# OpenText™ ALM

## Project Topology Best Practices

**For ALM Practitioners**

opentext™

# Legal Notices

© Copyright 2023 Open Text.

The only warranties for products and services of Open Text and its affiliates and licensors ("Open Text") are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Disclaimer
Certain versions of software and/or documents ("Material") accessible here may contain branding from Hewlett-Packard Company (now HP Inc.) and Hewlett Packard Enterprise Company. As of September 1, 2017, the Material is now offered by Micro Focus, a separately owned and operated company. Any reference to the HP and Hewlett Packard Enterprise/HPE marks is historical in nature, and the HP and Hewlett Packard Enterprise/HPE marks are the property of their respective owners.

# Contents

# Welcome to This Guide

Welcome to the ALM Project Topology Best Practices guide.

This guide provides concepts, guidelines, and practical examples for the best approaches on structuring ALM projects in various organizations.

## About Project Topology

Today, more than ever, the quality of your applications directly impacts your bottom-line business results. This is the reason IT departments of most enterprises invest heavily in products, people, and processes that can maximize application quality. Application software plays a dominant role in today's business, regardless of the vertical market or core competency. Every organization must be able to guarantee high quality, working software to properly position and deliver its products to the market. Now more than ever, software is a critical component of winning the competition.

OpenText is the leading provider of market-defining solutions Quality Center and Performance Center; both are focused on the automated and performance testing arenas. These two products forever changed the landscape of what was expected in the area of software delivery. Now with ALM, OpenText introduced complete ALM solution that covers all phases that software travels through during its existence. Embracing ALM practices and solutions enables software teams to meet the high expectations and demands of the business. Stakeholders in the application lifecycle, both from IT side and the business side, such as business analysts, developers or testing professionals, have to communicate, collaborate, and connect to ensure that every aspect of the software will meet the rigorous demands of the business.

ALM suite can serve various companies to achieve their specific needs based on industry segment, company focus and processes, amount of applications and their type and so forth. It is therefore an important aspect of each implementation to carefully plan the topology and internal structure of ALM projects so they best fit the requirements of a given organization.
Proper project topology has a profound effect on the reporting, user management, administrative tasks such as backup/restore etc.

The purpose of this document is to assist ALM customers in assessing their current development and testing practices and successfully defining ALM project creation methodology. All aspects of this process have been researched using best practice data and expertise from various sources including OpenText operating system administrators,

professional services organization, technical documentation, books from industry experts and personal experience of many customer testing organizations. These guidelines will help reduce in initial creation time and achieve maximum value in operating the ALM.

## Audience

This guide is intended for:

- Vice Presidents of Applications
- Business Analysts
- Testing CoE Managers
- Testing Automation Engineers
- Development Managers

## Prerequisites

To use this book, you should have a good acquaintance with major phases of Software Development Life Cycle (SDLC). You should also be familiar with the business processes in actual IT organizations.

Operational knowledge and administrative privileges of ALM are essential in implementing these best practices.

Note: some of the features discussed in this document are available only in Application Lifecycle Management.

# Structure

This guide is organized as follows:

- Introduction to Project Topology
- Planning Project Topology
- Project Topology Examples
- Other Aspects
- Conclusions

# 1 Introduction to Project Topology

## Importance of starting right

Most users do not work directly with databases, networks, servers or storage systems; they work with applications. Yet, traditional IT management is oriented toward managing the individual components that applications rely on. When application problems arise—and they always do— IT much search through silos without a clear idea of what or where the root cause might reside. ALM comprises tools and processes for keeping IT professionals informed about the quality of critical applications, and enabling IT to address the problems. A key ALM objective is to bring IT professionals closer to a "single pane of glass"—that is, one complete view of all relevant information about how an application is performing. Achieving such a goal with ALM could allow IT to be more productive, effective and responsive to users' problems. ALM is also about providing IT with an "end-to-end" understanding of application quality, performance and availability. This
means starting with the user's experience and moving all the way through the components that make up a large application. This view helps IT get past the conundrum in which individual systems seem to be working properly but users are still dissatisfied with their experience.

If you decided that ALM is the way to break those silos in order to deliver best possible quality, then the next step would be to concentrate on the implementation plan that includes careful allocation of the resources according to the company business processes and divisions.

Implementation of any testing tool or process requires much thought and planning, and no matter how good the product is, it will quickly become useless unless planning starts during the sales process and extends well beyond the software delivery date. Therefore, whether your company is an existing customer or a prospective customer, it makes sense to assess your current processes and needs according to the industry best practices. This paper should help you with defining the methodology so the very first steps of project creation are taken in the right direction. You may also consider working closely with OpenText and its partners to make sure your requirements, questions, and concerns are understood and addressed at every stage.

Effective implementation of application life cycle requires a systematic process. When your team manages multiple projects, each with its own cycles and priorities, it's easy to lose focus. Almost no organization can dedicate a testing team to one development project. At best, a testing team is part of the development group for a particular product. Additionally, you need to validate multiple functional areas, builds and baselines, different platforms and environments, and a vast number of integrations. To track multiple test cases, your testers need a process that allows them to manage multiple projects and clearly define the objectives of each one.

"Do it right or do it over" is the compelling slogan to convince company management to allocate time to properly plan ALM topology based on projected growth rate.

# Variety of approaches

Now more than ever, enterprise success depends on applications that achieve their desired business outcomes. If your IT organization lacks the ability to consistently deliver high-quality applications, your business is at risk.

No company is the same and therefore business processes vary quite significantly. This is reflected in the way application development and testing is performed. Prior to rolling out an ALM, it is required that you conduct comprehensive analysis of the quality ecosystem in place at your company.
You should analyze your environment, interview your personnel, assess current process maturity and product adoption, and review supporting documentation to define the approach that will suit your business needs best.

As you assess your company maturity level and amount of actual development projects being carried, one important decision is to decide what ALM project (database with resource repository) represents. This decision will have a profound effect on the way the project is managed, how requirements are written and the approach that is taken to test the application.

While ALM project is different from *Project Management Institute* definition of a "*project*", it features some very similar attributes such as definitive start and end, scope, milestones etc.

## Single vs. Multiple

The first step in planning is to decide whether you stay with single project or divide data and activities between multiple ALM projects.

Sometimes the company decides to maintain all activities of all teams inside one rather large project. This type of implementation may indicate the desire to keep everything in one place and separate various types of entities by granting certain privileges to certain users. Alternatively, such projects may be dedicated to one of the various ALM activity types, for example, a defects-only project or requirements-only project for all kinds of the applications.

While there are quite a few companies that chose to go this path, in most of the cases OpenText would **not advise** implementing this scenario. There are quite a few reasons:

- Contradiction with project management paradigm

  Projects should have clear start and end – if you put an entire portfolio of application development into a single project, you may lose the insight and focus.

- Access management burden

  It is nearly impossible to assign proper privileges to each and every requirement, test folder, defect, which creates the situation when everyone can access everything.

- Too much dependency

  Having all data in one project may affect maintenance tasks such as backup/restore and upgrade path – performing these routine procedures becomes nearly impossible due to interdependency and involvement of multiple teams.

- Data filtering is hard

  Not only managing permissions of requirements or tests tree is hard, filtering data in various lists cannot be easily accomplished.

- Difficult to get current status per project

  As everything is bundled together, reporting on overall quality status of the specific project is troublesome.

- Performance issues

  As usage grows and more development projects get on ALM to manage their quality needs, the amount of data kept in a single database may cause performance degradation.

In OpenText view, using multiple ALM projects brings some clear **advantages** to the usage model:

- Better structure

  We recommend examining granularity of the project to get optimal size for the content according to the purpose.

- Better user management

  When dealing with one project at a time, it is much easier to define the necessary privileges for folders and certain entities such as requirements, test sets, defects and the like.

- Better administration

  If you consolidate data assets between multiple projects, you are free to schedule regular maintenance tasks such as backup/restore and apply project-based timetable to ALM upgrades.

- Better data filtering

  Inside the project that is dedicated to one purpose – be it application, model or line of business – only relevant informational resources are kept.

- Better compliance

  Using multiple projects model helps achieve the compliance level required in many industries.

- Better performance

  Since each of the projects is smaller, both database performance and repository performance are better. Dividing a big database into smaller, more manageable parts also allows using advanced I/O features that modern RDBMS support thus improving the response time. File system- based repositories can also enjoy increased access time if they are spread between multiple physical storage drives.

ALM is designed from cross-project reporting and metrics point of view and therefore the request for an all inclusive mega project is irrelevant. ALM allows great flexibility in mapping software development processes – hence it is imperative to analyze them prior to implementing the tool.

## Industry Effect

It is well known that why no company is not same in terms of business processes, requirements and reporting, there are some strong similarities in the way the business is done in certain industries. Financial, insurance, automotive, pharmaceutical, software vendors – each of these sectors have common patterns of behavior, common usage models, common testing principles.

For example, in automotive industry the defining factor in choosing new project would be a model year. The next year model may be based on the same chassis and most of the components are also the same, but there are always modifications that alter car appearance or behavior. Therefore, the start of new model development would be accompanied by creation of the new, year-based project.

When cellular providers test mobile phones from different vendors, they usually have a separate project for each major phone model. This way all the requirements, tests and defects pertaining to the specific handset are stored together as long as the phone is being sold by the provider.

Independent software vendors (ISVs) base their development lifecycle on the major version. Hence the start of the new version of the product spawns a new project. This new project contains majority of the requirements definitions and test sets while test execution and defects data is not copied. This approach allows starting from the last point where development stopped in the previous version and concentrate on the new features. The older project is generally backed up and is used for patch management.

See later in this book for more specific examples of the topology implementation per industry.

# 2 Planning Project Topology

This chapter describes the basic steps for assessing the status of the development lifecycle and the building blocks in project topology decision making.

## What to consider

ALM implementation may be less successful if the current situation is not surveyed, evaluated and, if needed, changed. One of the most common misconceptions when purchasing a test management or even a complete ALM tool is that simple installation and minimal configuration of the tool will facilitate successful use and implementation.

### General

There are several factors that may influence the decision making process of the ALM planners.

- Assess organizational structure

  Start from evaluating team or group charged with the requirements design, code development and quality testing. For small to midsize companies, this may be a single centralized team. In larger organizations there may be separate teams. If each of the management types is handled by a separate team (with or without overlap) then aligning project topology with current organizational structure is critical in implementing ALM.

- Define reporting requirements

  One of the most important factors is what kind of the reporting including status is expected from the project team. If development project is expected to report individually then having all assets in one place may greatly simplify the task.

- Calculate projected growth

  Another common factor to be taken into consideration is when the software is going to be used and if the team or organization is going to grow in the foreseeable future. If the project is expected to grow both in terms of assets to be handled such as requirements, tests and defects and in terms of number of users, then it would probably be a good idea to consolidate all its related entities into a separate project even if currently it is not fully justified.

— Evaluate access needs

  Granting all users access to single ALM project may cause inadvertent security violations. A potential issue with unifying several development/testing projects into one ALM project is visibility to some of the test data, like Requirements or Defects. Often the actual testers are subcontractors from another company and should see only the data relevant for them. If development projects are split between multiple ALM projects, it gives better tools in managing user permissions while at the same time raising maintenance tasks level – you need to manage more than one place. It really comes down to what you feel makes business sense.

— How much customization expected

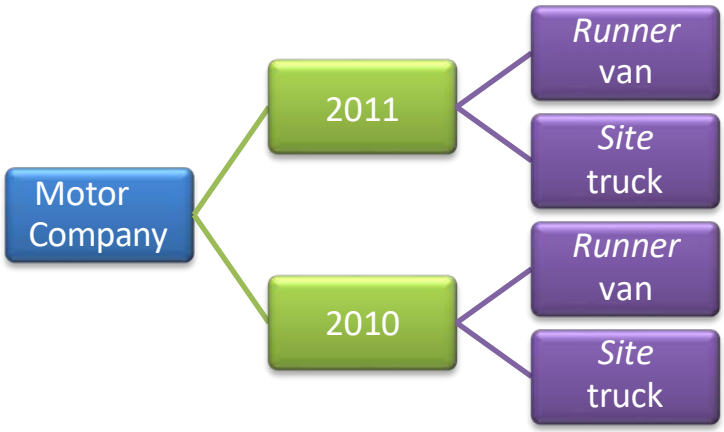
  The ability to place workflow in each module of ALM is one of its most powerful features. ALM is used to enable and enforce your process but you must define that process first. Once defined, you need to determine what level of workflow customization is required.
  Workflow can be created through wizards, or by someone who is well versed in code scripting. Plan on what customizations will need to be made to the tool to allow it to adapt to your process. Obviously, as each ALM project represents a unique set of development and testing activities, there might be specific customization needs for each project. However, it is highly advisable to establish common naming conventions, common list of values for various entities and possibly even the same set of user-defined fields and code scripts so all projects have consistent common ground. In the future this will allow cross project reporting and sharing.

## Domain

Another important factor in planning the ALM project topology is the grouping of the projects under certain domain. Companies do not bundle together different projects randomly – there is a business purpose behind the decision.

If we expand our examples from the previous chapter, then the automotive manufacturer would most probably group the projects by model year with projects representing a car brand.

For example, an imaginary *Motor Company* may have a *2011* domain for its *Runner* vans brand and *Site* trucks brand.




The previous year models, grouped under *2010* domain name, also has *Runner* vans and *Site* trucks, but their requirements, test cases and defects come from the corresponding, previous year.

In the world of cellular providers, domains are based on the equipment supplier so that Supplier N domain contains projects per its current models. ISV would pick product name to be the name of the domain where all recent versions and new version are maintained.

An additional benefit of aligning ALM projects with the functional area or domain is the ability to assign an administrator for certain domain so he/she would apply expertise in this functional area and drive the underlying projects.

Technically speaking, if you divided projects between multiple domains, you can better manage storage location of the associated file system repositories thus allowing easier maintenance and increased performance.

## Development Approach

One of the factors to consider during evaluation and planning is the development approach. To respond to the demands of a hyper-competitive marketplace, IT departments today are tasked with increasingly diverse responsibilities. They must support global, 24x7 operations and integrated supply chains while quickly delivering applications to market.

Most existing applications were developed using classic, so called *waterfall* approach. In the waterfall approach, each team may develop separate, unrelated features and deliver them at the end of the process for final integration. Their new versions and patches would probably continue this way.

With proliferation of modern, *agile* approach, it has become custom to see new initiatives or autonomous modules moving in this direction. Sometimes companies start on the smaller scale and transition pilot teams to work in agile fashion.

Agile teams would want liberty in applying agile manifesto principles to the collection of epics, work items and tasks they usually maintain.

Whatever the reason, ALM provides the capabilities to support both approaches. If certain development team wants to continue working in waterfall, you just maintain their dedicated project and they create artifacts as they go.

As both methodologies are supported, it is recommended that organizations create some common milestones for both approaches. This way reporting can be done across projects regardless of the methodology chosen.

Then there are quite a few teams that choose a *hybrid* approach – some steps resemble waterfall, some steps are clearly from agile world. Create a separate ALM project and let them use the most suitable arrangement.

ALM provides tools to achieve project objectives by including entity sharing between projects being developed *in parallel*, regardless of what approach is taken thus allowing quick synchronization of changes and traceability.

## Compliance

Since many industries are heavily regulated and must pass a variety of compliance-based tests, it is yet another important parameter to consider especially when these regulations are updated regularly.

Many organizations—especially those in the finance, healthcare, and government sectors—are required to comply with specific government regulations such as HIPAA, Sarbanes-Oxley, US Section 508, and many others. It is therefore obligatory for their IT divisions to demonstrate adherence to the highest level of regulatory compliance.

Since compliance requirements are usually stay without change until their new version arrives, it may be a good idea to store them in a separate ALM project which would serve as a repository and "plug" into the corresponding development project where execution data resides. When repository is modified, this change is propagated into corresponding development project. This way your company can comply with the regulators - keeping the compliance data untouched while generating the required audits accordingly.

## Initiatives

Mature IT organizations often adopt an approach called business initiative. An initiative is a set of applications that are developed to support a common business goal. An example of an initiative is "Improve printing" that may include replacement of network printing software, installation of new generation of multifunctional printers, development of printouts adjusted to these new MFP devices etc. To support this initiative, several new applications may need to be developed and a large number of the existing software applications may need to be modified.

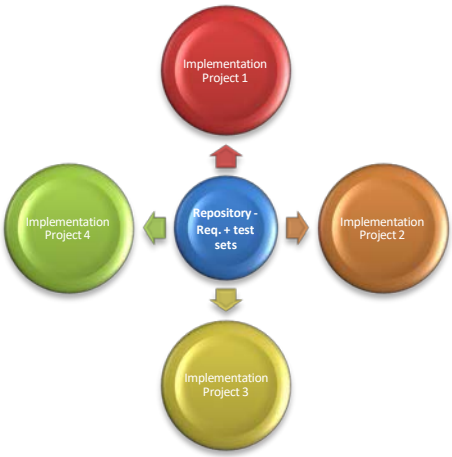
The scale of these developments and modifications could not be managed if each application team works alone in their respective silos. Instead, all requirements management, development, testing, deployment and change management processes need to be coordinated across multiple applications. With each application team introducing their own requirements, tests, components and defects, the key to a successful release of an IT business initiative is to enable visibility, coordination, and collaboration.

In most of the cases, organizations choose to maintain initiatives in separate ALM projects thus achieving higher levels of consistency.

# Star-based Project Topology

As there are quite a few parameters to consider when designing ALM project topology, one specific type of projects layouts may be a best fit for complex enterprise development projects.

We **recommend** using star-based configuration where there are a central repository project usually containing definitions and satellite implementation projects with execution data.




One of the common usage scenarios primarily aims to provide reusable compliance requirements and generic test case definitions through sharing in companies that need to address regulatory rules that change over time such as Sarbanes-Oxley, HIPAA, COBIT, and many others. When a compliance requirement needs to be updated within the repository itself, the Compliance Officer (or someone fulfilling that role such as ALM Administrator) makes the modifications within the repository project. The changes are applied to the implementation project using synchronization - a changed icon and alerts let the project owners know they should resynchronize the requirements.

Another typical scenario is to create a project with reusable requirements and test cases for one-time usage. It is often used as a repository for common requirements and artifacts that span multiple projects, for example, common GUI interface test set, common database or security requirements, performance requirements and so on. This type of project assets gets imported from the repository project into the implementation projects.

See *Entities Sharing Best Practices* book in this series for more practical details on how to most effectively use sharing between repository and implementation projects.

The common ground for these example is that global testing assets such as requirements and test cases are designed in one centralized repository project and are transferred into satellite implementation project that utilize them.
The actual test execution data is saved in implementation only, thus making global assets – usually regulatory-bound entities – unaltered until next wave of modifications.

There are many benefits to this type of project topology:

- Better access management

  You may limit access to sensitive global assets in repository while allowing necessary changes in the satellite projects.

- Better approval process

  If you keep information that requires approval in the repository, your company's review and approval process may concentrate on central repository only. This includes limiting number of users in the repository, special permissions to the approvers, visibility of approved items, easier access to the assets by management (they need to login into one central project instead of many), easier notification to the stakeholders etc.

- Better reporting

  Once you know which projects use certain requirement or test case, it is much easier to report requirement coverage status of the implementation projects. In some cases, such as in regulated environments, it may be a legal necessity.

- Better governance

  This approach enables lean and efficient set of projects that do not contain unnecessary data. Each project contains the data relevant to its purpose and repository serves as the common definition ground and "point of contact" for all satellite implementation projects.

Many large enterprises build Testing Centers of Excellence (CoE) that unequivocally choose this topology model for their operation because of the stated benefits. Numerous analyst reports attribute the importance of CoE to the significantly enhanced ability of an organization to meet or exceed the goals that the center supports. When governance, a support structure, guidance, measurements, and shared learning exist across an organization, success is far more likely. A successful organization can focus on specific projects goals. With proper project topology of the main ALM tool in place, it is much more likely your company excels in delivering better outcomes.

# 3 Project Topology Examples

In the previous chapters we emphasized the importance of project topology and its profound effect on ALM position inside the company. We also discussed the main factors that may affect the project topology decision and a recommended star-based topology that covers multiple implementation scenarios.

Majority of ALM deployments occur in IT organizations. Since no two IT shops are identical, the decision on how to build project topology depends largely on considerations such as the company culture, business processes in place and previous tool limitations (if any). Let's see examples of project structuring in detail.

## Application

This is the common case. Each project represents one application, with all of its requirements, business models, KPIs, test sets, test resources, defects and reports. This is the natural way to map development activities, to manage relationships between business analysts, developers and testers. For example, Billing, CRM and Portal are widespread names of projects and these are dedicated to their namesake applications.

In most of the cases, the focus is on application which has consecutive releases. Therefore many customers perform a backup of the previous release's project and restore under new name for the next major release. Hence we may find project *Billing 5.0* that is converted into *Billing 6.0* when all development and testing activities of the former are finished and the latter becomes a current project. In this case, the older version enters a maintenance state, and the new version moves through development milestones towards a *release*.

There are some clear gains in employing this topology type:

- Good visibility
- Easy control of all related activities and processes
- Simplified access management
- Effortless restore from backup when needed

## Model

Hardware manufacturers have processes similar to those in other areas – after all the development and testing concepts are the same and objectives are equivalent. ALM is implemented in various types of companies.
Sometimes a project represents a model (physical entity) of the product, such as a cell phone, printer or TV set. In this case too, a new project is created when new model development is started.

## Platform

If development and support activities are organized around some major "platforms", then structuring the projects in the similar way will be just natural. Examples of platform projects include splitting activities:

- by computer operating system

  Windows servers vs. Windows desktops vs. Linux vs. HP/UX

- by smartphone operating system iOS vs.

  Android vs. Windows Phone

- by infrastructure framework

  Hadoop vs. Hybernate vs. GWT

- by common device part

  Laptops vendor with Intel and AMD processors

- by automobile chassis or engine

and so on.

As long as this topology maps actual business processes, its advantages are self-evident:

- Common requirements
- Component-based testing helps in removing redundancy and improves progress tracking
- Entities sharing is done within one project

## Initiative

This is also a common usage pattern found in many IT organizations. For example, suppose there is a plan to add support for new version of internet browser. As the requirements are mostly specifics to this browser, this initiative is translated into the series of changes of existing applications.

To develop this new browser support, a new ALM project is opened to cover all aspects of planning, coding, testing and bug tracking. Integration of existing components and/or systems usually falls into this category too.

The main benefit of the approach is the focus on the activities directly related to the initiative thus making it easier to report progress.

## LOB

In a world of huge companies and endless mergers and acquisitions, one Line of Business (LOB) may be quite separate from another or just maintain independent sets of data, thus making the case for this kind of implementation.

The frequent scenario is when each LOB maintains one project where it keeps all applications or compliance information. In bigger organization, such grouping may go one level up to the domain – each domain representing one LOB with its specific projects.

While it may sound as support for the silos within the company, the business sense is the main factor to be considered – if different LOBs matters do not intersect, then having them separate would be a proper project topology decision.

## Internal Service

In some companies, there might be an internal organization which provides services to all users regardless of their LOB association – as opposed to the LOB only activities (see above). This organization may work with many different applications which are getting tested, improved and deployed but are grouped and supported together to provide a holistic customer experience. An example of department within organization may be *Messaging* which covers all spectrum of messaging applications – from Outlook to Outlook Communicator to Exchange Server to centralized fax service to Short Message Service Gateway. Other departments may concentrate on voice and

data communication, databases, company-standard desktop software and so on.

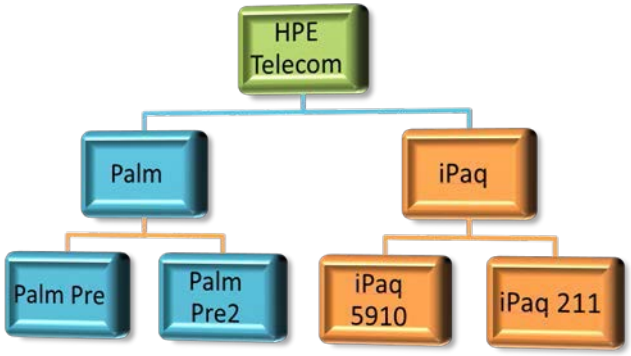
Another example of internal organization that serves the entire enterprise is Centers of Excellence. Many IT organizations have adopted the Center of Excellence (CoE) model as a practical way to consistently and continuously improve IT operations. A CoE provides the entire organization with visibility into standardized quality metrics and key performance indicators (KPIs).
This helps to keep all stakeholders informed, and keep applications aligned with business objectives. Many companies formed Performance CoE, Security CoE, Six Sigma CoE and the like.

It would make a lot of sense to consolidate the internal service activities inside one ALM project thus setting basis for company-wide report on object of service.

## Service Provider

There are numerous examples of ALM implementations on service provider sites. Service provider areas of operation are different – be it cellular operators, cable and satellite TV suppliers, internet providers and the like – but they all have a common ground of offering their services according to some business model. For example, cable TV supplier may organize its work by region of service, whereas internet providers usually break their activities by type of internet infrastructure or speed. Cellular operators divide their customers by business type – private, small-medium or enterprise as well as by other parameters such as mobile phone vendor.

Therefore the imaginary *HPE Telecom* provider would build its project topology based on vendor-specific domain with its related mobile phone models. Hence we would see *Palm* domain with *Palm Pre, Palm Pre 2* etc. models as well as *iPaq* domain with *iPaq 5910, iPaq 211 Enterprise* etc.




When new phone model is unveiled, ALM administrator would open new corresponding project to concentrate all relevant activities, such as regulatory pollution, radiation and frequency requirements, test cases and subsequent defects.

# 4 Other Aspects

In addition to the project topology planning and implementation, there are some other aspects of ALM planning process that need to be addressed in order to better utilize its capabilities.

## Folder Structure

Folders are the container of the assets stored in ALM. Therefore organizing them in the best possible manner would simplify the initial deployment and forthcoming usage.

Here are some tips that would help in everyday work:

- Naming conventions

  Folders and test sets should be named according to the defined company policy. Test set names should be unique so ALM does not calculate its results together. Consider having prefixes in the name of the test set, such as <test set name>_build5.1

- Hierarchy

  Based on functional area or feature, it should address certain part of the whole system and provide clear distinction for users. For example, one would build requirements tree in the following manner:

  - Common Requirements
    - Product-specific Requirements
      - Functional
    - Feature 1
    - Feature 2
      - Non-Functional
        - Performance
        - Security

- Categories

  You may consider adding standardized user-defined field (UDF) to folders to categorize them. It is helpful in filtering and reporting.

## Template Project

If we take the concept of naming conventions one step further, it is a good idea to coordinate user-defined fields, lookup lists, workflows, customizations according to company-wide policy. Plan in advance for common terminology for:

- Priorities
- Severities
- Test Phases
- Test Environments

and use throughout the entire ALM implementation.

Then there is a concept of cross-project customization that allows the ALM administrator to define the most effective workflow and key project attributes in a template project, link it to the projects that need to comply, and automatically propagate it to all projects that are linked to the template.

Since template projects are created per domain, it allows for great flexibility in actual implementation – as per selected project topology approach. For example, if projects are structured around LOBs, then it is recommended to create a template for each line of business which would contain its specific customizations. If the workflow changes or the process requires a new set of fields, the changes can be made only to the template project. All projects linked to the template can easily be synchronized to pick-up the changes.

Template projects provide visibility into the status and quality of all projects and enable entity sharing and a number of other critical features and concepts in ALM. We recommend using cross-project customization to make sure that the fields used in all your projects have common definitions, making it easier to import and export libraries. Only the fields that have common definitions can be shared across projects. Although template-based projects have a common customization, ALM is flexible enough to allow each project to still keep its own individual customization as required.

# 5 Conclusions

The ALM space is being driven by time-to-market issues, while seeing an increased need for governance and regulatory compliance. Although many companies begin by looking for the tool, most are in need of improved practices and guidance. Issues such as where should we focus, what roles are essential or how can automation be effective are at the root.

ALM meets the needs of the modern application lifecycle by providing increased alignment between teams, including integration into strategy and planning teams, an offering of best practices to spur innovation and prevent tactical delays, and a bridge to the critical last mile of the operations organization. ALM is a unified platform designed to master the application lifecycle from end to end.

All of us are interested in improving the quality of the applications we produce on behalf of our respective departments, lines of business and enterprises. What's required to make this goal achievable is a lifecycle approach to quality, structured planning and design of the environment, consistent use of best practices, state-of-the-art tools such as ALM and intimate knowledge of company business and "political realities."

The properly structured ALM projects that map real business processes bring unrivaled advantages to the organizations such as:

- Rework reduction

  Through entities sharing and reuse, companies maintain centralized libraries for managing requirements, test scripts, and test resources, change and configuration management, and keep track of shared entity traceability including execution history.

- Always-on reporting

  If ALM implementation resulted in creation of multiple projects according to the recommendations, then built-in feature of cross-project reporting provides customizable visibility into application projects, trending analysis and insight, dashboard, and aggregate data and metrics for quality, requirements coverage, defect trends in order to assess the overall quality and readiness of the release.

- Addressed business demands

  With process standardization described in this document, companies can address the definition of standardized lifecycle process, enforce consistent best practices, maintain centralized customization of project templates, and execute an automatic update push to projects at change time.

- Freedom in administration

  Once assets are spread between multiple projects, ALM site administrator is free to schedule maintenance tasks such as backup/restore on project basis without the need to stop the entire operation.

- User management under control

  Multiple projects may have different sets of users with different privileges granted. From ALM site administrator point of view, it allows assigning of a project manager to handle access to the specific project and grant permissions and therefore limit project content exposure. From the end user prospective, be it a business analyst or a developer or a testing person, it allows to concentrate on issues at hand and removes burden of unrelated entries.

These are just some of the benefits of properly implemented ALM project topology. We hope that best practices listed in this document will help better utilize ALM usage in your organization.