Qualität ist kein Zufall*

*Qualidade não vem por acaso

Autor: Alain Fagot Béarez

Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

(Lack of) Definition

- There is no industry definition of what constitutes and what does not constitute an ALM tool.
- The OSGi framework's application life-cycle management defines how a bundle is installed, updated and uninstalled.
- The IT Infrastructure Library (ITIL) defines five core management topics:
  - Service Strategy
  - Service Design
  - Service Transition
  - Service Operation
  - Continual Service Improvement

From Wikipedia

- Application lifecycle management (ALM) regards the process of delivering software as a continuously repeating cycle of inter-related steps: definition, design, development, testing, deployment and management. Each of these steps needs to be carefully monitored and controlled.

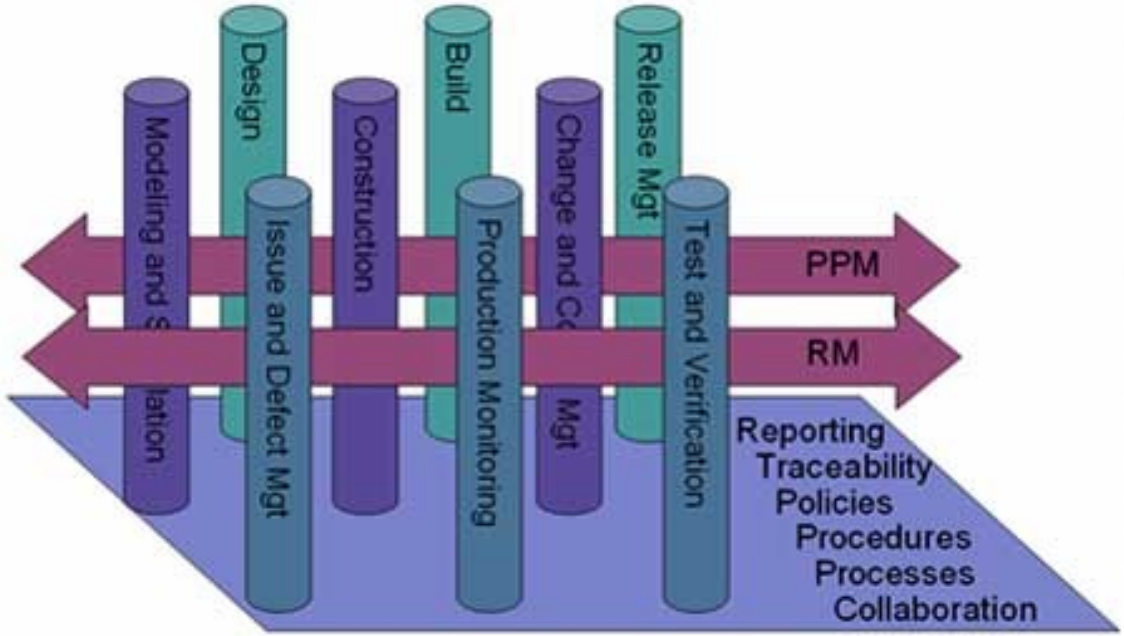
From ALF Project

- Application Lifecycle Management (ALM) is the set of activities required to support application development from initial inception through deployment, support and system optimization.
- In most cases organizations support ALM through a heterogeneous set of tools, myriad distinct processes and one-off integrations used to pull together the tools and processes across the organization.

Application Lifecycle Management

- define
- design
- develop
- deliver

- the right product
- the right way



von der Idee bis zur Ablösung

- Requirements Visualization
- Requirements Management
- Modelling
- Design
- Project Management
- Change Management
- Configuration Management
- Build Management
- Testing
- Release Management
- Deployment
- Monitoring and Reporting
- Issue Management

Technological Claims

- Maximizes investments in skills, processes, and technologies
- Increases productivity
  - as the team shares best practices for development and deployment, and developers need focus only on current business requirements
- Accelerates development
  - through simplified integration
- Cuts maintenance time
  - by synchronizing application and design

Claims About Quality

- Breaks boundaries
  - through collaboration and smooth information flow
- Increases flexibility
  - by reducing the time it takes to build and adapt applications that support new business initiatives
- Improves quality
  - so the final application meets the needs and expectations of users

Some of the Players

- Rational (IBM)
- Telelogic (IBM)
- TechExcel
- Borland
- Powerlogic

Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

Application Lifecycle Management

- Issue and Process Management
- Project Planning and Implementation
- Requirements Management
- Specification-Driven Development
- Software Quality Assurance
- Integrated Software Configuration Management
- Integrated Development and Support

A New Approach to ALM

- Product development is shifting from a tactical set of tools to more integrated solutions
- It is essential to infuse an implementation effort with the knowledge and intellectual capital of designers and high level management.
- By integrating business processes with product design, executive management can achieve more complete visibility and exercise a high degree of control over the entire development lifecycle.
- With the current trend of applications being designed by a central core team and implemented by outsourced and distributed teams, these tools become essential.

Knowledge Centric ALM

- Enterprises need a tightly integrated ALM solution to insure product quality.
- Companies need to improve product development quality using agile, iterative, waterfall, or CMMI methodologies.
- Design teams want to increase the business value of the product.
- Implementation teams are geographically distributed.
- Organizations would like to more closely align business goals with product development.
- Executives would like total visibility over the development process.

Building Success by Knowledge

- Knowledge, in the form of design documents, customer requests, detailed functional and behaviour maps, and any other collateral to support a feature, provides a platform for expressing the experience and business needs of the product.
- This platform allows for a strong foundation on which an implementation team can build a functional product.
- The framework presented by knowledge leads to a more focused development experience, regardless of methodology used.

Product Management and Product Portfolio Management

- Capture ideas and product requirements from internal stakeholders, customers, win/loss or competitive analysis and other inputs, through email routing and Web-based capture and analysis tools, easing the demands for reporting and decision-making.
- Understand customer value when making decisions about products, rather than depending on emotion or the loudest voice in the room to ensure your products are successful.
- Leverage your investment in best practices with support for key product management activities and take advantage of the ability to tailor the configuration to support your processes.

Enterprise Change Management

- Enterprise Change Management provides an efficient and systematic approach to change control, resulting in greater productivity, higher quality products and faster time-to-market. It provides functionalities such as:
  - change lifecycle management or lifecycle change management;
  - compliance support (CMMI, ISO...);
  - change request management, issue management, item and defect tracking;
  - customer information and action items;
  - product lifecycle management;
  - project management support.

System Design and Software Design

- Support comprehensive modelling of large and complex systems-of-systems, resulting in precise, easy-to-understand, and unambiguous specifications.
- Simulate the specifications to verify and validate the system, demonstrating its behaviour at an early stage.
- View textual requirements from within the design environment, quickly assessing the impact of any requirements changes.
- Correct mistakes early through comprehensive and immediate syntax and semantics checking.
- Automate the transition from design to implementation by generating executable software from the design models.

Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

Management Processes

- Issue and Process Management
  - Give the Insight
- Project Planning and Implementation
  - Removing the Hurdles
  - Specification-Driven Development
  - Robust Integration and Collaboration
- Requirements Management
  - Successful Project Completion
  - Requirements Management Process
- Software Change Management
  - Change Management Benefits

Management Processes

- Software Quality Assurance
  - Integrated Test Management
- Integrated Software Configuration Management
  - Complete Change Management
  - Software Configuration Management
  - Configuration Management Benefits
- Integrated Development and Support
  - Bridge the Gap
  - Software Delivery Automation
  - Automated Delivery Benefits
- Measure, Monitor, Improve

Issue and Process Management

- Companies need to:
  - Answer critical questions about their software development process;
  - Track issues through all stages of development;
  - Empower developers with flexible email and an intuitive user interface;
  - Enable teams to implement software following repeatable and scalable processes.

Give the Insight

- In addition to defining one project workflow, teams need to define multiple workflows based on an issue type, its status, and actions that a user can perform to move the issue to another status.
- Auto-routing allows issues to be assigned to the best resource based on the attributes of the issue.
- Administrators can define a time frame for types of issues to be resolved or windows in which work must start.
- Conditions or rules for when an email should be sent to developers let them know data they need while cutting down on unwanted emails.
- Developers can choose to subscribe to notifications for issues, giving them control over what updates or information is automatically emailed to them.

Project Planning and Implementation

- Companies do not have project planning tied to implementation management
- Executives want to use their vision, enforced by market and business goals, to steer a project.
- Project Managers need a dynamic view into their development efforts.
- Project Stakeholders demand accurate data at every level of the project.

Removing the Hurdles

- Manage features, defects, and work items in a highly organized fashion.
- Integrated knowledge communicates the requirements and intention of features and functionalities to the implementation team.
- Integrated meeting requests and events allow for intelligent resource planning and allocations.
- Familiar planning interface allows project managers and leads full visibility into the development effort.
- Fully configurable task workflows let managers see the entire history of work items, features, and defects.

Specification-Driven Development

- Enterprises need a tightly integrated ALM solution to insure product quality.
- Design teams want to increase the business value of their product.
- Development teams are geographically distributed.
- Organizations would like to more closely align business goals with product development.
- Executives would like total visibility over the development process.

Robust Integration and Collaboration

- Design documents, customer requests, detailed functional and behaviour maps, and any other collateral to support a feature, provide a platform for expressing the experience and business needs of the product.
- A specification provides a conceptual platform. This allows for a strong foundation on which an implementation team can build a functional product.
- The concepts represented by specifications lead to a more focused development experience, regardless of methodology used.
- Allow teams to follow their ideal methods, while providing the needed oversight to make sure that the processes work in conjunction with business goals.

Requirements Management

- Companies need to define and manage product or project requirements.
- Design teams want to increase the business value of their product.
- Product Managers need to confirm that product requirements will be met by the product implementation.
- Development managers need the ability to control when a change should happen, who is accountable for the change, and how that change will move the project towards completion.
- Project Stakeholders demand accurate data at every level of the project.

Successful Project Completion

- Manage requirements and life-cycle traceability through Feature Driven Development.
- Track what requirements are not covered by a development work items or test cases.
- Identify which features or functions in the design are not required internally or externally.
- Control requirement changes and view the impact of the implementation of such changes.
- Provide lifecycle traceability and review requirement implementation and validation.

Requirements Management Process

- The requirements definition and requirements management process isn't just a one-off, upfront step in the project lifecycle.
- The whole requirements process covers:
  - Requirements elicitation (requirements capture)
  - Requirements definition
  - Requirements validation
  - Requirements analysis
  - Requirements modelling
  - Requirements management
  - Requirements traceability
  - Requirements-based testing

Software Change Management

- Real-time reporting and process enforcement improve project visibility and control.
- Automated workflows and e-mail notifications enhance team communication and coordination.
- Test management unifies development and testing activities, from planning through results, for improved software quality.
- Access control, electronic signatures, repeatable processes and audit trails simplify compliance management.
- Web interface allows easy access from virtually anywhere.
- Integration with requirements, development, build, test, deployment and portfolio management tools facilitates rapid response to change.

Change Management Benefits

- Create repeatable, enforceable, predictable processes
- Effectively manage issues to resolution
- Improve visibility into projects and processes
- Jump-start implementations with out-of-the-box usage
- Quickly meet unique organizational needs
- Define and manage changes to software assets as activities for improved clarity and insight
- Manage and track global asset reuse
- Manage all development and test assets from a single integrated solution, support global test teams
- Extend traceability across the full software lifecycle

Change Management Benefits

- Access from the environments and locations from which you need
- Ensure changes are made only by authorized individuals
- Verify identities of individuals performing specific actions, help meet compliance requirements
- Trace the origin and detail of all activities
- Support evolving organizational needs
- Manage change across distributed and mainframe environments from a single point of control
- Manage change across the lifecycle, enable closed-loop software delivery
- Align project information with portfolio investments and business goals

Software Quality Assurance

- Companies are looking to standard their test management processes to improve product quality.
- QA teams are using manual and automated testing to test multiple products or projects.
- QA Managers need a dynamic view into the QA and development efforts of internal or outsourced teams.
- Project Stakeholders demand accurate data at every level of the project.

Integrated Test Management

- Provide testers with a centralized and secure tool through which they may review control documents and research previous bugs.
- Define what testing data to track and design a standard interface for collecting and tracking information.
- Make it easy for testing groups to save, manage, and reuse their test cases.
- Enable testing groups to plan future test assignments based on the analysis of previous test and defect results.
- Provide real-time access to all testing data so that testing groups can immediately respond to critical test blockers and other issues.
- Make information accessible.

Integrated SCM

- Users of Subversion, ClearCase, CVS, or other SCM systems want integration between source code and work items.
- Developers need to know why something was changed, when the change occurred, and what was affected by that change.
- Development managers need the ability to control when a change should happen, who is accountable for the change, and how that change will move the project towards completion.
- Users want to see source code artifacts affected by a feature or defect.

Complete Change Management

- Work items, defects, and tasks can all be put into a process-controlled environment.
- At defined states in its lifecycle, a development work item can require an association with a source-code artifact.
- Developers can commit their source code to the SCM system, and that action can be associated with one or more work items.
- Knowledge can be associated with work items to present a clear view of the work needed.

Software Configuration Management

- Integration with leading IDEs allows you to work in your preferred environment.
- Transparent real-time access to files and directories virtually anywhere in your organization.
- Sophisticated branching and graphical merge tools enable concurrent access and efficient use of time.
- Lightweight, feature-rich clients allow you to work locally or remotely.
- Support for open source environments provides added workspace flexibility.
- Seamless integration with software change management offers streamlined defect tracking for better team coordination and tracking of project progress.

Configuration Management Benefits

- Manage files, directories and other development assets across the lifecycle
- Gain control over personal workspaces and enable seamless access to the exact files and directory versions that are needed
- Work on the same code or release, more easily resolve conflicts, reduce confusion, and get more done in a short amount of time
- Fast access to virtually any version of any element
- Continue development efforts while disconnected from the network, easily synchronize changes when reconnected
- Define and manage related changes to assets as activities for improved insight and efficiency

Configuration Management Benefits

- Maximize development asset reuse across projects
- Optimize build times, improve reproducibility of builds
- Provide unified enterprise application development allowing developers to work either on or off host
- Access from the environments and locations from which developers work
- Allow developers to work in their preferred environment
- Control access to software assets
- Trace the origin and detail of changes made to software assets, help meet compliance requirements
- Scalability from small workgroups to geographically distributed enterprises
- Manage and control software assets across the lifecycle

Integrated Development and Support

- Manage the whole issue lifecycle from when it is reported by the customer to its resolution by the support and development teams.
- Ensure the support team and development team can work together to resolve a customer issue efficiently.
- Provide issue visibility to project stakeholders in multiple departments.

Bridge the Gap

- Allow customer support teams and development teams to easily collaborate, share data, and automate processes during the issue lifecycle.
- View each support item along with the corresponding development features, defects, and work items without having to switch workspaces.
- Dynamic updates ensure the most up-to-date information from the customer is also instantly available to the development team.
- While the development team is making progress in resolving the issue, the support team can maintain customer communication.

Software Delivery Automation

- Error log filtering and automated notifications allow rapid error detection and resolution
- Concurrent tasks and efficient use of hardware enable fast build and release cycles
- Self-documenting audit trails and role-based security simplify compliance management
- Compatibility with existing build scripts, batch files and development tools enables fast implementation
- Integrates with existing development technologies to make use of existing investments
- Self-service access to preconfigured build processes from within leading IDEs accelerates developer productivity

Automated Delivery Benefits

- Automate repetitive tasks, enable consistent, repeatable processes
- Integrate into your current environment, leverage existing investments, implement rapidly
- Run processes concurrently for increased performance
- Distribute build activities across a server pool to accelerate build cycles
- Meet individual project needs with both as-needed and continuous integration builds
- Identify and resolve build issues quickly
- Trace the origin and detail of all build and release activity
- Document the contents of each release for better reproducibility

Automated Delivery Benefits

- Get real-time, targeted information
- Improve visibility into build and release processes
- Manage and control user access
- Allow user access and administration from virtually anywhere
- Empower developers with self-service capabilities
- Enable global usage
- Coordinate and execute repeatable build and release processes enterprise-wide
- Integrate with leading source control, testing and defect tracking tools; correlate data for in integrated project view
- Integrate with third-party tools for additional flexibility

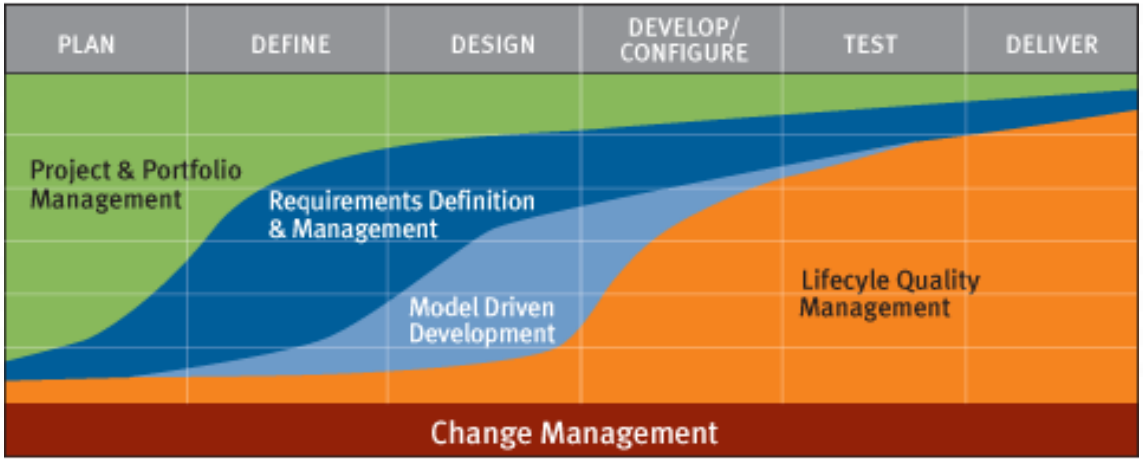
Measure, Monitor, Improve

- Give software managers quick insight into project risk, status, and trends by automating the collection, measurement, and analysis of data.
- Deploy rapidly, packaging together metrics with best practice performance measures and compliance tracking.
- Develop transparency in IT audit procedures, and a repeatable process for managing line of business applications, to help organizations achieve sustainable compliance with strict regulations.

People, Process and Tools

- Increase productivity by enabling the adoption of best practices.
- Improve collaboration by integrating processes and tools.
- Minimize risk and coordinate development with the right amount of development process for your project.
- Pass audits that require a documented systems development lifecycle that is current and in use.
- Improving technology and process together is ten times more effective at increasing productivity than improving technology alone because process improvement enhances effectiveness while development tools increase efficiency.

ALM in two pictures

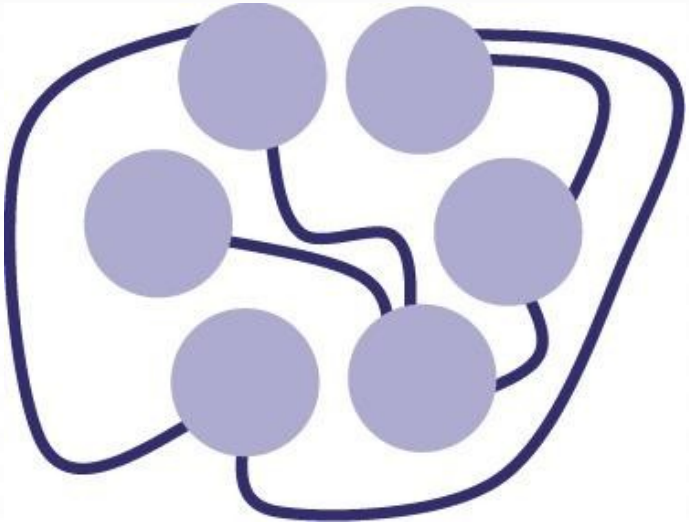


Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

Services-Oriented Event Manager

- Organizations need only
  - integrate the tools to the event management system,
  - use web services exposed by the tools,
  - use BPEL orchestrations to define integrations using loosely-coupled techniques.



Application Lifecycle Framework

- ALF isn't an Eclipse plug-in, although there are features of ALF, such as the event management GUI, that are exposed through a plug-in.
- ALF is primarily a server-side event management system that associates important events within ALM (successful build, failed test, new requirement) with BPEL processes to automate and orchestrate tools within ALM.
- ALF complements the Eclipse plug-in framework by enabling server-side asynchronous and automated integrations.

Application Lifecycle Framework

- ALF handles the exchange of information from one tool to another, the business logic governing the sequencing of tools in support of the application lifecycle process, and the routing of significant events as tools interact.
- ALF achieves this by providing a common infrastructure (SOAP Web Services, BPEL orchestration engine and the ALF Event Manager), and a set of domain vocabularies that define the events, objects and attributes.
- ALF provides various Common Services (logging, notifications, security, etc.) that are easily integrated into BPEL processes to create richer interoperability processes.

ALF ecosystem

Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

A New Industry

- The software business is still so new that it has not yet come to maturity as an industry.
- Other industries mastered the production process creating higher quality products.

- Where is the production line when you produce one thing only one time?
- How do you ensure quality when one product is as different from the next as a television is different from a refrigerator?

Software Production Line

- The software production line exists in the transition of raw ideas into software products.
- Producing software does not require a traditional rigid production line but a sophisticated and flexible framework.
- The problems with various software development processes and with certain quality control methods are threefold:
  - they are costly to implement,
  - they create vast amounts of documentation to simply certify that a company follows a written procedure for writing software,
  - they require enormous amounts of human labor to maintain and verify.

Quality Control Certification

- The processes do not guarantee that the end product is error free.
- Only that the human elements involved in the process are regulated in the hopes of reducing inefficiency.
- In *CMMI for Development*, the ultimate generic goal is the “optimizing process”.
  - “Process improvements that address common causes of process variation, **root causes of defects**, and other problems; and those that would measurably improve the organization's processes are identified, evaluated, and deployed as appropriate.”

Elements of Error Prevention

- Coding Style
  - Coding Standards
  - Design by Contract
  - Defensive Programming
  - Code Reviews
- Test Infection
  - Unit Testing
  - Regression Testing
  - Load Testing
- Infrastructure Systems
  - Source Control Systems
  - Automated Nightly Builds
  - Bug Tracking Systems
  - Monitoring

Effectiveness of EP

- Manually implementing these software development techniques does little to ensure their effectiveness.
- The automation of error prevention can solve the problem of quality in the software industry.
- Automatic error prevention tools in the software industry must be more sophisticated than their traditional production line counterparts.
- Effective automatic error prevention tools must adapt to any number of test subjects, and they must do so without human intervention.
- The difficulty in building automatic error prevention tools may explain why quality has historically been absent from the software creation process.

Automating Coding Standards

- Without automation, there is little that can be done to enforce coding standards within a development group.
- The only way to enforce coding standards in a consistent manner is to use a static analysis tool that automatically checks the code against industry accepted coding practices.

Costs of Manual Unit Testing

- Design and build test harnesses that will run the class.
- Design and create stubs that simulate external resources referenced by the class under test.
- Design and build appropriate test cases that will test as much of the code as possible.
- Perform regression testing to ensure no new errors were introduced into the code by modifications or corrections.

- The full benefit of unit testing can only be achieved by automating the process.

Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

Culture for Error Prevention

- Roles
- Group Culture
- Defensive Programming
- Source Code Review Process

- Group culture is a team's way of working together, including their shared habits, traditions, and beliefs. A positive group culture should promote code ownership, group cooperation, peer learning, common working hours, and mutual respect. The team is typically more creative, self-regulating, effective, and satisfied.

Roles

- Developer actually writes source code in a specific programming language.
- Architect designs the code in a given project.
- Project Manager delivers the project on time with the designed functionality and under budget.
- Tester/QA Personnel verifies the functionality of a system.
- Database Administrator (DBA) monitors and verifies the performance of databases.
- Webmaster monitors the performance and functionality of both stage and production servers within a web development environment.

Group Culture

- Code is the group’s greatest asset because it is the main thing that they have to show for all of their work. It also serves as means of communication: developers exchange the majority of their ideas by reading and writing code.
- Build a culture where the developers' attitude towards the code reflects the code's importance. This prevents group members from doing anything that harms code quality.
- Where group members feel a strong investment in code quality, any developer who does not care about the code will alienate himself from the group.
- The point of egoless programming is that the group owns the code, and each developer takes responsibility for the code, but each developer should not take criticism of the code he wrote as a personal attack.

Defensive Programming

- Anticipate where failures can occur and then create an infrastructure that tests for errors, notifies you when anticipated failures occur, and performs damage-control actions you have specified:
  - add assertions to the code,
  - implement Design by Contract,
  - develop software defensive firewalls,
  - or simply add code that validates user inputs.
- Detect problems that might otherwise go unnoticed.
- Prevent minor problems from growing into disasters.
- Save yourself a lot of debugging and maintenance time in the long run.

Source Code Review Process

- Exchange ideas about how the source code is written and to establish a standard group interpretation of the code.
- Identify problems and envision new solutions for previously troubling dilemmas.
- Focus on important issues such as algorithms, object-oriented programming, and class design.
- Be fun and creative: it is much more effective.

Summary

- Application Lifecycle Management
  - (Lack of) Definition
  - Knowledge Centric ALM
  - Management Processes
  - Eclipse Application Lifecycle Framework Project
- Automated Error Prevention
  - A New Industry
  - Culture for Error Prevention
  - Infrastructure for Error Prevention
  - Tools for Automated Error Prevention

Automated Error Prevention

- Source Control System
- Bug Tracking System
- Web Application Staging Area
- Connectivity Verification
- Monitoring
- Data Pollution Identification
- Automated Builds
- Software Coding Standards
- Accessibility Testing
- Performance and Stress Testing
- Regression Testing
- Unit Testing
- Coverage Analysis
- Confidence Factors

Source Control System

- Provide a central place where the team members can store and access the entire source base.
- Copy code that is already available in the repository, modify code that is already available in the repository, and add new or revised code to the repository.
- Track the history of the code but also improve efficiency by ensuring that revisions are not carelessly overwritten.
- Revert back to archived versions also enables you to take risks with your revisions and to start over again when so many bugs have been introduced that recoding is easier than debugging.

Bug Tracking System

- Record and track all errors not detected by your test suite.
- Record feature requests not yet being implemented.
- Provide valuable data about the types of errors teams or developers tend to make.
- Improve error-prevention and error detection efforts.

Web Application Staging Area

- Provide a safe zone where application modifications can be tested before they are made live.
- Establish two staging area levels: a personal staging area and a project-wide shared staging area.
- The staging area looks like the actual application but contains copies of the same components used in the actual application.
- The personal staging area lets developers start testing their work as early as possible so that they can find and fix problems before they check code into the common source code repository or have QA start testing their updates.

Connectivity Verification

- Verify that Web services and other systems connected with SOAP exchange information correctly.
- Verify the connectivity between databases, Web servers, application servers, middleware, ERP systems, CRM systems, and legacy systems.
- Isolate each module of the system and test it on its own, and then test the interactions between modules, adding them one at a time until you have tested all possible interactions.
- To test the interactions between multiple modules, you would create global test cases that impact multiple modules.

Monitoring

- Identify and prevent problems with deployed client/server or Web-based systems to ensure their continued functionality and performance.
- Cover all the pieces that come into play when a user exercises the application, including the application logic, the data back-end, the hardware, and so forth.
- Active tests simulate user actions using test drivers, virtual users, and so on to determine what problems could affect potential users' experiences.
- Passive tests unobtrusively monitor system and transaction details to identify major system problems and to collect data that helps you diagnose the source of application-level problems.

Data Pollution Identification

- Scan through the specified fields of a database and check whether the data conforms to your restrictions.
  - Streamline the database maintenance process.
  - Improve the accuracy and completeness of query and search results.
  - Increase the confidence and efficiency of all employees who rely on the database as a source of information.
  - Prevent application functionality problems that can stem from incorrect or illegal database values.

Automated Builds

- Build an application every night through an automated build system.
- Start with a clean slate by pulling all necessary code from the source code repository into an empty directory.
- Run all available test cases and report any failures that occur.
- Provide early detection of incompatible changes in the application components.
- Ensure that the application continues to run as expected and detects any errors introduced by newly integrated code.
- Help the development team work together more efficiently and encourages team members to work more carefully.

Software Coding Standards

- Reduce the probability of introducing errors into your applications, regardless of which software development model is being used to create that application.
- Offer incredible value to software development organizations because they are pre-packaged automated error prevention practices; they close the feedback loop between a bug and what must be done to prevent that bug from reoccurring.
- Ensure uniform coding practices, reducing oversight errors and the time spent in code reviews.
- Meet all quality guidelines mandated by the client company.

Accessibility Testing

- Ensuring that a Web application is accessible to people with disabilities.
- Access your application with adaptive devices such as screen readers, refreshable Braille displays, and alternative input devices.
- Two available sets of Web accessibility guidelines:
  - Section 508 Web Guidelines: Any technology produced by or for US government agencies must be accessible to people with disabilities.
  - Web Accessibility Initiative (WAI) Guidelines: A set of guidelines developed by a W3C organization committed to making the Web more accessible for people with disabilities.

Load Testing

- Exercise your application with virtual users and measuring performance statistics to verify whether the application supports the anticipated traffic load as well as possible traffic surges, growth, and so on.
- Determine what problems, in addition to slow load times and rates, might occur in different situations.
- Begin load testing as early in the development process as possible. Ideally, you want to begin performing load testing on a staging server as soon as you can exercise any segment of the application.
- Perform stress testing from different locations inside and outside the network to identify bottlenecks caused by the network and the system.

Regression Testing

- Run all existing test cases and verify that all test cases pass.
- Detect especially those faults that occur because a developer did not fully understand the internal code correlations when he or she modified or extended code that previously functioned correctly.
- Ensure that modifications did not introduce new errors into code or to check whether modifications successfully eliminated existing errors.
- Perform regression testing nightly (during automated nightly builds) to ensure that errors are detected and fixed as soon as possible.

Unit Testing

- Isolate and test the structure and function of individual units of code to resolve and prevent errors throughout all software projects.
- Test not only the functionality of the code, but also to ensure that the code is structurally sound and robust, and able to respond appropriately in all conditions.
- Performing thorough unit testing reduces the amount of work you will need do at the application level, and drastically reduces the potential for errors.

Coverage Analysis

- Coverage is typically measured either as
  - line coverage,
  - branch coverage,
  - or path coverage.
- Coverage data can be used in two ways:
  - To monitor how well your current test suite is covering the code.
  - To increase coverage by pinpointing which additional code lines, branches, and paths need to be exercised in order to achieve the desired level of coverage.

Confidence Factors

- Determine how successful your tests have been in analyzing a project.
- Understand what types of tests may still need to be conducted.
- See an increase in all statistics of the confidence factors as the project matures.
- Show at a glance if any one module that is added to a project is complete enough to stay in the final build or if it needs further work.

AEP in one Picture

Webography

- http://en.wikipedia.org/wiki/Application_Lifecycle_Management
- http://www.stickyminds.com/se/S3489.asp
- http://www.techexcel.com/solutions/alm/
- http://www.guideinformatique.com/output/index.php?act=fiche&id=837
- http://qualitypark.de/themen/requirements/requirements.htm
- http://www.powerlogic.com.br/powerportal/ecp/comunidade.do?app=comunida
- http://aelinik.free.fr/cmm/tr25_l5a.html
- http://wiki.services.openoffice.org/wiki/Defect_Prevention
- http://www.defectmanagement.com/
- http://www.defectprevention.org/
- http://FAGOT.Alain.free.fr/formations/ALM-AEP.pdf