – Is there a high degree of dependencies and referential integrity maintenance that is required which may cause inefficiencies. Look for recurring tables and content. Look to see if effective use of lookup tables has been applied which is a technique often used to make up for design shortfall in an attempt to increase operational performance in table lookups at the great expense of supportability and reliability. Visual modeling would vastly improve the supportability through effective knowledge transfer.

Scalability:

- **Volume** – From a context of the application threading mechanisms and database update mechanisms and evaluation of metrics such as number of transactions per time increment, number of concurrent users that can be handled, and understanding where the degradation occurs along the graph and if the degradation curve is linear or exponential are important characteristics to understand. Will applying more hardware into the environment effect scalability or is there a software threshold which when reached will no longer result in increasing transactions or users. Most importantly, does the application and DB design render themselves to be tweaked and improved upon, in other words can they be scaled and does do they respond to adjustments? To what degree can foundation components be easily modified to provide different functionality by the application, i.e. change once, effect many vs. change many to effect once.
- **Extensibility** – Detail design extensibility while a bit cumbersome with some script based loosely typed and COM like partitioning can be adequate and manageable, but somewhat burdensome if there are high degrees of redundancy of base rules in multiple modules. Effective uses of “infrastructure” vs. “business” rules encapsulated within appropriate modules are utilized to limit potential proliferation of redundant base rules spreading throughout an application causing increased future maintenance costs.

