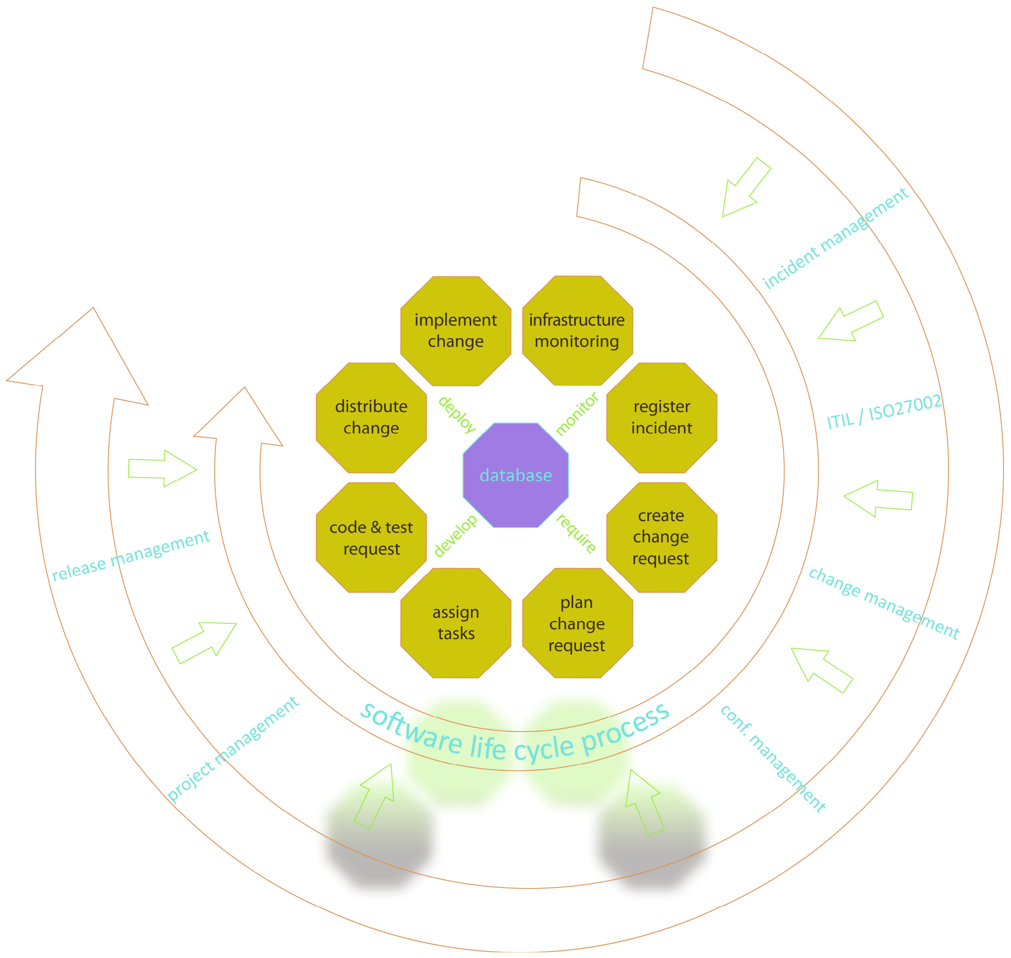
REMAIN

EGL Interface

White Paper on the importance of interfacing with modern development languages



Content

Introduction

Chapter 1

What is EGL?

EGL (Enterprise Generation Language) is a high level, modern business oriented programming language, designed by IBM to be platform independent. EGL is similar in syntax to other common languages so it can be learned by application developers with similar previous programming background.

EGL application development abstractions shield programmers from the technical interfaces of systems and middleware allowing them to focus on building business functionality.

EGL applications and services are written, tested and debugged at the EGL source level, and once they are satisfactorily functionally tested they can be compiled into Cobol, Java, or JavaScript code to support deployment of business applications that can run in any of the following environments:

- Microsoft Windows, Linux, UNIX running JVM, for example in the context of a Java EE servlet container (IBM WebSphere Application Server, Apache Tomcat, GlassFish)
- IBM System z: CICS Transaction Server, IMS, z/OS Batch, UNIX System Services, WebSphere Application Server, z/VSE, Linux
- IBM System i: IBM i5/OS, IBM WebSphere Application Server, Apache Tomcat
- Internet Explorer, Firefox, Safari browsers for Ajax rich web applications

TD/OMS EGL Interface

Chapter 2

TD/OMS EGL Continuous Integration for Java

In a pioneering effort Remain has forged an automated build environment for the build, test and deploy of EGL web services. Running EGL automated builds from within a builder server such as Continuum or Hudson instantly creates a Continuous Integration environment for EGL projects. With Continuous Integration in place, code and integration tests can be put to effective use, enhancing the overall software quality and reducing the risk of distributing failing software.

The EGL development process on the IBM i platform is fully supported by Remain's Software Change Management solution, "TD/OMS".

The automated build environment takes the responsibility of automatically checking out the EGL web service and related projects from a source repository, for instance Rational Team Concert.

The following build steps are performed:

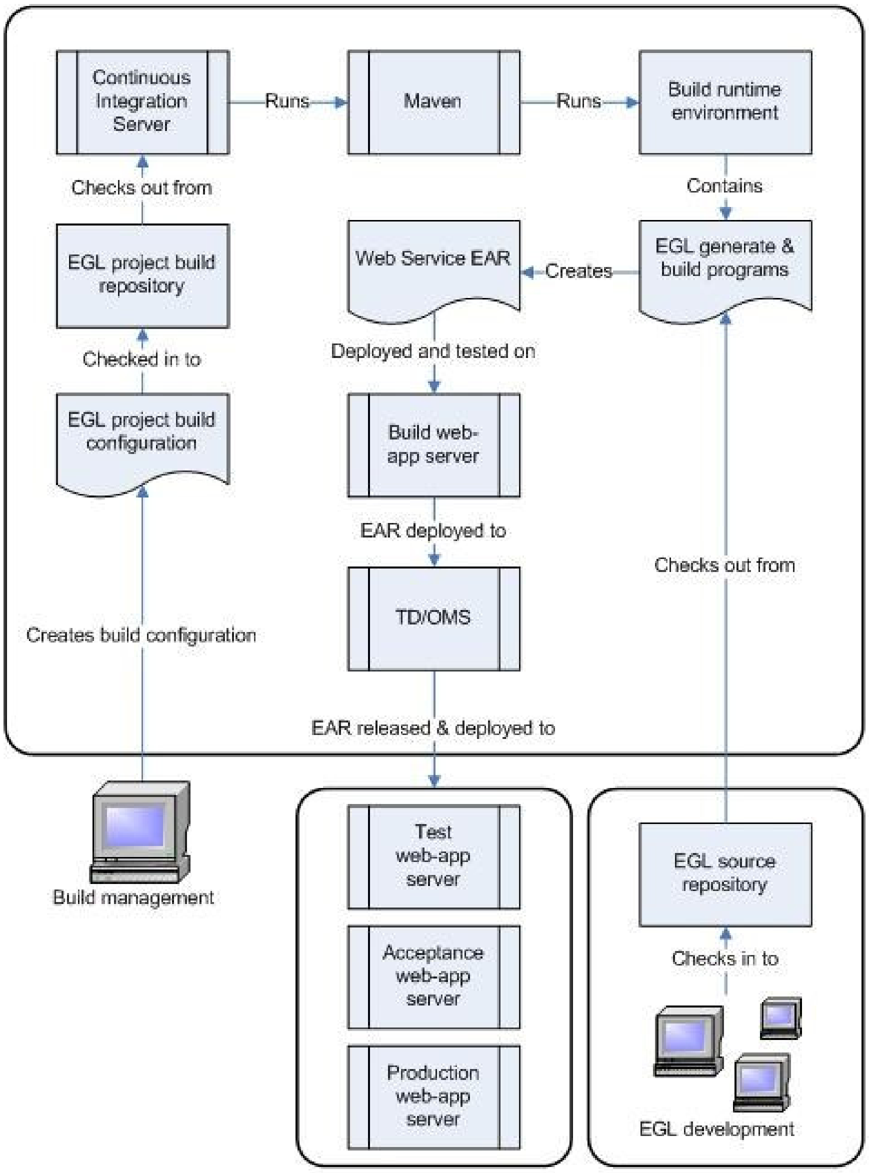
- Generating java or java script from EGL sources,
- Packaging the EGL generated resources into an EAR,
- Deploying the EAR to a local WAS server,
- Running integration test against the deployed EAR,
- Connecting the web-service EAR to a TD/OMS task for further processing.

Once in TD/OMS the web-service ear and related components, such as RPG programs and SQL files will be processed and distributed to target systems, whereby the web-service EAR can be deployed directly into a WebSphere Application Server alleviating any tedious manual handling often required for a WAS EAR deployment.

The Benefits

- Rely on the EGL automated build environment to construct the EGL web-services and to run the essential test suites automatically.
- Reduces the time normally required to release web service applications.
- Pulls EGL into a more Agile development process.

A basic setup of TD/OMS EGL Continuous Integration for Java


Continuous Integration Server
Runs
Maven
Runs
Build runtime environment
Checks out from
Contains
EGL project build repository
Web Service EAR
Creates
EGL generate & build programs
Checked in to
Deployed and tested on
EGL project build configuration
Build web-app server
EAR deployed to
Checks out from
TD/OMS
Creates build configuration
EAR released & deployed to
Build management
Test web-app server
Acceptance web-app server
Production web-app server
EGL source repository
Checks in to
EGL development

Interfacing with other languages

Chapter 3

Interfacing with APIs

An application programming interface, or simply an API, is a set of program instructions and norms that allow computer applications (software) to communicate directly with each other. APIs often form the separation between the different layers of abstraction, allowing applications at a high level of abstraction to function and outsource the lesser abstract work to other programs.

An API defines the access to the functionality behind it. The outside world does not know the details of the functionality or implementation, but thanks to the API the functionality can be used. One advantage of this is that multiple deployments can be accessible with an API, as long as they conform to the API.

Streamlining of the Change Management process is possible only when data is available from every available application in the market and applications not yet built. Interfacing makes this possible. TD/OMS supports interfacing with all major languages.

TD/OMS and Interfacing

TD/OMS is built from the ground up to integrate with existing processes and technologies. TD/OMS connects seamlessly with 4GL languages such as LANSA, AS/SET and IDDOS. Furthermore, TD/OMS integrates fully with Eclipse and IBM WebSphere to deliver a robust platform for software development. Lastly, TD/OMS integrates with the Open Source tools Mylyn, Subversion and BIRT.

TD/OMS manages everything that can be seen as an object. In addition to PC objects, messages and documents, for example, this also includes objects derived from 4GL tools. Remain provides 4GL interfaces for all 4GLs working on IBM's Power Systems: LANSA, AS/SET, CA:2E, CA:Plex, IDDOS and Magic.

The Remain Software Roadmap

Chapter 4

The future of TD/OMS

In general, our software environments grow and become more complex. Because of this complexity, more functionality must be added to applications in order to cope with issues such as distributed and remote development, change control, process support, web support and more. To remain competitive, organizations must be able to change strategies, rapidly adapt to new market conditions, and quickly introduce new products and services.

Remain Software recognizes that businesses have made significant investments in time and money in order to adapt their original applications. The solutions that Remain has developed help customers to develop new applications or modernize their legacy applications. In order to anticipate the needs of customers and the market, Remain uses new methods and a clear roadmap for the development of new generations of the TD/OMS Software Change Management solution.

IBM i Roadmap

Remain Software is committed to the roadmap of the IBM Midrange Systems (Power Systems, formerly System i, iSeries and AS/400). Therefore, it is very important for Remain to closely follow IBM's strategic developments in the IBM Power Systems Portfolio.

Project Gravity – Multi-Platform Framework

Parallel to the dedication to IBM's Power Systems, Remain Software is working toward the first official release of a ten-year development effort that has been given the code name Gravity. Gravity will divide target platforms into modules and bring them together into one uniform Change Management Framework. All of our current technology has a place within this project, with the first version to be launched in 2011. Project Gravity is based on Eclipse, Service-Oriented Architecture (SOA), OSGI, and is multi-platform and multi-database.

About Remain Software

With more than nineteen years of experience and expertise in the Software Change Management field, Remain Software has built an independent and dynamic Software Change Management platform for heterogeneous software development environments.

Remain B.V. (HQ)
Dukatenburg 82b
3437 AE NIEUWEGEIN
The Netherlands

Tel.: (+31)30 600 5010
Fax: (+31)30 600 5019

info@remainsoftware.com
www.remainsoftware.com