Component Oriented Development And Assembly (CODA) with Equinox

Executive Summary

Eclipse Equinox simplifies the process of developing and deploying modern software solutions, a process we call Component Oriented Development and Assembly (CODA). Rather than carving down large, complex, feature-rich stacks to suit your needs, Equinox provides a lightweight, component-based method of building stacks tailored to your needs. More than just a framework and runtime, the Equinox community provides many core components such as data access, SOA tooling, and other server-side functions, along with rich desktop and Ajax UI facilities. In short, Equinox allows you to stop coercing monolithic, of-the-shelf stacks to fit your world and, start designing and assembling stacks to fit your project's needs. The result is a flexible and reusable component and programming model that spans your application's tiers and layers. This simplification gives you the power to more rapidly adapt your existing software assets, integrate them with other systems, and create new function to meet the demands of new markets and opportunities.

Equinox and CODA is a proven component model that has been used by the Eclipse to simplify the integration and development of developer-oriented tools. The Eclipse community is now extending the technology and concepts to allow software developers to use Equinox as the runtime platform for applications that run on different platform and tiers.

The Thick Layer of Complexity

The world of software development has evolved into a powerful but complex world of computing tiers, distinct deployment scenarios, specialized environments, and integration frameworks required to glue together these different layers into cohesive solutions. For example, Java on the desktop, mobile Java, and enterprise Java employ different development models despite all being based on the Java language. New applications types, such as Software as a Service, mobile, and Ajax, introduce new technologies to the list of consideration when delivering software. Integrating these layers together from project inception, to development, to delivery, and maintenance has become the most difficult and expensive part of software development.

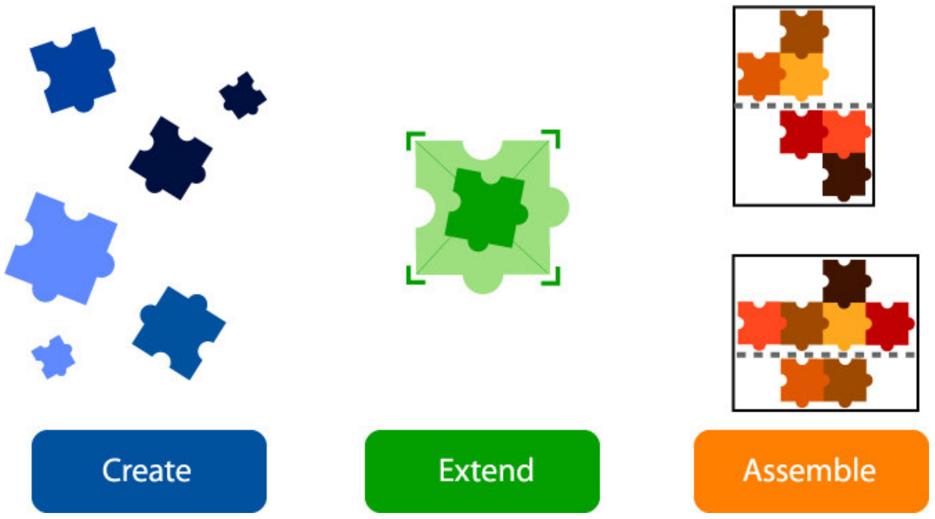
At the same time, customers and partners are demanding faster responses to their requests. Software providers can no longer take the 'one size fits all' approach – more flexibility is required in software development and delivery. Software developers need a model that delivers customized solutions to meet customers' desires – a 'customer of one' approach that enables adding, updating, and tailoring features for specific situations.

More than just being highly customizable, software applications must also integrate with third party technologies and services. For instance, integrating CRM systems with different e-mail servers and services is standard in successful business deployments. Value in the IT world comes from making this integration of diverse technologies more than just possible, but natural. Unfortunately, there are often too many ways to integrate systems and services. Vendors define their own integration mechanisms specific to their solutions while industry standard solutions are often too general, requiring proprietary extensions and knowledge. This leaves development teams and third parties with the responsibility of understanding the different integration schemes and correctly using them to accomplish their goals.

Simplicity and Flexibility with Equinox

Today, the primary technical hurdle for software development is increased complexity. Part of the problem is the lack of a common component model. Properly addressing these thick layers of complexity is key to accelerating the development and deployment of software modern systems. The standardized, lightweight component model provided by Eclipse Equinox is a solution to this problem.

Equinox enables a development process in which software is developed and delivered using a common component model and runtime. In the Equinox enabled process, software components can be developed by a wide variety of organizations. These components may then be further customized and assembled to create tailored end-user solutions. We call this process Component Oriented Development and Assembly (CODA). CODA is both enabled by Equinox and core to the Equinox philosophy.



The basis for CODA is an industry standard, called OSGi, and the Eclipse runtime platform called Equinox. Over the past 5 years, Equinox, OSGi and the concepts of CODA have been used by the Eclipse community to simplify the integration and development of developer-oriented tools and end-user Rich-client applications. Equinox is now bringing this simplicity and flexibility to the wider software development world to address the ever-growing collection of application tiers, deployment options, and integration challenges. Equinox provides one methodology that normalizes the decomposition of code into components and the collaboration between these components within, and across, tiers. The result is a system that has the feel of a tier-less architecture.

Equinox facilitates dividing your code-base into different bundles and provides a runtime for collaboration between those bundles. Bundles may be small and precise, or large and encompassing. How exactly you decide to decompose your code is determined by collaboration requirements, grouping of like or unique services, required dynamic deployment options, and other concerns such as clustering. The end result is an architecture that is well-factored, whose parts work together to deliver a unified whole.

The idea of specifying units of like-functioning code exists in the Java language itself. Indeed, the language and associated Java standards provide many ways to group code: packages, access modifiers, and grouping mechanisms like JARs, WARs, and EARs. This wide array of options allows teams to decide how to mix and match the various packaging schemes. While a wide variety of architectural and design choices is handy it can result in every project choosing a different packaging method, resulting in proliferation of different practices that can be time consuming for development teams to keep up with.

Equinox's scheme of dividing code into "bundles" provides a self-describing, highly expressive and manageable way of packaging code. API’s and dependencies are explicitly defined, enabling bundles to be shared with other projects internally and industry wide. In addition to packaging project components, Equinox facilitates inter-component collaboration through its service and extension point mechanisms. Finally, Equinox provides deployment tools and a runtime for execution. By providing a runtime, Equinox manages the full life-cycle of bundles and the resulting software by dynamically starting, stopping, updating and extending components without requiring an application reboot.

Key Features of Equinox

Bundles
The fundamental component of Equinox is the OSGi bundle. Bundles are self-describing, versioned components. Think of them as JARs with dependency and API descriptors. Bundles can include code and/or resources that can be installed, updated, and uninstalled while the application is running. Typically, a bundle supplies a discrete unit of function to be used by other parts of the system.

For example, the Equinox HTTP bundle provides an extensible web server component that implements the standard OSGi HTTP service specification. With this bundle, applications can host servlets, JSPs and other web resources.

Dependencies
As mentioned earlier, bundles describe their API and dependencies. Rather than leaving you to manually ensure that your bundle’s dependencies are satisfied, Equinox manages dependencies and the development environment for you. The tooling for Equinox also supports componentized

builds and advanced provisioning features not available to native Java systems. Put simply, Equinox's component orientated tooling and runtime eliminates classpath and JAR-hell.

Services and Extensions
The combination of OSGi's service and Equinox's extension facilitates the service-oriented collaboration between different bundles. The service mechanism provides a familiar publish/subscribe model for services within a given runtime. Much like FireFox's plug-in architecture, Equinox extensions expose extensibility points allowing other bundles to contribute data or functionality. These mechanisms can be used separately or in concert. For example, in the HTTP case mentioned earlier, the application bundles can collaborate with the HTTP bundle programmatically or declaratively to serve up web content.

Either way, Equinox allows you to clearly and explicitly define and limit how your bundle collaborates with others.

Runtime
At runtime, Equinox supports and enforces the CODA concepts described in this paper. The dependency resolution and classloading management that Equinox performs allows multiple versions of the same code and libraries to co-exist and execute. Equinox's runtime also allows you to hot-swap bundles, allowing you to add and remove bundles dynamically without restarting the application. A key characteristic of the Equinox approach is that the dependencies, lifecycles and modularity expressed in the bundles is enforced at runtime. Service access enforcement at runtime further strengthens the API boundary and, in turn, the abilities to re-use and compose groups of bundles.

Flexible Deployment of Equinox into Existing Infrastructure

Most organizations have substantial investment in their existing infrastructure and application code. The good news about Equinox is that it's easily deployed into an existing IT infrastructure.

Equinox can be deployed as a WAR file into a Java EE application server. Server-side applications can then be composed from a set of bundles running beside applications written as EJBs, servlets, or JSPs. Systems can take advantage of the high availability and scalability features provided by the application server environment while benefiting from the flexibility and reuse that CODA and Equinox brings.

Equinox can also be deployed as a stand-alone runtime directly on an operating system or device. The same Server-side application bundles deployed to the Java EE application server can be deployed and run as bundles directly on the stand-alone Equinox runtime. This write-once, deploy anywhere functionality is another key benefit of Equinox.

Who Is Using Equinox and OSGi?

Equinox is proven technology and is deployed on millions of computers.   For the last 5 years, Equinox has been the heart of the Eclipse platform and has dramatically changed the way the software industry develops and deploys developer tools.

The exciting opportunity is that organizations are now using a similar approach to modernize their software infrastructure technology.   For instance, the major application server vendors (IBM, BEA, RedHat JBoss, Oracle, and OW2) are enhancing their software with OSGi and Equinox. New special purpose servers, such as the BEA Event Server, have been developed using Equinox to address the specialized needs of event driven applications. IBM Lotus has adopted a platform strategy based on Equinox and Eclipse RCP for their Lotus Notes, Expeditor, Sametime and Symphony product lines.   Enterprise organizations like NASA and Deutsche Post are also using Equinox and OSGi to optimize how they deploy their client and server infrastructure.

The Benefits of Using Equinox

The job of writing and maintaining software will never be easy, but it can be much easier. Equinox directly addresses today's core issues of complexity in software development: the lack of a standardized component model for code packaging and collaboration, and, the need for a runtime layer that can begin small and grow larger as needed. Because Equinox's CODA methodology is based on the mature OSGi standard, projects that use Equinox avoid proprietary, custom lock-in. Even better, Eclipse Equinox is itself open source, ensuring the stability, availability, and innovation an application's core architecture.

Applications using Equinox have both the flexibility to change and the ability to re-use components in new ways – that is, the capacity to address diverse and dynamic customer requirements quickly and without starting from scratch. Because mistakes are easier to address and prototypes are faster to create the IT lifecycle becomes shorter.

Applied to business, these benefits allow you to use existing assets in new ways, improve developer productivity, and achieve faster entry into new markets. Equinox's common, CODA packaging and collaboration methodology opens up existing assets for use without having to write and maintain custom integration code. Development itself is more productive because of a range of time savings from having one, common architectural scheme, to faster build and deploy. The combination of increased re-use and faster development allows you to more quickly address new customer demands and markets as they emerge rather than having to go through the time consuming and frustrating experience of creating a new stack each time.

Getting Started with the Equinox Community

The Eclipse Foundation has launched an initiative to help organizations adopt Equinox, OSGi, and CODA into their software infrastructure. A new top-level project, called the Eclipse Runtime (RT) project has been created to help foster and grow this community. Eclipse RT is a significant supplier of runtime oriented technology that organizations can use to build their unique software solutions. Eclipse RT will include open source projects that provide Ajax support (Eclipse RAP), a SOA Runtime (Swordfish), a persistence service (EclipseLink), technology to allow deployment to embedded devices (Embedded RCP) and more to come in the future. The wider Equinox and Eclipse community also includes runtime technology, based on Equinox, for enterprise reporting (BIRT), modeling (EMF), and data access.

To get started and better understand the concepts in this white paper, we suggest that you take advantages of the following resources:

1. Demo of Equinox in Action
2. Two Webinars: Getting Started with OSGi and Introduction to Eclipse Equinox and OSGi
3. Complete the Introduction to Equinox Tutorial