CWM: A Model-based Architecture For Data Warehouse Interchange

Doug Tolbert doug.tolbert@unisys.com
Unisys Corporation (949)380-6606

The diversity of operational data sources and target data warehouse engines has made the construction and maintenance of data warehouses challenging. Source and target data engines may differ not only semantically (i.e., their core data models) but also infrastructurally (i.e., the operational details of how data is extracted and imported). The absence of common, sharable descriptions for the structure of both data sources and target data warehouse engines has meant that data warehousing tool vendors must address the interchange of data in a pairwise fashion (Figure 1). The combinatorics of the situation aggravate the already substantial semantic and infrastructure problems faced by tools that must cross data engine boundaries and have limited deployment and use of data warehouses in organizations with diverse assortments of source and target data engines. Even where data warehouses have been successfully deployed, keeping them synchronized with operational data sources remains a labor-intensive, pairwise process. Furthermore, the flipside of this situation — i.e., support for drilldown discovery of the operational origin of warehouse data — remains a pairwise metadata analysis problem.

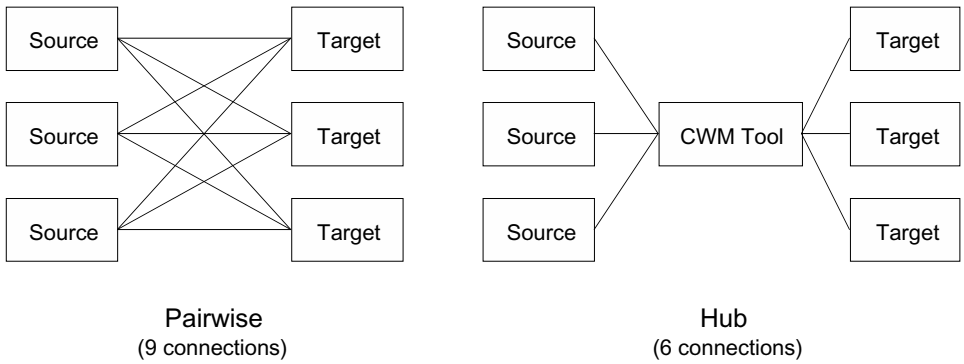


Figure 1. Data warehouse source and target arrangements.

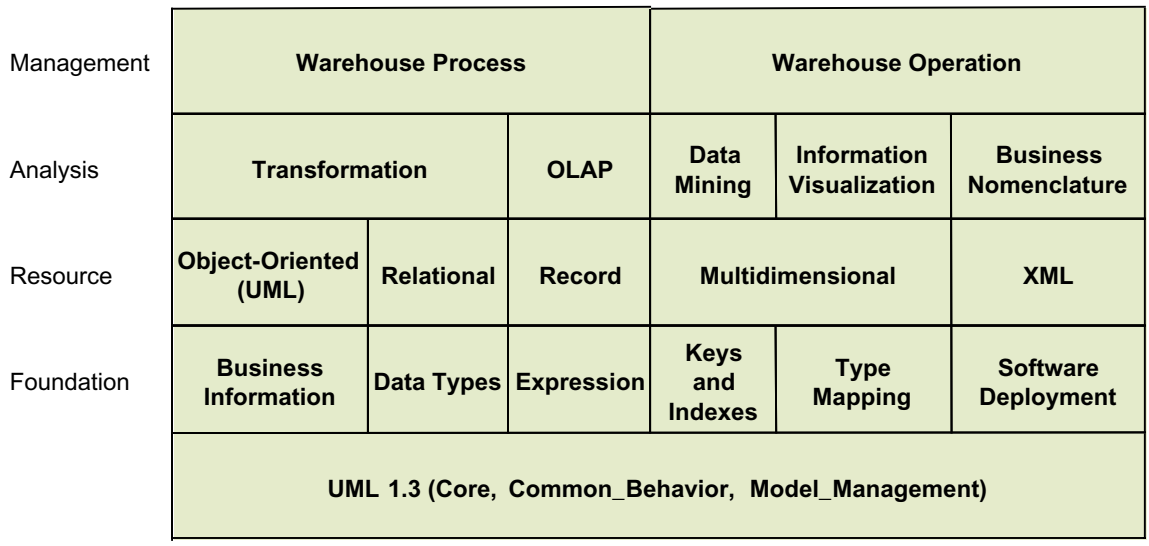
Fortunately, recent support for model-based software architectures in multi-vendor industry organizations offers hope that the pairwise combinatorics of these data warehousing tool scenarios can be reduced to something closer to the hub configuration in Figure 1 — a substantial savings even when the number of operational data sources and warehouse targets is as low as 3! In such configurations, data warehouse deployment and maintenance tools interface with a shared store containing metadata about the structure of various operational data sources and target data warehouses as well as descriptions of the transformations required to move data between them. Suitably designed tools can then use the metadata and transformations to orchestrate the extraction of data from operational sources and its transformation into forms appropriate for import into target data warehouse stores. In addition, the metadata and transformations in the common model store can be traversed in the opposite direction (i.e., from target to source) to support drilldown discovery of the origins of warehouse data. Besides reducing the number of semantic connections with which data warehousing tools must contend, such model-based architectures modernize data warehouse applications by making them more compliant with a component/connector architecture in which the tools are the components and the shared model store serves as the connector.

Doug Tolbert doug.tolbert@unisys.com
Unisys Corporation (949)380-6606

The Common Warehouse Metamodel (CWM) specification recently adopted by the Object Management Group (OMG) is an important milestone on the way to fully model-based architectures supporting data warehouse interchange. The CWM is based on OMG s Meta Object Facility (MOF) specification and employs OMG s XML Metadata Interchange (XMI) specification to interchange CWM-resident metadata between MOF-compliant repositories. The CWM metamodel is described in OMG s Unified Modeling Language (UML) and can be thought of as an extension specializing UML for data warehousing applications. To emphasize the importance of multi-vendor interchange, the CWM was developed as a single submission by eight co-submitting vendor and user-community companies (including IBM, NCR, Oracle, Hyperion, UBS, and Unisys) and supported by seven other companies (including HP, Sun, Hitachi, and John Deere). The CWM specification is available from the OMG s web site at http://www.omg.org.

The CWM metamodel is organized into 18 packages arranged in four layers on a UML base (Figure 2). CWM breaks new architectural ground by defining its sub-metamodel as individual packages. Because CWM uses modeling techniques that minimize the number of dependencies between its packages, tool integrators can select only those metamodel services they need while avoiding problems common to large, monolithic metamodels. The CWM co-submitters believe that similar modeling architectures can be leveraged to reduce the complexity of other monolithic models (such as UML itself).

Figure 2. CWM layers and package structure.



The four layers of the CWM collect together different sorts of metamodel packages:

- The Foundation layer contains general services that are shared by other packages.
- The Resource layer contains data models used for operational data sources and target data warehouses.

Doug Tolbert doug.tolbert@unisys.com
Unisys Corporation (949)380-6606

- The Analysis layer provides metamodels supporting logical services that may be mapped onto data stores defined by Resource layer packages. For example, the Transformation metamodel supports the definition of transformations between data warehouse sources and targets, and the OLAP metamodel allows data warehouses stored in either relational or multidimensional data engines to be viewed as dimensions and cubes.
- The Management layer metamodels support the operation of data warehouses by allowing the definition and scheduling of operational tasks (Warehouse Process package) and by recording the activity of warehouse processes and related statistics (Warehouse Operation package).

UML is the modeling foundation on which CWM is built. Wherever possible, the CWM co-submitters have directly reused existing UML classes and associations rather than creating CWM-specific versions of them. This choice both reduces the number of new CWM classes and associations and leverages the existing skills of UML-knowledgeable modelers. For example, the Object-Oriented package in the Resource layer is really just a reuse of existing UML classes that are already sufficient for describing object-oriented data sources (such as object-oriented DBMSs or application systems built in object-oriented languages like Java).

Although pains were taken to make the CWM sufficient to support many data warehouse interchange scenarios, the co-submitters realize to no general metamodel will provide all of the support necessary for the diversity of scenarios that will be encountered in active information processing installations. To accommodate this fact, the CWM co-submitters have planned for extensions in two ways. First, standard UML extension mechanisms such as inheritance and tagged values can be used to adapt the normative CWM metamodels to particular uses; of course, the extent to which such extensions can be interchanged depends on the ability of tools at the endpoints of the interchange to understand the extensions they receive. Even though they may be extended, CWM metadata objects can still be interchanged with full fidelity at the level of normative CWM classes. Second, a set of non-normative CWM extension models for widely known services have been provided, including Entity-Relationship, COBOL Data Division, Unisys DMS II network DBMS, IBM s IMS hierarchical DBMS, Hyperion s Essbase MOLAP tool, Oracle s Express ROLAP tool, and support for questionnaires, surveys, and reporting applications. The extension models are provided both as heuristic examples of proper CWM extension techniques and as starting points for further extensions and tool integrations.

The formal definition of the CWM metamodel is contained in a set of MOF-DTD based XML streams, one for each of the CWM packages, and corresponding CORBA IDL and XMI DTD files generated directly from the CWM specification itself. These files are available on OMG s web site at http://www.omg.org.

