COMPONENT DATABASE FOR THE APS UPGRADE*

S. Veseli, N.D. Arnold, D.P. Jarosz, J. Carwardine, G. Decker, N. Schwarz, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract

The Advanced Photon Source Upgrade (APS-U) project will replace the existing APS storage ring with a multi-bend achromat (MBA) lattice to provide extreme transverse coherence and extreme brightness x-rays to its users. As the time to replace the existing storage ring accelerator is of critical concern, an aggressive one-year removal/installation/testing period is being planned. To aid in the management of the thousands of components to be installed in such a short time, the Component Database (CDB) application is being developed with the purpose to identify, document, track, locate, and organize components in a central database. Three major domains are being addressed: Component definitions (which together make up an exhaustive "Component Catalog"), Designs (groupings of components to create subsystems), and Component Instances (“Inventory”). Relationships between the major domains offer additional "system knowledge" to be captured that will be leveraged with future tools and applications. It is imperative to provide sub-system engineers with a functional application early in the machine design cycle. Topics discussed in this paper include the initial design and deployment of CDB, as well as future development plans.

OVERVIEW

The Component Database (CDB) application is a tool for organizing and tracking components and designs used for the APS storage ring upgrade. It helps capture component documentation, provides a repository for inspection and measurement data (e.g., travellers), and supports logging of component history through the component’s life cycle.

CDB also serves as a user portal for finding all known information about a particular component or a design. To that end, it provides links and interfaces to external systems commonly used at APS, such as various drawing and document management systems, procurement applications, etc.

Although CDB has been designed and developed from the ground up in order to satisfy APS-U requirements, in many respects it draws ideas from IRMIS2 [1], which is still in use by Controls Group at APS.

SOFTWARE COMPONENTS

CDB is built around relational database, web portal and REST web service [2] technologies. The architecture, shown in Fig. 1, provides users with a number of options for accessing the system. At the same time, it also offers developers a fair amount of flexibility for integration with external applications. The most important CDB system components are described below in more details.

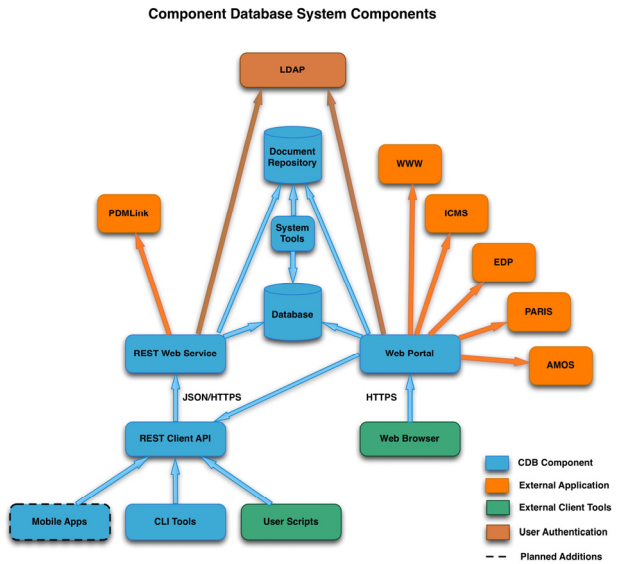

Figure 1: CDB system architecture. Dashed lines indicate future components.

Relational Database

The database contains all system data, other than uploaded documents. Users cannot access the database directly. Except for a small number of administrative tools, other software components access the database through object-relational mapper (ORM) libraries and APIs, which provide an abstraction layer from the rest of the system. CDB currently uses a MySQL database [3].

Document Repository

The Document Repository is a storage area designated for use and managed by the CDB software. It stores various documents, images, and log attachments uploaded by CDB users.

Web Portal

The User Web Portal is a Java EE application running in a GlassFish application server [4], and built using modern technologies like Java Persistence API (JPA) and Java Server Faces (JSF) [5]. In particular, CDB uses the PrimeFaces component suite [6].

Web Service

A REST Web Service and its Client APIs provide programmatic interfaces for accessing the system that are based on JSON data-interchange format [7] over the

* Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under contract DE-AC02-06CH11357.

ISBN 978-3-95450-148-9

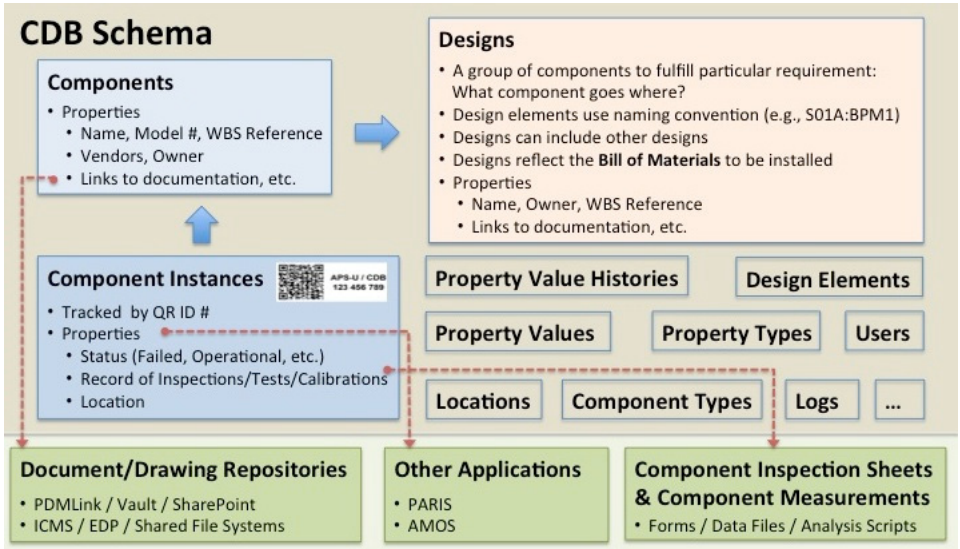

Figure 2: Major CDB domains and their relationship with external sources of information (depicted as rectangles with green background).

HTTPS protocol. The service is implemented in Python using CherryPy web framework [8] and SQLAlchemy ORM [9].

CDB users and administrators can write scripts and higher-level applications on top of the provided Python and Java APIs. It is worth pointing out that the CDB Web Service supports user sessions. This allows users to add new or modify existing CDB objects. Also, note that some of the Web Portal functionality, such as integration with a drawing management system used at APS, also requires accessing the Web Service.

Command Line Interfaces

Command Line Interfaces (CLIs) are built on top of REST Client APIs and expose Web Service interfaces for use within a UNIX shell. All CLI tools have a common set of options including those for command usage, debug level, output display format, etc. In addition, all commands have uniform session and error handling, which simplifies shell scripting.

SYSTEM FUNCTIONALITY

CDB software captures information about component definitions (to form a "Component Catalog"), component instances ("Inventory") and designs, which represent groupings of components used to create subsystems. These three major domains are illustrated in Fig. 2.

Component Catalog

A core CDB purpose is to provide a "Component Catalog" for the MBA. This catalog contains all components planned for use on the new machine, both custom-fabricated and commercially available. For example, each design of a gate valve, magnet, or a vacuum chamber and each unique VME module will have an entry in the Component Catalog.

The minimum metadata required for a component definition is a component name, type (representing generic components used on an accelerator), owner and owner group. Optional metadata may include a description, sources (i.e., vendors), and various other properties, such as images, links to documentation and external systems, etc. The system also supports "complex components" or "assemblies" via a special property that links a component to its associated design.

Inventory

Component entries in the Component Catalog describe a specific component design or a particular model number of a commercially available component. The actual hardware device fabricated or procured is referred to as a "Component Instance" of that component.

Component instances require inspection, testing, storage, installation, and maintenance. Their tracking becomes an inventory management challenge. Each component instance for the MBA will be uniquely identified with a QR code. If possible, a sticker with the QR code will be adhered to the device in a visible location. A database entry will relate the component instance to a particular component definition; thereby allowing all relevant information about this component instance to be referenced using the QR code.

Figure 3 shows an example of the component instance details view in CDB Web Portal. Both the component and component instance properties are displayed, and additional information is easily accessed from links in the property tables.

ISBN 978-3-95450-148-9

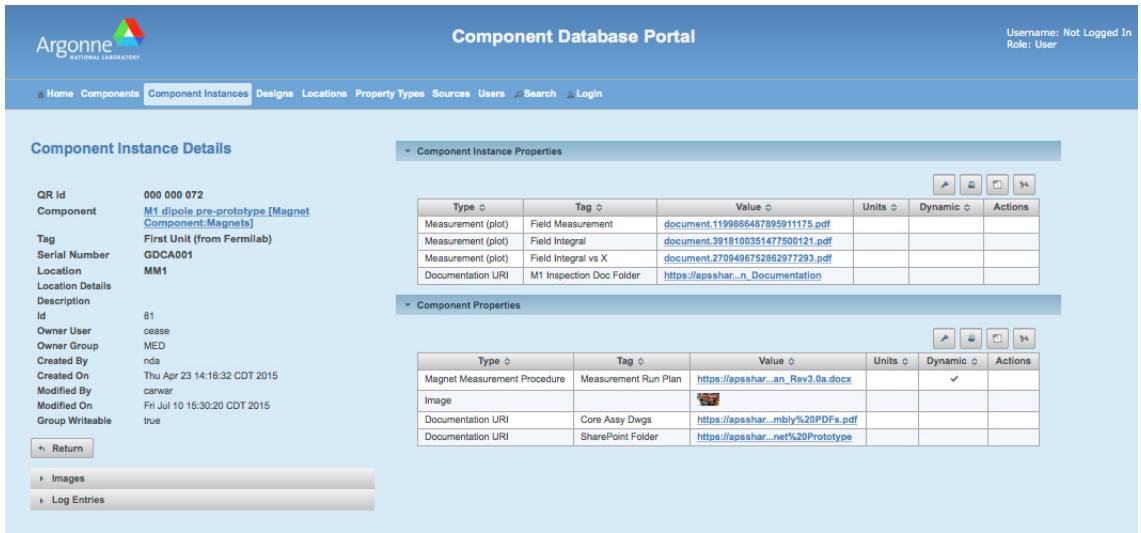

Figure 3: Component instance view in CDB Web Portal.

Designs

The CDB allows a user to define "Designs", which consist of several components grouped together to fulfill a particular functional requirement. One example would be a design for a BPM Processing System consisting of an analog front end chassis, an ADC chassis, four cables for the BPM buttons, and two cables between the other units.

Key concepts of Designs are described below:

- Designs are made up of "design elements".
- Each design element may be a component, a complex component (assembly), or another design (allows hierarchical designs).
- Each "design element" is given a unique element name, normally derived from the official naming convention.

"Designs" are the mechanism by which an exhaustive Bill Of Materials (BOM) can be acquired for the MBA. Groups will define designs necessary to fulfill their particular technical system requirements and by so doing will be contributing to a detailed list of the all the components required to build the new machine. Since this data resides in a relational database it can be "viewed" or analyzed in numerous ways.

Properties

The information one would like to capture in the CDB varies widely depending on the *type* of component or design being defined. Hence, it is impractical to attempt to define a single set of metadata that would be common to all objects. To provide a flexible mechanism for capturing object-dependent information, the CDB associates *properties* with individual components, designs, and instances. This allows each object to have its own unique set of metadata.

For example, a VME Chassis component might have properties of number of slots, height, and AC power requirements. In contrast, a quadrupole magnet component would have properties of maximum current, slot length, weight and maximum field.

Some of the most important features of CDB properties are as follows:

- Property types may be associated with a restrictive set of "allowed values".
- Property types may be linked to a unique "handler" class, which enables different view or edit modes in the Web Portal, or integration with an external system.
- A time-stamped history of each property value is kept to provide a historical log of each property.

Authentication and Authorization

The CDB allows any user to view and retrieve information without logging into the system. However, any changes to the system, such as adding new or editing existing entities, requires users to be registered, authenticated, and (in case of modifying existing objects) authorized to make changes. The authorization model is based on an object's user and group ownership, and the user's group membership. This model has been implemented for both Web Portal and Web Service. This allows, for example, adding new components or loading lattice designs using scripts reading spreadsheets.

DEVELOPMENT PROCESS

The CDB development process is based on lessons learned from developing similar systems in the past. Keys for a successful tool include management support, flexible software design, and an agile development process driven by user requirements and the need to provide solutions for real problems. Hence, our goal is to get new features into users' hands as quickly as possible, and our planning for future software releases heavily relies on user feedback from previous versions.

ISBN 978-3-95450-148-9

FUTURE PLANS

Near term plans include the addition of design instances, adding relationships between design elements (e.g. powered-by, controlled-by, etc.), and ability to capture cable connections.

CONCLUSION

The Component Database (CDB) application has the potential to capture a complete Bill of Materials for the new APS-U accelerator well before the installation timeframe. Having an exhaustive BOM in a relational database will facilitate careful planning and tracking of the construction and installation process, a prerequisite for an ambitious schedule anticipated by the APS-U project.

REFERENCES

[1] D.A. Dohan and N.D. Arnold, "Integrated Relational Modeling of Software, Hardware, and Cable Databases at the APS", p. 365, Proceedings of ICALEPCS 2003, Gyeongju, Korea (2003).
[2] R.T. Fielding and R.N. Taylor, "Principled design of the modern Web architecture", p. 407, Proceedings of ICSE 00, Limerick, Ireland (2000).
[3] MySQL website: https://www.mysql.com
[4] GlassFish website: https://glassfish.java.net
[5] For overview of Java EE features see http://www.oracle.com/technetwork/java/javaee/overview/index.html
[6] PrimeFaces website: http://primefaces.org
[7] JSON website: http://json.org
[8] CherryPy website: http://cherrypy.org
[9] SQLAlchemy website: http://sqlalchemy.org

ISBN 978-3-95450-148-9