AbstractALM

Framework

Version 1.0

Author: Michael Olschimke (michael@olschimke.eu)

Table of Contents

Introduction

The Problem

Today, there is a variety of version control (VC), software configuration management (SCM), and application lifecycle management (ALM) tools available. Each system has unique features and addresses unique clients. Therefore, each solution provides a unique SDK for integration with other tools in the lifecycle, such as IDEs. To take advantage of all the features that are provided by the vendor, the usage of the SDK is obligatory.

On the other hand, some tool vendors are only using a small subset of the features. For example, a text editor with ALM integration might only need to check out files from a specific repository, edit the file, and check it back in. The problem begins if the developer of this editor needs to provide integrations with various VC / SCM / ALM tools. Since each solution has its own SDK, all SDKs from all solutions to be supported needs to be integrated into the application.

There is already a partial solution for this problem: Microsoft's SCC Interface (Microsoft SCCI). The drawback is that the interface is closed source and all of its components (libraries, documentation) are only available to Microsoft Visual Studio Partners. In order to get the libraries and documentation, the signature of a non-disclosure agreement (NDA) is required. Tools based on the SCC interface are not allowed to publish their source code.

Another problem is SCC's limitation to version control features. The API is not supporting other artifacts of the software development lifecycle, such as change requests, requirements, etc.

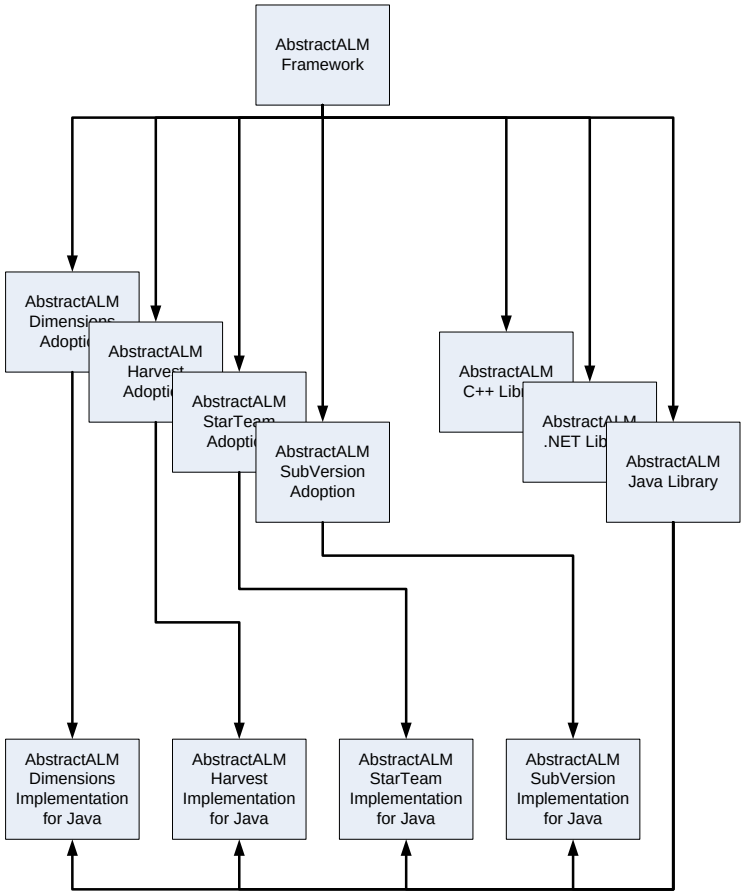
The Solution

AbstractALM tries to solve the problems introduced with Microsoft's SCC interface:

- It is Open Source (published under the LGPL), therefore not requiring a NDA, but also not requiring publishing the source code of the using application. Therefore, it can be used in any environment, both open source and commercial environments.
- It is available to the public. All documentation, binaries, and source code is available at www.AbstractALM.org.
- It is open to other artifacts than file revisions. While not implemented yet, AbstractALM can easily extended to support such artifacts as change requests, requirements, test cases, and whatever else.
- It is open to other solutions. While the initial release of AbstractALM supports some vendors, other vendors are encouraged to support AbstractALM by adopting the framework.

In addition, AbstractALM is platform and language independent. This is reached by various abstractions of the framework:

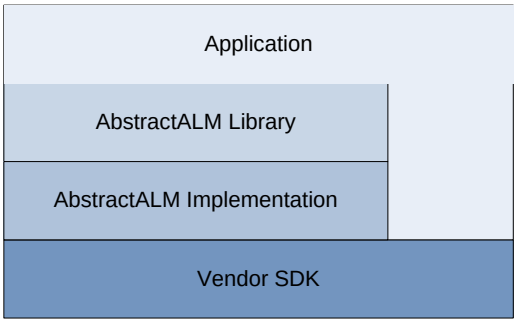
1. **AbstractALM Framework (this document)**
   The framework defines what AbstractALM is, which artifacts and features are supported by AbstractALM, etc.
2. **AbstractALM Adoption**
   For each supported proprietary solution, another document is required. The document describes how the AbstractALM is adopted to the solution, e.g. how artifacts are mapped to the proprietary artifacts.
3. **AbstractALM Library**
   The AbstractALM is a language/platform dependent implementation of all required interfaces. For example, the Java interface IFile is part of the AbstractALM Java Framework.
4. **AbstractALM Implementation**
   The vendor-specific implementation of the AbstractALM Framework. It contains adapters to access the vendor library, classes to connect to the repository, etc.
5. **Unit tests** for implementations complete the framework.



However, AbstractALM is not replacing the vendor-specific, proprietary SDK:

- AbstractALM is only a very limited subset of the SDK. Its goal is to abstract very common features, such as the check-in and check-out process of specific artifacts.
- AbstractALM is only supporting very limited administrative features, such as creating new users. Since the architecture of SCM solutions is very different, it is not possible to manage projects, views, processes, etc.

In fact, AbstractALM is still using the vendor SDK. The library uses the vendor-specific SDK to access the SCM solution and execute operations as required by AbstractALM. The library also provides access to the instance of the SDK (Connection.NativeSDKObject), therefore providing both an abstract interface for common tasks and a native interface for vendor-specific tasks.



The degree of which the application uses the vendor SDK depends on the functionality and the application's level of abstraction.

Connections

Each implementation shall provide a driver to connect to the SCM solution. The driver is instantiated by a connection manager in the library. Since every SCM solution requires different connection parameters (such as host, port, protocol, project, view, package, repository, folder, etc.) the driver shall use connection strings to identify the desired artifact location.

Connection Strings

A connection string has the following general format:

```
alm://driver-ident/driver-specific/driver-specific?param=value&param=value
```

The following table describes the sections of the connection string in detail:

Part	Description
alm://	Indicates that the string is an AbstractALM connection string; obligatory
driver-ident	Identifies the driver to use. The identification is language and platform dependent. For example, it could be the name of a DLL, assembly, or Java package. There is only one driver-ident section. Example: `org.abstractalm.subversion`
driver-specific	Identifies the artifact location. There can be multiple driver-specific sections. Example: `svn-server:1234/calcApp/trunk`
param=value	Additional parameters required for connection. Each connection string can have multiple parameters. Each parameter is uniquely identified (using the text param) and has exactly one value. Example: `user=olschimke&password=blah`

The complete string could look like this:

```
alm://org.abstractalm.subversion/svn-server:1234/calcApp/trunk?user=olschimke&password=blah
```

Common Parameters

The following table lists common parameters:

Name	Description
user	The name of the user to connect with.
password	The password of the user to connect with.

Common parameters are also available as properties in the driver manager, for example to securely set the password of the connection.

Objects

This section describes the objects (such as the artifacts) that are supported by the framework. Each object contains of 4 parts:

1. A reference to the native SDK object. If there is no native SDK object, this reference might be null. This reference is untyped, e.g. of type java.lang.Object.
2. Required properties: these properties are required by the framework. Each implementation must provide values for required properties.
3. Optional (but common) properties: these properties are not required by the framework. Not all SCM solutions support those properties. If the solution supports it, the framework provides the value. Otherwise, it tries to find an associated custom property value. If this fails as well, accessing the value raises an exception ('not implemented').
4. Custom properties: these properties are specific to individual SCM solutions. Also the user of the SCM solution might add user defined properties. An array provides these properties.

Associated custom property values store a value for the AbstractALM Attribute. They are named like the AbstractALM value and a namespace. For example: if company is not supported by the native SDK, but it allows custom properties, it creates a custom property *ALM:Company* or *ALM-Company* or *ALMCompany*. This custom property is used to map the AbstractALM attribute to.

Libraries implement these objects by providing interfaces (if available) or abstract classes. Implementations provide adapter classes that implement the interfaces or extend the abstract classes.

User

This object contains all available information about a user of the SCM solution. Users are not versioned.

Attributes

Attribute Name	Data Type	Attribute Type	Description
NativeObject	Object	Optional	The native object from the vendor SDK. This is an untyped object, e.g. of type *java.lang.Object*.
ID	String	Required	A system wide unique ID of the user. Could be the login name, email address, primary key of the database table, etc. The value becomes part of the user's URI.
Login	String	Required	The login name of the user.
Active	Boolean	Required	Indicates if the account is active (usable). If this value is false, the user is not able to log into the system using this account. If the field is not available in the adopting SCM system, this value might be just true, indicating that every available user is active.
Name	String	Optional	The full name (first + last name) of the user.
Company	String	Optional	The name of the company.
Email	String	Optional	The email address of the user.
PhoneNumber	String	Optional	The phone number of the user.
MobileNumber	String	Optional	The mobile phone number of the user.
FaxNumber	String	Optional	The fax number of the user.

PagerNumber	String	Optional	The pager number of the user.
WebSite	String	Optional	The URL of the user's web site. Includes the protocol such as *http://www.olschimke.eu*.
TimeZoneShift	Integer	Optional	The time zone shift of the user.
PostalAddress	String	Optional	The postal address of the user.
VoiceMailNumber	String	Optional	The voice mail number of the user.
IMNumber	String	Optional	The number of an instant messaging service, such as MSN or ICQ.
Language	String	Optional	The language of the user. The format follows TBW.
CreatedDate	DateTime	Optional	The date and time the user was created.
CustomProperties	Collection (Property)	Optional	The custom properties for the user.
Connection	Connection	Required	The connection (server) this user is stored at. Is defined in the library.
CreatedUser	User	Optional	The user (administrator) who has created the user.
ChangedDate	DateTime	Optional	The date and time the user was updated the last time.
ChangedUser	User	Optional	The user who has changed the user the last time.
Description	String	Optional	A description for the user.

Methods

TBW: Save

Data Definition

User = [NativeObject] + ID + URI + [NativeURI] + Login + Active + [Name] + [Company] + [Email]
+ [PhoneNumber] + [MobileNumber] + [FaxNumber] + [PagerNumber] + [WebSite]
+ [TimeZoneShift] + [PostalAddress] + [VoiceMailNumber] + [IMNumber] + [Language]
+ [CreatedDate] + 0[Property]*

Property

Represents a custom value. The library could define multiple properties, for various value data types (such as IntegerProperty with a value attribute of int, StringProperty with a value attribute of string, etc.). There should be properties for the following datatypes:

- Integer numbers
- Float numbers
- Strings
- Dates and Times
- Enumerations
- User
- Boolean
- Objects (as defined in AbstractALM: User, Repository, Package, Artifact, Folder, File)

The type of the property should be retrieved using language features, such as the instanceof statement.

Attributes

Attribute Name	Data Type	Attribute Type	Description
NativeObject	Object	Optional	The native object from the vendor SDK. This is an untyped object, e.g. of type *java.lang.Object*.
ID	String	Required	A system wide unique ID of the property. Could be the property name (if unique), path, primary key of the database table, etc. The value becomes part of the repository's URI.
Name	String	Required	The name of the property.
Value	Variant	Optional	The value of the property.
CreatedBy	User	Optional	The user who has created the value (not the property).
CreatedDate	DateTime	Optional	The date and time the value was created.
ChangedBy	User	Optional	The user who has changed the value the last time.
ChangedDate	DateTime	Optional	The date and time the value was changed the last time.
Description	String	Optional	The description of the property.

Data Definition

Property = [NativeObject] + ID + URI + [NativeURI] + Name + [Value]

Repository

This is the repository where the file revisions are stored. It is actually not the folder but a more abstract 'location' like a project, view, or repository path. The repository has one root folder. This root folder provides access to the file revisions. The repository provides access to meta-information, such as the description.

Attributes

Attribute Name	Data Type	Attribute Type	Description
NativeObject	Object	Optional	The native object from the vendor SDK. This is an untyped object, e.g. of type *java.lang.Object*.
ID	String	Required	A system wide unique ID of the repository. Could be the repository name (if unique), path, primary key of the database table, etc. The value becomes part of the repository's URI.
Name	String	Optional	The name of the repository. Could be the product or project name. Does not need to be unique.
Description	String	Optional	A description for the repository.
CustomProperties	Collection (Property)	Optional	The custom properties for the repository.
CreatedDate	DateTime	Optional	The date and time the repository was created.
CreatedUser	User	Optional	The user who has created the repository.
Packages	Collection (Package)	Required	All currently active and applied / past packages. The package does not contain any artifacts as long as not checked out.
RootFolder	Folder	Required	The root folder of the repository.
DerivedRepositories	Collection (Repository)	Optional	All derived (branches) of this repository.
Connection	Connection	Required	The connection this repository is using. Comes from the library.
Project	String	Optional	The name of the project this repository belongs to. The project can be retrieved using the native object only.
ChangedDate	DateTime	Optional	The date and time the repository was changed the last time.
ChangedUser	User	Optional	The user who has changed the repository the last time.

Methods

Method Name	Parameter	Param Type	Description
CheckOut	Result	Package	Checks out all artifacts of a package.
	Package	Package	The package to be checked out.
	Lock	Boolean	Indicates if the files should be locked on the system (if supported).
CheckIn			Checks in the artifacts of the package.
	Package	Package	The package to be checked in.
	Description	String	A comment to be used for the change log. If this

	value is not set or empty, the package comment is used.

Data Definition

Repository = [NativeObject] + ID + URI + [NativeURI] + [Name] + [Description] + 0[Property]* + CreatedDate + 0[Package]* + Folder

Package

A package is used to check-in files in an atomic transaction. Packages are also used to get information about transactions run in the past.

Attributes

Attribute Name	Data Type	Attribute Type	Description
NativeObject	Object	Optional	The native object from the vendor SDK. This is an untyped object, e.g. of type *java.lang.Object*.
ID	String	Required	A system wide unique ID of the package. Could be the package name (if unique), path, primary key of the database table, etc. The value becomes part of the repository's URI.
Name	String	Optional	The name of the package.
Description	String	Optional	A description for the package. Should be the check-in comment if appropriate.
Artifacts	Collection (Artifact)	Required	All artifacts of the package.
ParentRepository	Repository	Required	The repository this package belongs to.
CustomProperties	Collection (Property)	Optional	Custom properties for this package.
CreatedUser	User	Optional	The user who has initiated the package / transaction.
CreatedDate	DateTime	Optional	The date and time the package / transaction was initiated.
Repository	Repository	Required	The repository this package belongs to.

Methods

Method Name	Parameter	Param Type	Description
CheckOut			Checks out all artifacts of a package from the currently assigned repository.
	Lock	Boolean	Indicates if the files should be locked on the system (if supported).
CheckIn			Checks in the artifacts of the package into the currently assigned repository.
	Description	String	A comment to be used for the change log. If this value is not set or empty, the package comment is used.

Data Definition

Package = [NativeObject] + ID + URI + [NativeURI] + Name + Description + Artifacts + Repository + 0[Property]*

Artifact

Represents an artifact stored in the project.

Attributes

Attribute Name	Data Type	Attribute Type	Description
NativeObject	Object	Optional	The native object from the vendor SDK. This is an untyped object, e.g. of type *java.lang.Object*.
ID	String	Required	A system wide unique ID of the artifact. Could be the artifact name (if unique), path, primary key of the database table, etc. The value becomes part of the artifact's URI.
Name	String	Optional	The name of the artifact.
Description	String	Optional	A description for the artifact.
Comment	String	Optional	A comment for the revision.
Attachments	Collection (Attachment)	Optional	An optional collection of attachments (files).
CustomProperties	Collection (Property)	Optional	An optional collection of custom properties.
CreatedUser	User	Optional	The user who has created the first revision of this artifact.
CreatedDate	DateTime	Optional	The date and time the first revision of this artifact was created.
ChangedUser	User	Optional	The user who has changed this revision of the artifact.
ChangedDate	DateTime	Optional	The date and time this revision of the artifact was changed.
History	Collection (Artifact)	Optional	The previous artifact revisions.
RevisionNumber	Int	Optional	The revision number of the artifact revision.
VersionNumber	String	Optional	The version number of the artifact revision in dot notation.
ParentFolder	Folder	Required	The folder where the artifact revision is stored.
Repository	Repository	Required	The repository the artifact is stored in.

Data Definition

Artifact = [NativeObject] + ID + URI + [NativeURI] + Name + Description + 0[Attachment]* + 0[Property]* + CreatedUser + CreatedDate + ChangedUser + ChangedDate + 0[Artifact]* + RevisionNumber + ParentFolder + ParentRepository

Attachment

An attachment is a file that has been attached to an artifact.

Attributes

Attribute Name	Data Type	Attribute Type	Description
NativeObject	Object	Optional	The native object from the vendor SDK. This is an untyped object, e.g. of type *java.lang.Object*.
ID	String	Required	A system wide unique ID of the attachment. Could be the file name (if unique), path, primary key of the database table, etc. The value becomes part of the attachment's URI.
Name	String	Required	The file name of the attachment.
Description	String	Optional	A description for the attachment.
Content	Stream	Required	The file content of the attachment.
Repository	Repository	Required	The repository this attachment belongs to.
CreatedUser	User	Optional	The user who has created the attachment.
CreatedDate	DateTime	Optional	The date and time the attachment was created.
ChangedUser	User	Optional	The user who has changed the attachment the last time.
ChangedDate	DateTime	Optional	The date and time the attachment was changed the last time.

Data Definition

Attachment = [NativeObject] + ID + URI + [NativeURI] + Name + [Description] + Content

Folder

A folder is an artifact that can contain other artifacts.

Attributes

Attribute Name	Data Type	Attribute Type	Description
		Same as Artifact	
Artifacts	Collection (Artifact)	Optional	The sub artifacts.

Data Definition

Folder = Artifact + 0[Artifact]*

File

A file is an artifact that has content, such as source code, or binary data.

Attributes

Attribute Name	Data Type	Attribute Type	Description
		Same as Artifact	
Content	Stream	Required	The file contents.

Data Definition

File = Artifact + Content