

Heterogeneous Model Distribution and Integration using CDIF and the Web Infrastructure

Johannes Ernst

Founder and President, Aviatis Corp.

<http://www.aviatis.com>, <mailto:jernst@aviatis.com>

1 INTRODUCTION

In large engineering projects, models tend to be one of the most valuable piece of information. For example, in a project that might involve dozens or hundreds of engineers, how do engineers on different sub-teams discuss how the various components that they are building interact and fit together? Whether the means of communication is a white board or a modeling tool (such as a CASE tool), one of the primary ways of expressing facts about the system are “schematics”, abstractions of the system under construction that help engineers understand the participating components and modules, and their relationships and interactions.

Thus, one of the primary challenges for a large project team is to communicate the information contained in models effectively and efficiently throughout the team. While not all participants will be interested in all aspects of the system, or even the subset that can be conveyed in models, virtually all projects today err on the side of too little communication between participants, rather than too much. As a result, the project participants often do not effectively work as a team, have a different understanding of what the project’s end product will look like and how it will work, with consequences that may range from significant schedule overruns, over features that do not work as specified or intended, all the way to complete project failures.

Modeling tools in general have improved the situation somewhat: few whiteboard models survive the afternoon on which they were drawn, they are not maintainable, and generally only communicated and accessible to very few people (whoever was in that particular meeting), among many other disadvantages (e.g. lack of analyzability); modeling tools do not share these disadvantages. Of course there are advantages for white board models as well, such as that they typically come with a tutorial on the model by the person drawing it.

Although models created and held by modeling tools are generally better accessible and usable, typical modeling tools have never been particularly good at communicating model content to the potentially large number of engineers that was not part of the sub-team who authored the particular model in the first place. There is a variety of reasons for that, among them that models are inherently difficult to understand (just like code is inherently difficult to understand), but also unnec-

essary issues such as the lack of team support by many tools, and the requirement that even “read-only users” (such as engineers on the team that only want to understand someone’s model, not change it) have the tool installed locally, pay license fees and know how to use it.

The problem gets compounded in what we call “heterogeneous projects”. As long as every member on the project writes, say, part of the same application using C++, there is a reasonable chance to “standardize” on one modeling tool and mandate its use for everyone. With a shared repository, all the users can access all artifacts in that repository and communication about modeling issues can work reasonably well. But what if only some of the project’s engineers develop object-oriented software, but others are, let’s say, mechanical engineers (for example in a robotics application). The likelihood that the software engineers will convince the mechanical engineers to throw away their CAD tools to use the same CASE tool the software group is using, or vice versa, is (hopefully) zero. But nevertheless, the groups working on different technologies need to communicate to each other what each of them are doing, what their assumptions are and how everything relates to each other.

For us, this “heterogeneous, large, distributed project” problem is the quintessential model integration problem. In this contribution, we discuss the application of some internet and world-wide-web approaches to large-scale model integration. The CDIF architecture has been used successfully as the foundation on which the integration technologies are built, and integrates very well with internet concepts.

2 THE WORLD-WIDE-WEB AND ITS RELATION TO ENGINEERING MODELS

Enter the internet, and in particular, the world-wide-web. If the world has ever known an “information integration medium”, it’s the web. Why?

First of all, the architecture underlying the web is content-type agnostic. While HTML is the de-facto content type for container content (and there are others, such as PDF), the web does not depend on it, and anyone can add new data types on the web at will, something that happens every day. To make the web work, only three concepts are required:

- things of arbitrary types (such as files or other resources), the “content”
- names for these resources, and ways of accessing them using those names, known as URLs and URIs that encode a protocol. Optionally, in order to link various “content items”, names for other resources may be attached to or embedded in any resource (e.g. the ubiquitous `` tags in HTML)
- content handlers that can do something meaningful, such as displaying, the resources identified by the URLs¹.

Obviously, this extreme simplicity is one of the reasons why the web took off in the first place. And it is one of the reasons why its application on new domains, such as engineering, is possible and opens up far-reaching possibilities.

In order to use the web infrastructure for distributed integration of models, we thus only have to map models, potentially distributed across multiple tools, onto the above bullets:

- The “content items” / resources for models are obviously either the models themselves or subsets, depending on the granularity of the desired application.
- These resources need to be identifiable by URLs in the desired granularity.
- Content handlers need to exist that can display models, or subsets of models, given a URL.

All of these items are discussed in more detail below. If we have all three, models can “live on the web” just like HTML files do, and model integration is as easy as adding a hotlink to HTML files.

3 MODELS AS WEB RESOURCES, THEIR REPRESENTATION AND THE CDIF MODEL INTEROPERABILITY ARCHITECTURE

The biggest issue for this distributed model integration using the web-infrastructure is deciding on the granularity of model information. Modeling tools typically expose their models to the user with the granularity of

- a repository/database at time, that may include models from multiple projects
- a project, that may include a (large) number of models, or
- a model, that may include many different instances of the concepts provided by the modeling language that was employed by the user (such as UML [12], CDIF Data Flow Modeling[2], or whatever)
- More recently, tools sometimes allow access to individual instances of the modeling language’s concepts (e.g. a particular class “Customer” using the UML).

The necessary granularity of information obviously depends on the application, but most applications, including model integra-

tion, require fine-granular access to individual instances of individual concepts of the modeling language. In the above mentioned robotics project, for example, the software engineers might want to express that a particular class they have in their class-association model, called “RobotArm”, corresponds to a particular mechanical component called “RobotArm” that the mechanical engineers use in their CAD software. Obviously, any coarser granularity would not allow this fine-granular point-to-point link.

Thus, we assume that the granularity of all information is the granularity of individual instances of the basic concepts provided by the underlying modeling language. Now what are those? (Note that we have not assumed any particular modeling language for this discussion)

Modeling languages can be defined in a variety of ways, but the concept of meta-modeling is probably the most powerful and, after many years of niche applications, is finally gaining increasing acceptance in the community. Examples where the concept of meta-modeling has been used are CDIF [3], the UML [12], the Microsoft “Open Information Model” [11] and others. Various meta-modeling architectures have been proposed, and unfortunately, more of them has been standardized (one should have been enough!). All of them have various advantages and disadvantages that shall not be discussed here. For our work, we have used the CDIF Meta-modeling architecture as documented in [3], because it is both very simple (thus requiring little implementation effort with secondary consequences such as allowing higher-quality tool software), and fairly powerful (including concepts such as meta-model federations, non-conflicting meta-model extensions, advanced inheritance features, versioning etc.).

The CDIF meta-modeling architecture can represent arbitrary meta-models either directly or with a minimum amount of mapping. This has been shown by CDIF meta-model standards themselves, the representation of UML within the CDIF architecture [4], mappings to and from STEP [1] (which is a set of standards that includes meta-models for mechanical engineering) as well as numerous proprietary meta-models.

To represent any model element as a web content item, the only requirement is to give this model element the facade or representation of a CDIF meta-entity (a.k.a meta-class) or meta-relationship, using a standardized or proprietary meta-model.

It is worth noting that the approach discussed here does NOT require the adoption of a universally agreed meta-model, be it the CDIF Integrated Meta-model, the UML, or any other meta-model: it works as well with standardized meta-models as with proprietary extensions or fully proprietary meta-models. It is important to point out that this is different from virtually any other interoperability or integration approach for modeling, but that is essentially the web model: no standardization on the one and only file type was necessary for the web, and this ability to cope with new file formats is one of its greatest abilities.² Similarly, we do not require the agreement on one meta-model either.

¹ For the purposes of this article, we’ll use the term URL for both URLs and URIs.

4 URLs FOR MODEL ITEM NAMING AND LINKING

The purpose of a URL is to name a particular resource somewhere on the internet in a way so that it can be universally found and accessed. To accomplish this, a URL is a compound of location information (the host name, and via DNS, the IP address), a hierarchical name relative to the location (typically a path to a file or a subset of a file), and an access protocol (such as http, ftp etc. with optional parameters such as port number, login name etc.).

To identify a model item, we use the exact same scheme. The location information section identifies the name of the host where the item is located. The hierarchical relative resource name is resolved by the server process located there, and the resource is returned using the specified protocol.

In terms of protocol, there are a number of choices, and some of them have interesting applications:

- the use standard internet protocols such as http, ftp etc. has obviously advantages over new protocols. On the other hand, these protocols are one-way file-exchange protocols, and to use them, we need a corresponding file format that is, or can be, as pervasive for model information as HTML is for formatted text information. The best candidate for this at this time is of course the CDIF Transfer Format [5][6][7]
- use a “distributed object protocol” such as IIOP or Java RMI. In this case, no copying takes place as the client talks “live” to the server such as described by the CORBA bindings to CDIF [8].

In case of a file-transmitting protocol, it is of course possible to exchange more than one model item per interaction, and that can lead to far higher efficiency in case more than one item needs to be accessed at a time. To allow both, we propose to use the ‘#’ URL separator to distinguish URLs that identify files from URLs that identify concepts within files.

This leads to three possible URL syntaxes:

- (1) `http://somehost.com/one/two/myfile.cdif`
- (2) `http://somehost.com/one/two/myfile.cdif#someobject`
- (3) `iiop://somehost.com/one/two/someobject` or `rmi://somehost.com/one/two/someobject`

The first one identifies a collection of model objects, represented in the CDIF file format, on host `somehost.com`, to be accessed using the http protocol. It is the http server’s responsibility to translate the relative name `one/two/some-`

² Of course, it is not possible to exchange model information between various software products without an agreed-upon meta-model (which is the original goal of CDIF), but this not our goal here. Having said this, although an agreement on a standard meta-model is not necessary for our approach to be technically feasible, an agreed-upon meta-model has very significant economic advantages as it reduces the number of software components that need to be developed and maintained. Consequently, we actually use a significant part of the standardized CDIF Integrated Meta-model in our implementation.

`file.cdif` into a CDIF file. Of course, the server can use a script to dynamically create such a file, for example by dynamically querying a tool’s repository.

The second case is identical to the first, except that we explicitly identify an object with the identifier `someobject` in that CDIF file. The CDIF Transfer Format uses the concept of a `CDIFIdentifier` to identify individual instances in the transfer, and the component of the relative name after the ‘#’ sign corresponds to that. So, for example, if the file `somefile.cdif` contains the data

```
...
(Class someobject
  (Name "Customer")
  (IsAbstract -TRUE-))

(Class otherobject
  (Name "RepeatCustomer"))

(Class.Extends.Class ext otherobject someobject)
...
```

the above URL identifies only the Customer class out of the file, discarding the rest of the information.

In the third case, the URL acts as “long pointer” to an object supporting IDL interface `Class`, according to the CORBA bindings to CDIF, and the client content handler is free to apply any operations on it.

We also extended the URL syntax as specified by the W3C slightly in order to be able to deal with versioning (a subject that the W3C has not addressed). We picked the character ‘*’ as an additional separator to separate the model item identity from its version specification, so that

- (1) `http://somehost.com/one/two/somefile.cdif*LATEST`
`http://somehost.com/one/two/somefile.cdif`

both mean that we want to access `somefile.cdif` in the latest version and that

- (2) `http://somehost.com/one/two/somefile.cdif*1.2`

means `somefile.cdif` in version 1.2.

5 MODEL CONTENT HANDLERS

Content handlers on the web are invoked based on the data type of the content received. For example, a web-browser might receive a file, and because it knows that its type is HTML (based on a fairly complex set of rules that involves MIME types, default rules, and product specifics on both client and server), it will delegate the file to the browser’s HTML rendering component. Most browsers have capabilities to add new content handlers for new data types.

In case of URLs identifying CDIF files or portions thereof, a content handler type needs to be defined that can parse CDIF files, and then display them. A number of CDIF parsers has been implemented by various organizations, and at least one of them is publicly available, called Jack [9] (“Johannes Adaptable CDIF Kernel”). Such a parser can interpret the information contained in the CDIF file, and when used in conjunction with a renderer, can then graphically or textually render the

model information. An implementation is possible either using a separate helper application or a browser plug-in.

6 PUTTING IT ALL TOGETHER

Putting all of this together, we are getting the following capabilities:

- Any model item that any developer on the project creates, regardless of location, type or vendor of tool, or discipline (such as software construction or mechanics), has a corresponding URL that acts as its “global name”. The URL encodes the preferred protocol of model item access, such as http or iiop.
- Because it is just a regular URL, it can be pasted into e-mails, referenced from intranet pages, or in files documenting the project
- This URL functions just like any other URL the user is familiar with: type or paste it into the web browser, and the resource described by it will show up in the web browser. The latter is implemented using appropriate content handlers that can parse CDIF files.
- Tool vendors can (and do) add functionality to their tools that allows the association of arbitrary URLs with model elements. This very simple to do for the vendor, in particular as most desktop OSs now incorporate URL opening functionality in the operating system. For the user this means that clicking on a model item in a particular tool will open up the browser and show associated information that might come from a different tool and from somewhere on the network, without having to worry about installation, platform etc.
- Tool vendors also add functionality to their tools (note that this can often be performed by third parties) that allows the identification of individual model elements in the repository of the tool, so that the model information can be accessed by other clients.

As a result, it is possible to reference, without regard to location, networks, vendors or tools, any model item from anywhere else in the project, creating a “virtual model” of the system “in cyberspace”.

From a user’s perspective, this is very attractive: the user does not need to change tools, nor does he need to learn how to use other tools. From a simple web-browser (augmented by the appropriate plug-in etc.), he can traverse the overall system model as it exists, distributed across so many machines, locations, and tools.

From a tool vendor’s perspective, adding the capability of referencing and opening arbitrary URLs from the model elements is quite easy to implement. Letting other vendors access model elements held by the tool through URLs requires a little more effort, but is still relatively easy to implement, in particular if that vendor already provides a CDIF export interface.

7 AN EXAMPLE

In an control system project, such as found in robotics, aerospace, automotive and other industries, projects are often structured in a group responsible for “the system”, and one group each for mechanics, software and other technologies that might be used for the project. In this example, we assume that a control algorithm for a mechanical device needs to be constructed.

The “systems” people will likely use a tool such as a CASE tool to model the overall system structure. This structure is shown in Figure 1. Note that control systems terminology has been applied: “Controller” stands for the software that controls the operation of the mechanical subsystem, called the “Plant” (primarily for historical reasons).

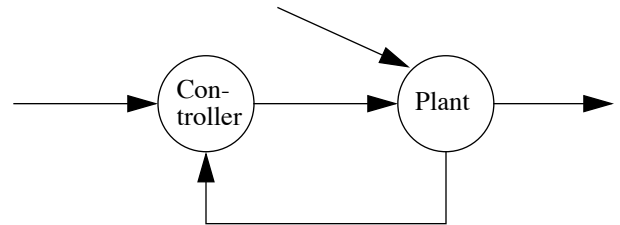


FIGURE 1. “System” model

From the perspective of the “systems” people, the details how either the controller or the plant are implemented are irrelevant, and the responsibility of the respective teams, so the “system” team stops here.

The team responsible for the controller might come up with a model for the control algorithm in a computer-aided control systems design (CACSD) tool. Such as model is shown in Figure 2. Note that the modeling technique is different: in the “system” case a variation of data flow modeling has been chosen by the “system” team (they could have chosen an object modeling technique, or anything else, it does not really matter), while the CACSD people chose a variation of CACSD modeling. (CACSD modeling techniques have more or less been used unchanged for about a hundred years, and it is not important what exactly the model means below) The important point is that they chose a different technique (which they could, as there was little incentive to use the same), but the models need to be integrated.

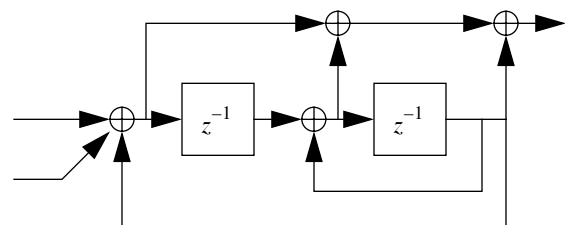


FIGURE 2. Controller sub-model

Similarly, the mechanics team creates a model using a CAD tool for the mechanical component of the overall system. This model is not shown.

We also assume that the various groups are distributed in different departments, or even organizations.

Using the technology discussed above, every model element in any of the models is accessible through a URL. For example, the Process “Plant” could be accessed by anyone on the project right from their browser, simply by typing in the URL off that process. More importantly, though, the “system” team can access the CACSD sub-model and the mechanics sub-model through a URL.

Assuming that the “system” team uses a modeling tool that can associate URLs with other modeling items, the team can associate the URL for the CACSD model with the process called “Controller” and the URL for the mechanics model with the URL for “Plant”. As a result, when a user browses the “system model”, he can traverse to the correct CACSD model as easily as by clicking on the process “Controller”.

Note that the underlying implementation might involve CGI scripts that create CDIF files on the fly out of the repository of the CACSD tool, that then are parsed by the client browser’s plug-in etc., but all this complexity is hidden from the user. The user only sees URLs, and hot-links, and an integrated model. After all, that’s what the user is interested in when he wants to find out how the system works. Anything more complex is unwarranted.

8 SUMMARY

Models are one of the core assets facilitating communication of large, distributed engineering teams that work on complex, heterogeneous projects. In such projects, it is neither possible nor desirable to standardize on one particular tool. Also, the “overhead” required for model integration needs to be very low, so that individual engineers voluntarily use the integrated information.

The idea of using URLs to identify distributed model items was discussed. It was shown that this idea is a direct extension of the CDIFIdentifier concept that has been standardized in [6]. By defining an appropriate URL syntax, arbitrary model items that are represented as instances of the CDIF meta-meta-model, can be accessed across the internet through simple browsers, regardless of the type of model item and its location.

By adding capabilities to existing tools that allow the association of URLs with modeling elements, and through the use of CDIF Exporters that can be invoked by web servers, model integration is possible “in cyberspace”, without requiring significant changes to the state of the art.

The technology discussed in this article is part of the Aviatix Internet Engineering Platform™ currently in development by Aviatix Corp.

9 REFERENCES

- [1] Davis, Hugh: A mapping between CDIF and STEP. British Standards Institute, 1998.
- [2] Electronic Industries Association / CDIF Technical Committee: CDIF - Integrated Meta-model - Data Flow Modeling Subject Area. EIA Interim Standard, 1995.
- [3] Electronic Industries Association / CDIF Technical Committee: CDIF - Framework for Modeling and Extensibility. EIA Interim Standard, 1994.
- [4] Ernst, Johannes: Using the CDIF Transfer Format with the UML. www.cdif.org, 1997.
- [5] Electronic Industries Association / CDIF Technical Committee: CDIF - Transfer Format - General Rules for Syntaxes and Encodings. EIA Interim Standard, 1994.
- [6] Electronic Industries Association / CDIF Technical Committee: CDIF - Transfer Format - Syntax SYNTAX.1. EIA Interim Standard, 1994.
- [7] Electronic Industries Association / CDIF Technical Committee: CDIF - Transfer Format - Encoding ENCODING.1. EIA Interim Standard, 1994.
- [8] Electronic Industries Association / CDIF Technical Committee: CDIF - OMG IDL Bindings. EIA Interim Standard, 1996.
- [9] Ernst, Johannes: Jack. Johannes’ Adaptable CDIF Kernel. www.metamodel.com, 1995.
- [10] Ernst, Johannes: Intro to CDIF. www.cdif.org, updated since 1995.
- [11] Microsoft Corp.: The Open Information Model. Microsoft Corp., 1997.
- [12] Object Management Group: Unified Modeling Language V1.0. <http://www.omg.org>. September 1997.