Evergreen Computational Sciences

Automated Generation of Testable Software - Applications and User-Interfaces

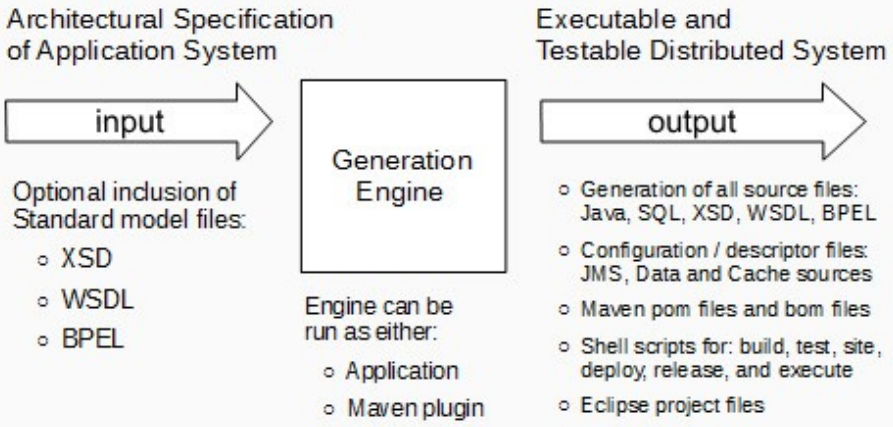
No Dependence on Hand-coding | -► Become Reliable and Predictable | -► Become Thorough and Extensible

-► Stimulate and Accelerate Experimentation and Comparison of Solutions

Java Enterprise Application Systems

We help organizations create Java Enterprise Application Systems in a practical, thorough and cost-effective way. Generated systems will run on RedHat JBoss standalone, or on clustered or cloud-ready JBoss EAP servers.

We focus on Application Architecture and Network Architecture and how to automatically generate practical and standard Java Enterprise Application Systems in a straight-forward way. We demonstrate this by example.



Unprecedented value with predictable cost

Unprecedented value comes from using our approach. For example, you can accomplish more work in a single week (with better quality) than 20 solid engineers could effectively do in one year – really. Let us show you.

Savings in costs

The savings in cost is immeasurable. The quality and thoroughness of the resulting generated application system implementation, tests, management components, and user-interface(s) far outweighs any efforts that could be done by hand.

Generation of tests

Just the generated tests alone (unit-tests, multi-level integration-tests and system-tests) provide immeasurable value that would take literally months to accomplish with hand-coding.

Demonstration by example

We will demonstrate what we mean. After inspecting, building, and running example generated systems – and after running generated tests, we believe your organization will agree – there is much immediate value to be gained and at reasonable cost.

Creation of target system

Whether starting from scratch, or from an initial code base – we work with you to develop an *initial target system*. You will see and understand how our approach can provide a concrete implementation. *This is immediate value in a short time*.

Integration with external systems

It is straight-forward to integrate with external systems such as a gateways, incoming streams, data-sources or external processing or payment systems.

Integration of hand-written code

We demonstrate how to integrate with hand-written code – pre-existing code, or empty skeleton to be filled in by-hand.

Building systems of arbitrary size and complexity

It is straight-forward to architect systems of arbitrary size and complexity. Systems may be organized in various topologies. Systems may be composed of other systems or reference other systems.

Load-balancing and Failover

Organized replication of servers and replication of services, and distribution of load makes it possible to implement different types and levels of application performance management and fault tolerance.

Usage of JBoss

Application systems are generated to run on JBoss – which is possibly the most effective way now to build enterprise applications based on the latest and most evolved technologies.

For *production environments*, generated systems will run on JBoss Enterprise Application Platform (JBoss EAP).

JBoss Technologies

Depending on the unique design and content of the target system, various capabilities will be enabled. This results in the generated system incorporating a combination of the following JBoss technologies:

- JBoss A-MQ
- JBoss HornetQ
- JBoss Data Grid
- JBoss Infinispan Cache
- JBoss Transactions (long-term as well as short-term transactions)
- JBoss Byteman - for different levels of integration-tests and system-tests
- JBoss Arquillian - for different levels of integration-tests and system-tests
- JBoss Richfaces - for implementation of JSF user-interface

Experimentation and comparison

Sample target systems of different types can be designed, generated, built, tested and *experimented with* and *compared against each other* in rapid / efficient turnaround times.

This allows an organization to try-out various potential solutions in different environments and in different contexts – mixing and match different constructs, services and communication methods.

Verification of essential behavior

The generation of software cannot be complete unless its behavior can be accurately verified. To realize this, the engine generates different levels of tests, and components that check (and test) themselves at runtime.

Integration and system tests

In addition to *unit tests*, multiple types of *integration tests* and *system tests* are generated for each module and setup to run in multiple test contexts.

Highly-specific test cases

A computational approach is used to systematically establish test cases that are highly specific to each component.

Multiple test contexts

Tests should run successfully in different *test contexts* and different environments. This is important since certain conditions and certain errors may only likely occur in one *test environment*, while ***not*** in others – such problems can otherwise be ***very hard*** to detect and fix.

Self-checking components

For runtime interactions, associated components will "check themselves" and enforce security and validate inputs and expected outputs for consistency.

Features of generated implementation

- **Well-established Architecture**
  The generated system implementation includes a powerful and appreciable set of architectural features, capabilities and best-practices.
- **BPEL and WSDL Input Models**
  The generation engine will accept input from BPEL (and WSDL) files. The engine will generate Java code and tests for logic specified in BPEL.
- **Web-based User-Interface**
  Generated for each application is a well-structured JSF 2 / AJAX based user- interface. Mobile phone UI included - future will support HTML5.
- **User and Permission Management**
  Each application has built-in security. Users, roles and permissions may be specified per application or service.
- **Management and Monitoring**
  A functional JMX *management layer* is generated for each application.
- **Testing and Verification**
  Straight-forward and advanced use of Junit, Arquillian, and Byteman for multiple levels of integration tests and system-tests.
- **Maven Integration**
  All generated modules are organized into properly configured maven multi-module project systems.
- **Eclipse IDE Integration**
  All generated modules readily import into plain Eclipse enterprise edition.
- **Dependency Management**
  This challenge is resolved using sets of maven bom files. This is an effective approach to managing dependencies and eliminating hidden conflicts.
- **Integration with Existing Codes**
  Straight-forward integration with pre-existing hand-written components.

Capabilities of generated implementation

Depending on the unique design, content and configuration of the target system, various capabilities will be enabled.

- **Service Invocation** – Several ways of invoking services are supported: Web-services, REST-services, JMS MDB, EJB, RMI, as well as CORBA.
- **Enterprise Messaging** – JMS based constructs are generated to implement specific Messaging Architectures.
- **Distributed Caching** – Various options, configuration and defaults are provided and may be customized at generation time, and at runtime.
- **Data Persistence** – JPA based data layer implementation is generated along with SQL schema. Custom relationships and queries may also be represented and generated.
- **Transactional Processing** – Generated service and process components will execute within short-term as well as long-term transactions.
- **Event Propagation** – Support for application event logging, persistence, and event subscription and monitoring is also provided.

Supporting capabilities

- **Nexus Repository Manager**
  Local and global repository managers are provided to organize and maintain versions of applications and modules. This facilitates *Experimentation and Comparison* of potential solutions.
- **Bourne Shell Scripts**
  Shell scripts are generated to enable starting, stopping, and deployment of configured sets of JBoss servers. Other scripts are provided to enable *build*, *test*, *site*, *install*, *deploy*, and *release* of modules.

About ECOS

Evergreen Computational Sciences (ECOS) is dedicated to the science and engineering of automated software generation. ECOS is a facilitator and provider of distributed system application software and related infrastructure software for companies to use on-premise or as part of cloud-based environments. We help organizations establish testable Java and JBoss based application systems in a practical, thorough and cost-effective way.

Evergreen Computational Sciences
1455 San Marino Ave, suite B
San Marino, CA 91108

Tel: **+1 (213) 379-3806**
web: **www.ecos-net.com**