

# Asset Locator – an Enterprise Development Resource Management System

Alex Akilov, Iftach Ragoler, Avi Yaeli, Gabi Zodik  
IBM Haifa Research Lab, Matam, Haifa 31905, Israel  
{akilov, ragoler, aviy, zodik}@il.ibm.com

Ken McClamroch  
IBM Research Triangle Park, 3039 Cornwallis Rd. RTP, NC, 27709  
klmclam@us.ibm.com

## Keywords

Reuse, impact analysis, resource management, knowledge management

## 1. INTRODUCTION

Throughout the software life cycle, developers produce an enormous amount of software artifacts, or 'development resources' such as design, code, documentation, test suites, and so forth. Although these are some of the most important assets in software organizations, little has been done to leverage the information that exists in these resources. Previous attempts at managing development resources to solve the reuse problem failed for the most part because they required a large investment in infrastructure, introduced additional overhead, or relied on radical changes to the software process and programming habits of developers.

Enterprise Development Resource Management (EDRM) is a new paradigm for the management of development resources in the enterprise. EDRM leverages the various information assets that exist in the enterprise and applies enterprise-centric management to the information in order to provide developers with Search and Reuse, Collaboration, Impact Analysis, Knowledge Management (KM), and other capabilities. Unlike previous attempts at managing development resources, EDRM is based on an automated mechanism which integrates with existing development processes and systems to provide a low-cost, low-maintenance solution.

## 2. INFORMATION SOURCE IN THE ENTERPRISE

What are the information assets and how can they be leveraged? Take for example, a Java source file. The source file typically contains javadoc comments that describe the class behavior or parts of it, along with semantic information describing the super types, declarations of class members, and the implementation of the class. This semantic information provides additional input regarding the class functionality. Names given by developers to language entities such as class or method, also tend to describe the entity to some extent.

Configuration Management (CM) systems, or other repositories in which development resources are stored, are themselves a source of information. They contain information about the location of resources and dependencies between components, as well as information about the developers and how they work with the resources.

EDRM suggests a way to leverage these information resources for various uses. Consider, for example, a Java developer who needs to develop a servlet that accesses a DB2 database. He will want to know if something similar has already been developed in the organization or how to contact other developers with experience in servlets and databases. This type of information is actually hidden inside development resources and configuration management systems. Such capabilities are of special importance for large and distributed software organizations where just knowing what exists, where it is located, and who is doing what, is a tremendous challenge.

## 3. EDRM

EDRM is comprised of the following major phases:

- **Discovery** – involves discovery of information sources from legacy repositories and systems, and collection of relevant information about them. This type of activity is complex because the information may be distributed among multiple locations and may exist in various formats. To make the discovery process profitable, it must be performed with minimum user/administrator interaction and include incremental discovery of new resources. If established organizational processes exist where users contribute information sources (e.g., component library), these should be collected as well.
- **Analysis** – entails extraction of textual and semantic features from the discovered development resources and meta information from the domain repositories. Identifying the "correct" semantic features and how they can be leveraged is essential for the success of the EDRM solution. Additional processing can also be applied at this phase, and may include cataloguing or categorizing resources into predefined domain taxonomies.
- **EDRM central repository** – lies at the heart of the EDRM solution and stores the extracted features and the analysis results. The repository should be scalable to support large amounts of data and support both structured and unstructured searches. It should also be kept in sync with the discovered domain as information sources are constantly being added, deleted, and modified in the domain.
- **Global analysis** – provides additional types of analysis that take into consideration a global view of the entire domain; this is done once the EDRM repository has been populated. Global analysis can include finding relationships between

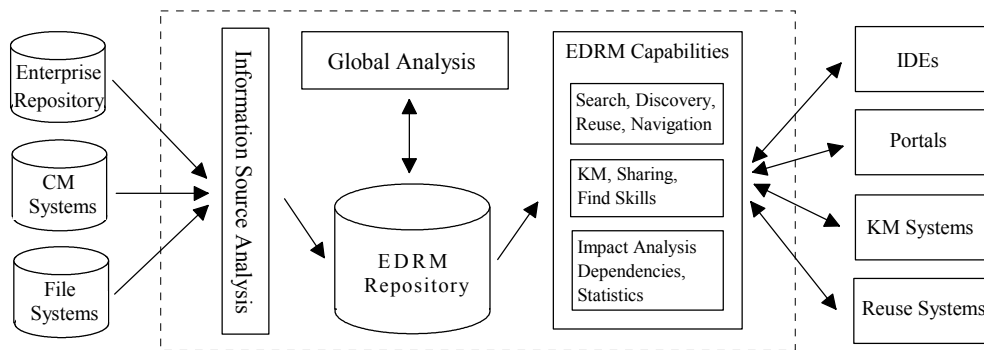
development resources, finding duplications, identifying dependencies, accumulating statistics, performing data mining, enforcing coding conventions, and more. The results of the global analysis are also stored in the EDRM repository.

- **Implementation of EDRM capabilities** – provides services that will leverage the information accumulated in the EDRM repository and transform it into EDRM features that can be utilized by developers and/or by development tools. For example, this may include a search interface for finding reusable code, navigation interfaces for exploring the enterprise repositories, and more. As users begin to work

with the system, usage profiles about the users and other runtime information such as statistics, can be recorded in the EDRM repository and leveraged for other types of activities. The tools must be open and extensible to allow easy integration of the capabilities into existing tools and processes.

- **Integration of EDRM capabilities into existing systems and processes** – integrates and ‘sews’ the EDRM capabilities into existing processes and tools such as IDEs, Portals, KM systems, etc.

The following diagram illustrates the major components and processes of an EDRM solution:



#### 4. THE DEMONSTRATION - ASSET LOCATOR

We developed Asset Locator — a low cost, low maintenance, extensible, and scalable solution— which represents a first step towards achieving EDRM. Asset Locator uses a set of autonomous crawlers (WebDAV, ClearCase, TeamConnection, file systems) to discover development resources in enterprise repositories and CM systems. A set of domain specific analyzers (Java, C++, COBOL, XML, HTML, JSP, Java ClassFile, Text) analyze the discovered resources and populate a centralized repository based on DB2. Asset Locator is built as an extensible development framework which enables it to crawl into new and proprietary domains, analyze additional resource types, and integrate its EDRM capabilities into any client application.

We will demonstrate Asset Locator and some of the EDRM capabilities it provides, including:

- Semantic search server enables developers to search for development resources and improve component/code reuse. The search capabilities have already been integrated into several tools and IDEs.
- Automatic categorization of development resources into domain taxonomies improves search and understanding, and enables navigation of the repositories according to domain taxonomies.
- Global analysis finds relationships across Java resources and enables graphical navigation over these relationships.