OLAT: A Service Oriented Enterprise Platform

Dr. Alan Moran (Head of Development, OLAT)

University of Zurich[UZH]

IT Services

Architecture: Strategy and Principles

A regenerative programme

OLAT has embarked on a **long-term regenerative programme** to create an architecture that focuses on the delivery of value through services capable of providing for the needs of users in a secure, performant and scalable Enterprise environment.

Through clear architecture and standardized and improved development practices the roadmap for future development and the manner of tackling new challenges will be guided by the underlying principles that OLAT embodies.

Commencing in 2011 a series of targeted refactorings will bring about this transformation of OLAT. The first phase of this programme is described in this presentation.

Guiding Principles

Underpinning the strategic decision making within OLAT are the principles of **agile software development** and the perspective of **OLAT as a service rather than a product**.

Agile principles drive the notion of **incremental delivery of evolving solutions** together with the **acceptance of the inherent exploratory nature of the development process**.

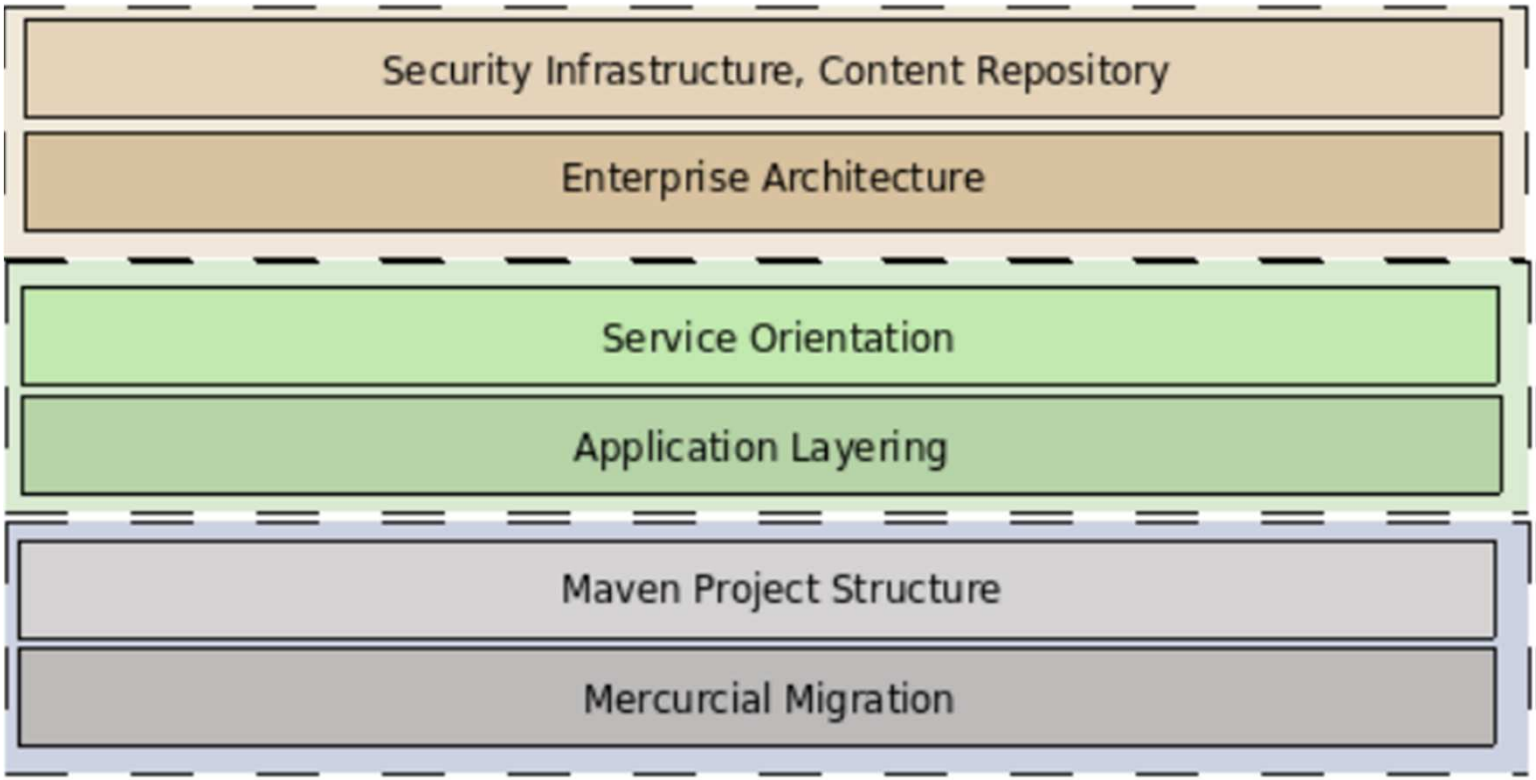
The perspective of OLAT as a service has focused greater attention on **service architecture**, **quality of service delivery** and on **integration** in the wider Enterprise context.

Best of Breed Focus

OLAT should seek to establish itself as **best of breed** in the learning environment and expose what it does best as services. In turn OLAT should be capable of integrating the best services available to it within its immediate IT landscape. This encourages a enterprise based service oriented attitude!

A service is a strategic statement of value and as such should be capable of reporting on its key performance criteria which feed into service management interfaces that form part of a wider continual service improvement feedback loop. This represents a shift towards a proactive attitude to quality and focus on the value that OLAT generates through its services.

Refactoring (Phase 1)



University of Zurich[UZH]

IT Services

Agile Development Environment

One Project, One POM

Maven embodies a set of best practices by defining a project structure that expresses a uniform build process based on conventions drawn from Java EE development at the centre of which is the Project Object Model (POM)

The OLAT development policy revolves around a **single POM** in which build and packaging, an AOP enforcement of architecture, a comprehensive and embedded test infrastructure and interfaces to third party tools are defined.

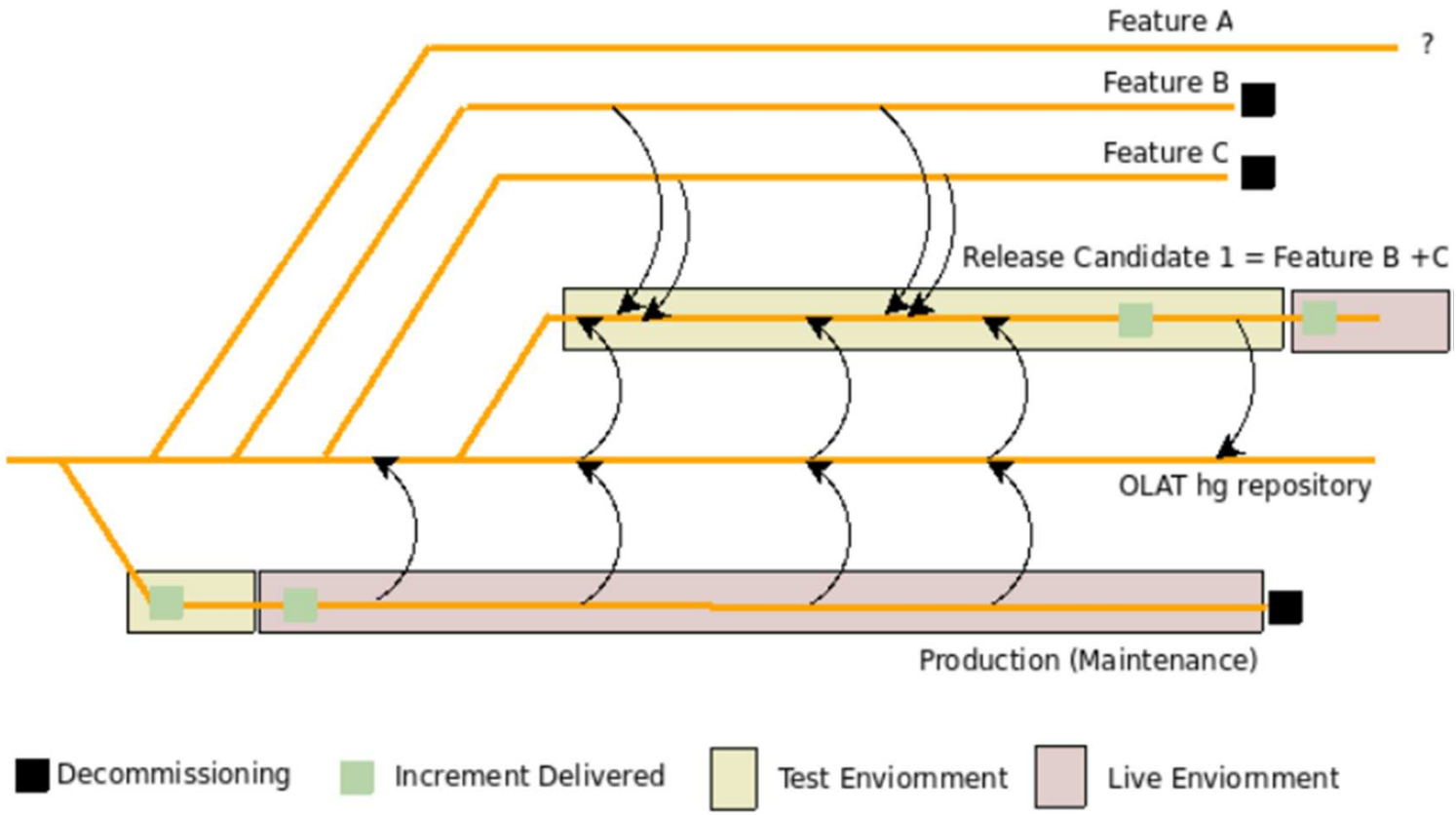
Maven dependency management is linked to the Nexus asset manager in which all OLAT deployment assets, shared resources and documentation are definitively maintained.

Feature Branching

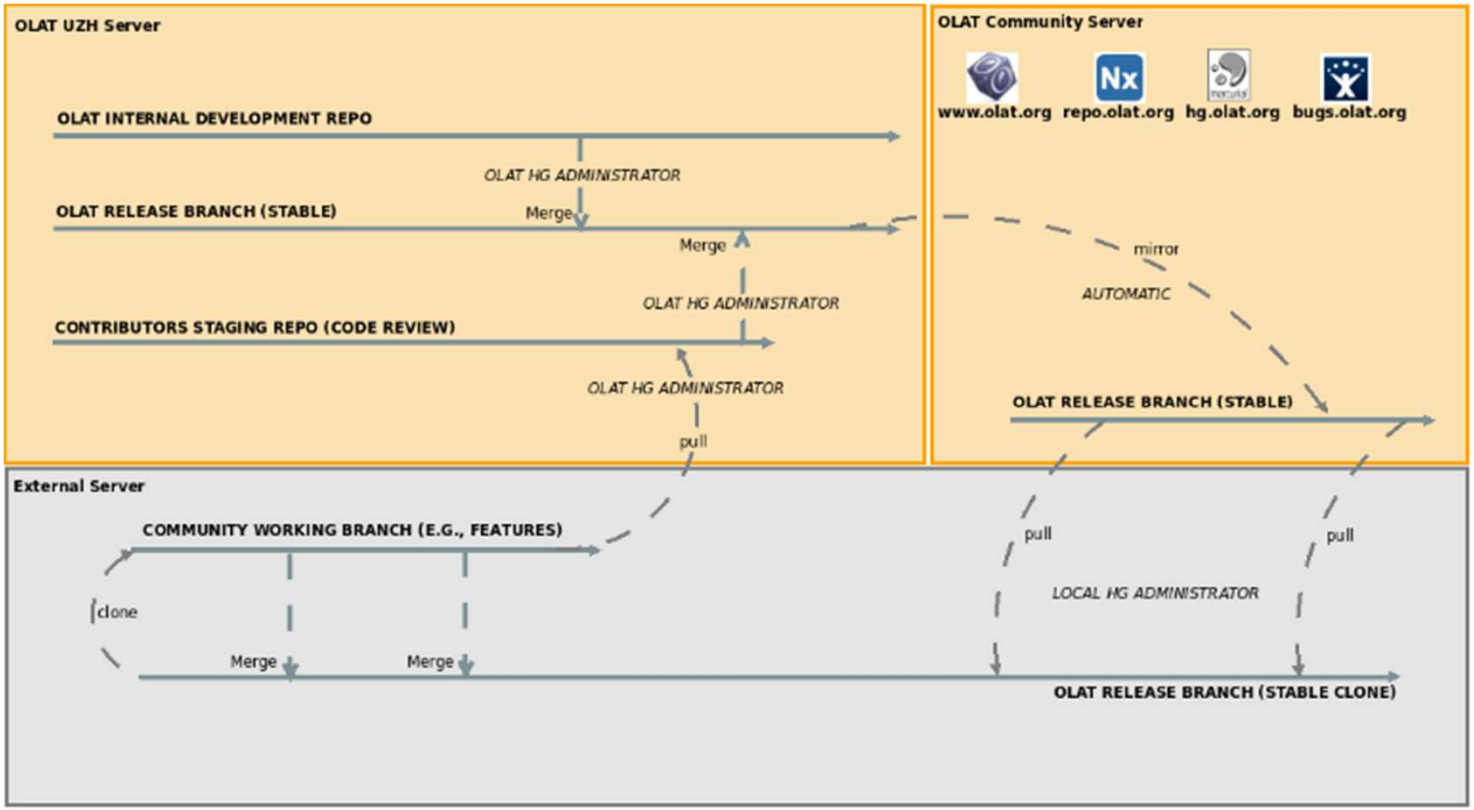
Feature branching is a common DVCS strategy that refers to the practice of isolating important changesets in repositories. OLAT promotes feature branching within the context of a dedicated continuous integration environment based on frequent merging from the parent branch. Arising from feature branches are release candidates that undperin the OLAT release and deployment policy.

This approach reflects the agile recognition of the often **variable and experimental nature of the development process** requiring isolated environments. Release candidates reflect the **incremental nature of the delivery of an evolving solution**.

Feature Branching and Release Candidates



Branching and Community



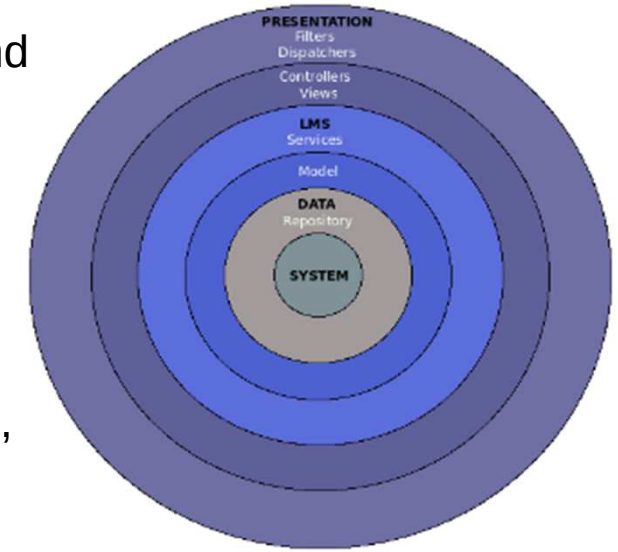
University of Zurich[UZH]

IT Services

OLAT Service Architecture

Application Layering

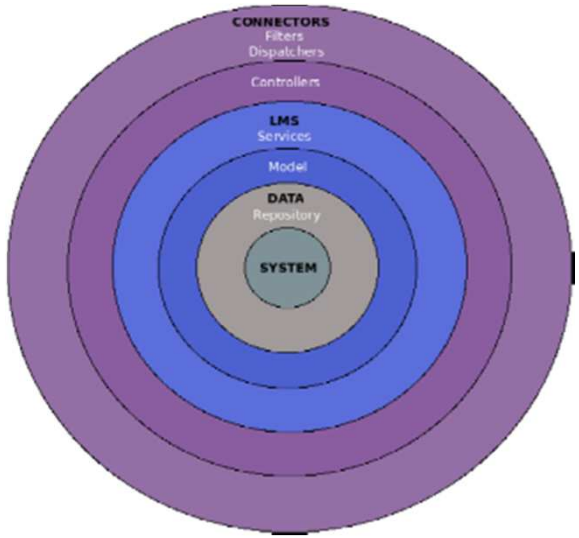
OLAT layering establishes a clear organization of the code with well-defined boundaries, **consistent cohesiveness and abstraction** within each layer and **loose coupling** to the lower layers. These are the key ingredients for a **modular architecture** that exhibits healthy dependency management and testability exposure.



Only system services have access to resources (e.g., filesystem, database, queues etc.) thereby enforcing a rigourous separation of application and system architectural concerns.

Dual Stack Architecture

OLAT envisages a dual stack architecture exposing services via „presentation“ (i.e., action and view management) and „connectors“ (i.e., protocols and proxies) layers each of which cater for the needs of the calling context.

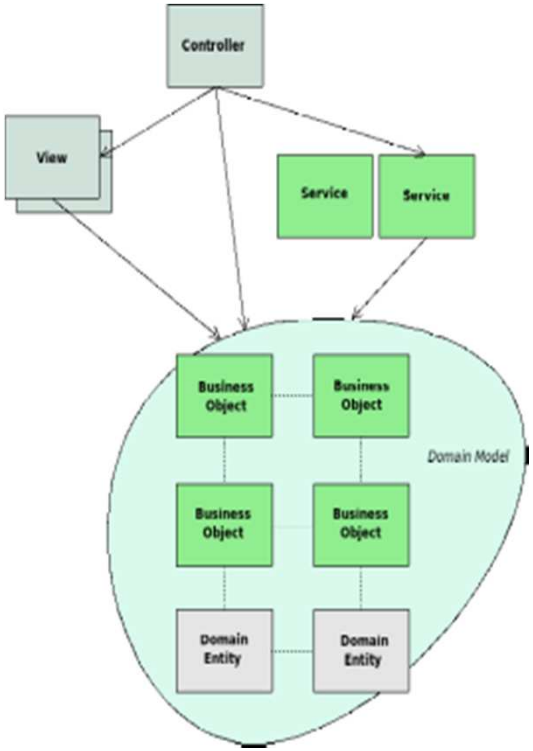


Services remain independent of their calling context and are capable of being inherently asynchronous if necessary. Invocation semantics can be interface or message based as required.

All layering decisions are AOP enforceable !

Service Delivery

Services are an architectural feature that encapsulate the value in the application as perceived by end users (e.g., enrollment, assessment, course learning etc.) Stateless and headless in nature, services are responsible for co-ordination, transactions and exception handling and expose key indicators concerning their operational health.



Hiding behind these services are the **Business Objects** and rules that define the functional units and internal model of OLAT. They exhibit an optimal degree of focus, modularity, coupling/cohesion balance and re-usability. Business objects require the highest levels of unit test coverage.

University of Zurich[UZH]

IT Services

OLAT in the Enterprise

Enterprise Perspective

OLAT envisages an Enterprise architecture that delegates matters of the deployment to the container e.g., security configuration (JAAS), service monitoring (JMX), service interfaces (JAX-RS/WS) and directory lookup services (JNDI) invoked by the application.

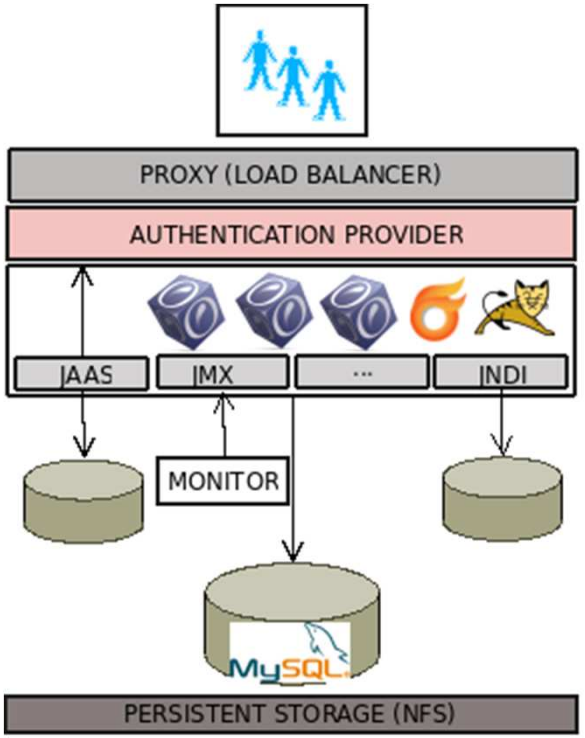
This reflects the growing maturity of an application expected to perform as expected in industry standard operational environments that increasingly assumes responsibility for scalability (incl. clustering) and application management (incl. availability, demand etc.)

OLAT will continue to support Tomcat as the container of choice for developers and for simple installations.

Some Practical Implications

Practical implications include:

- Reduced configuration reliance on olat.properties files
- More container management configuration (e.g., JAAS modules etc.)
- Fewer embedded application assumptions concerning the deployment environment
- Abstracted resource acquisition and management
- Interfaces via the container rather than the application (e.g., JMX)
- Simpler deployment of Enterprise capable services



University of Zurich[UZH]

IT Services

Thank you! Questions ?