

Metracker Version 1.5

Life-Cycle Performance Metrics Tracking

Robert J. Hitchcock
Lawrence Berkeley National Laboratory
Building Technologies Department
Environmental Energy Technologies Division

LEGAL NOTICE

Disclaimer

This manual was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Copyright

© 2000-2002 Regents of the University of California
Lawrence Berkeley National Laboratory

Table of Contents

Introduction.....	1
BLISS Information Model	1
Using Metracker.....	3
Project Versions.....	4
Version Data.....	5
Performance Objectives.....	7
Performance Metrics.....	8
Performance Metric Data Types	12
Metric Data Visualization.....	13
Actors.....	15
Installing Metracker	15
Acknowledgements.....	16
References.....	16

Introduction

Buildings often do not perform as well in practice as expected during pre-design planning, nor as intended at the design stage, nor even as measured during commissioning and maintenance operations. While this statement is generally considered to be true, it is difficult to quantify the impacts and long-term economic implications of a building in which performance does not meet expectations. This leads to a building process that is devoid of quantitative feedback that could be used to detect and correct problems both in an individual building and in the building process itself.

A key element in this situation is the lack of a standardized method for documenting and communicating information about the intended and actual performance of a building. This deficiency leads to several shortcomings in the life-cycle management of building information. Planners have no means of clearly specifying their expectations. Designers do not concisely document their design intent. Commissioning personnel have no standardized method for documenting the results of performance testing. Post-occupancy building performance cannot readily be compared to expectations in an attempt to evaluate and improve design and operation decisions. Lastly, without quantification of the magnitude of performance problems it is difficult to motivate building process participants to alter their current practice.

This document describes an information management concept and a prototype tool based on this concept that has been developed to address this situation. The Building Life-cycle Information System (BLISS) has been designed to manage a wide range of building related information across the life cycle of a building project. Metracker is a prototype implementation of BLISS based on the International Alliance for Interoperability's (IAI) Industry Foundation Classes (IFC). The IFC is an evolving data model under development by a variety of architectural, engineering, and construction (AEC) industry firms and organizations (IAI, 2001). Metracker has been developed to demonstrate and explore the process of tracking performance metrics across the building life cycle.

BLISS Information Model

The overall concept behind a building life-cycle information system is to provide a distributed computing environment for managing, archiving, and providing access to the wide variety of data that are generated across the complete life cycle of a building project. One goal of providing such a system is to initiate the industry-wide development and standardization of an interoperable set of tools to address a variety of information transfer problems in the building life cycle. Each individual tool must be tailored to respond to the needs of project participants within a specific phase of the project life cycle. Yet the data used and produced by each of these tools must be standardized to allow information sharing between project participants throughout the project life cycle.

Figure 1 shows a high-level representation of the key elements of the Building Life-cycle Information System (BLISS). One element of such an information system is a detailed data model describing the physical components and systems contained within a building design, such as walls, windows, spaces, and HVAC and lighting equipment. This type of data model is commonly referred to as a product model. In addition to a product model, BLISS contains a clear and concise representation of the performance objectives for a

building project based on quantitative performance metrics. BLISS also provides a means of representing the rationale behind design decisions by documenting the relationship between performance objectives and the product components selected to achieve these objectives. Within BLISS, this combination of informational elements is maintained for various versions as the project moves through its life cycle. This provides an historical record of previous versions along with the current version.

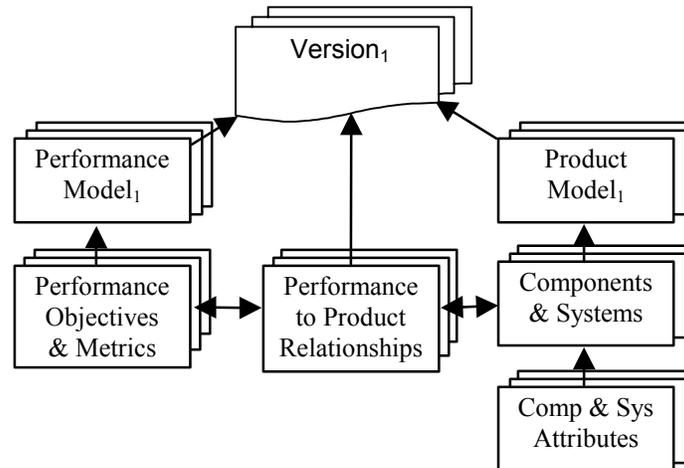


Figure 1: Key elements of a building life-cycle information system.

Figure 2 outlines a scenario for the use of BLISS focused on tracking performance metrics. The scenario begins in the programming phase (Step 1), where a set of key performance metrics is selected and recorded in BLISS to represent desired building performance objectives. In Step 2, computer aided design tools are used for the architectural and engineering design of the building, and the resulting data are used as input to various simulation tools to predict the performance of the current design for evaluation against the desired performance. Results from the final design simulations are summarized in an updated set of performance metrics, which establish a set of benchmarks for use in commissioning. Along with the design stage product model, these metrics more clearly document design intent. Modifications to the building design due to construction changes, or to the building operation due to occupancy or use changes, must be consistently documented in BLISS to provide as-built information. Note that the impacts of these changes can be evaluated more easily and comprehensively given the performance data contained in BLISS. As installation of each building system is completed, commissioning tests are conducted to determine if the design intent was met (Step 3). Also at this stage, in-situ test results are used to re-calibrate simulation models and to update the appropriate performance metrics. In this manner consistent up-to-date documentation of both the building and its expected performance is maintained for use during building occupancy.

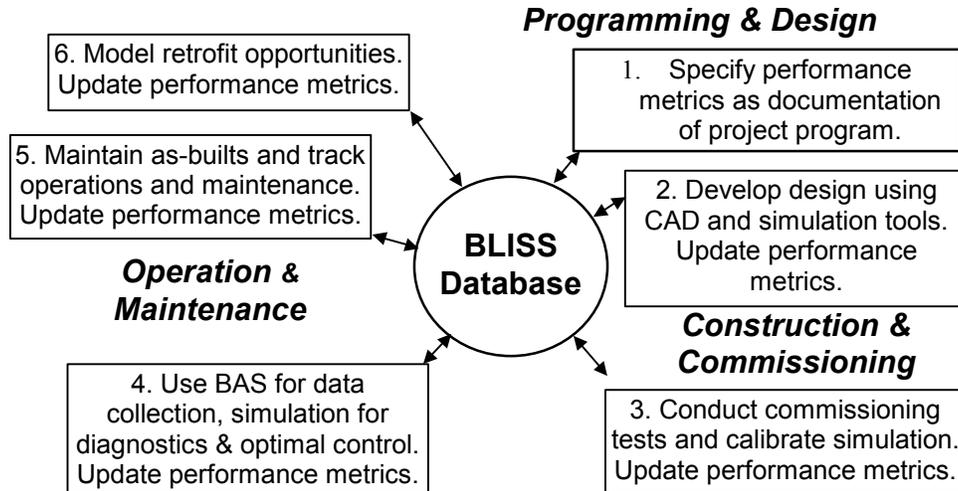


Figure 2: BLISS scenario focused on tracking performance metrics.

In Step 4 the building automation system (BAS) is used to continuously monitor the building and provide diagnostics using real-time simulation that checks actual operation against current performance benchmarks. These data are also used in Step 5 to track operations and maintenance (O&M) actions. One benefit of such an integrated information system is that one can readily identify the energy impact of O&M actions. For example, when the chillers are cleaned, their efficiency is improved and their new energy performance can be compared to previous performance. O&M can therefore be optimized. The system can also be linked to a retrofit simulation tool that would allow the facility manager to explore the energy savings from possible major or minor system changes (Step 6). Each step involves the generation of metrics, which are archived and accessed in the BLISS database format. The facility manager has a clear record of the design, as-built, and as-operated equipment, along with the performance of the building. Furthermore, the history of building design decisions and the resulting performance can be used to undertake post-occupancy evaluation of the building in order to better inform future designs.

Using Metracker

This section describes Metracker Version 1.5, a prototype tool based on the BLISS concept discussed in the previous scenario. Metracker focuses attention on the specification, tracking, and visualization of performance metrics. This is accomplished through a user interface that displays the organization and details of a BLISS archive, and graphical visualization of performance metric data comparing intended and actual performance across the building life cycle.

Metracker capabilities are intended for use by building owners and developers, design team members, project managers, and facility operations and maintenance personnel. Metracker is also intended as a prototype implementation that can be used to demonstrate its capabilities to these project participants, and to developers of related software tools. For more details on the purpose and structure of Metracker, see the companion Software Specifications Document.

The Metracker prototype is built upon the IAI Industry Foundation Classes (IFC) data model, extended with the BLISS definition of performance metrics. The performance metric extensions to the IFC data model conform to IAI prescribed extension methods, enabling other IFC-compliant tools to exchange data with the prototype. The data exchange mechanism uses the IFC method of writing and reading STEP Part 21 files, an international standard for the exchange of product data.

Metracker Version 1.5 is compliant with Releases 1.5.1 and 2.0 of the IFC model. IFC Release 2.0 includes class specifications for performance metrics, however these definitions have not been used in the Metracker 1.5 implementation for two reasons. The BSPro COM-Server IFC toolbox that is used in Metracker development does not yet support the R2.0 specifications for performance metrics. Also, a more flexible version of the R2.0 specifications have been implemented in Metracker 1.5 to allow further exploration of their structure.

Metracker has been developed using Microsoft Visual C++[®] 6.0, the BSPro COM-Server IFC toolbox (Olof Granlund, 2001), and the Olectra Chart[®] 6.0 graphics package (ComponentOne, 2001). The prototype runs under the Windows 9x/NT/2000 operating systems. The current capabilities of Metracker are described here within the context of using the prototype tool. The illustrative examples used below are based on a sample Metracker archive.

Project Versions

Metracker uses the Windows Multiple Document Interface metaphor similar to tools such as Microsoft Word[®]. Each document, or archive, in Metracker is referred to as a Project, and contains one or more Versions (i.e., snapshots of the Project as it moves through time), as illustrated in Figure 1. You can create a new Project, or open an existing Project, using the standard File/New or File/Open menu options. Upon creation each new Project is given a user-defined name for identification.

Figure 3 shows an opened Project named “Example Project” containing three Versions that have been archived in the Project. A list of the Versions archived in the opened Project appears in a tree view window as shown. The first of these Versions, named “Version One - Performance Planning,” is highlighted. Usually, the first Version archived in a Project will contain a set of Performance Objectives and Performance Metrics (see below) that have been established during project planning to represent the desired performance for a new building project.

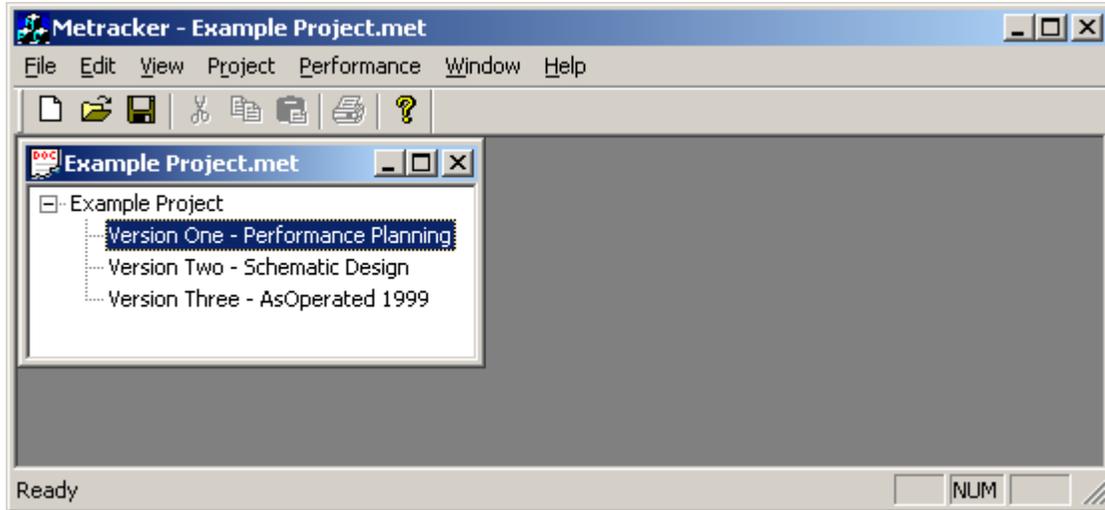


Figure 3: Versions within a Project document.

The other Versions in this Project are named “Version Two – Schematic Design” and “Version Three – AsOperated 1999.” Each of these Versions archives the details of the project as a snapshot in time. It is entirely up to the user as to when each of these snapshots is archived, but it will generally be at common project milestones.

In general, each subsequent Version will initially be created from the previous Version and modified to document changes that have occurred since the previous Version was archived. You create a new Version by choosing the Project/New Version menu option, and entering a name to identify the version. After a new Version is created its name appears in the Project tree view window.

Version Data

As illustrated in Figure 1, each Version within a Project potentially contains a wide variety of data including a product model, and a performance model composed of a set of performance objectives and metrics. Within Metracker, the data for the highlighted Version can be viewed and edited by choosing the View/Version Data menu option. Figure 4 shows the data for the Version named Version Three – AsOperated 1999, which is highlighted in the Project tree view window.

The data for each Version are actually archived within a separate IFC data file to facilitate interoperable sharing and editing with other IFC-compliant software tools. When you choose the View/Version Data menu option, the IFC data file associated with the highlighted Version is imported, and the data within this file are displayed in a separate window as illustrated in Figure 4. When the View/Version Data menu option is first selected for a newly created Version, you must associate an existing IFC data file with the new version. The data will then appear in the Version data view window.

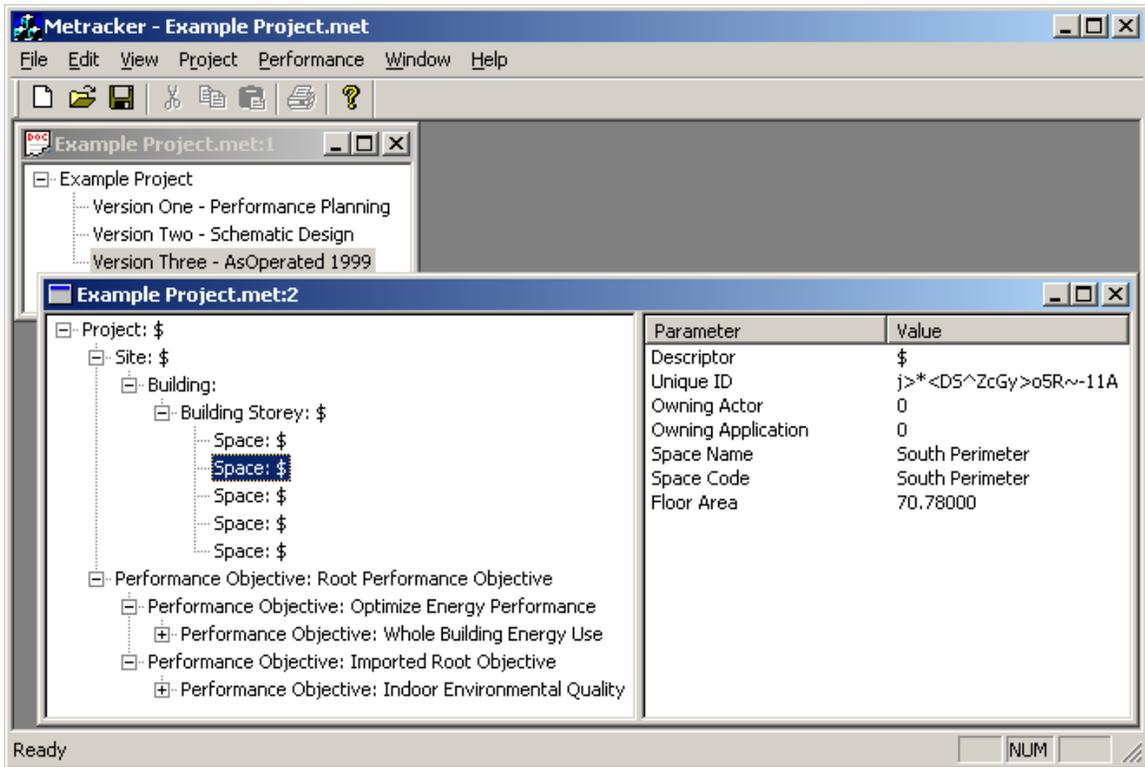


Figure 4: Version data view window.

The Version data view window is divided into two panes. The pane on the left displays a hierarchical tree view of the components (objects) within the IFC data file associated with the Version. Each node within the hierarchy is displayed with its component type (e.g., Space), and the identification name associated with the particular node (unfortunately “\$” in this example, indicating that a name has not been entered in the IFC file). The pane on the right displays a list view of the parameters (attributes) for the component node that is currently highlighted in the tree view, which in Figure 4 is a Space object. The parameter list view includes both the name of a parameter (e.g., Space Name) and the value of that parameter (e.g., South Perimeter).

The hierarchical organization of components that is displayed in the Version tree view parallels the containment organization of objects with the IFC data model. As shown in Figure 4 the hierarchical tree view begins at the Project node, which is the root node of all Metracker Version hierarchies. A Project object is also the highest-level container within the IFC data model.

In the example shown in Figure 4, the primary tree branches below Project include Site, Building, Building Storey, Space, and Performance Objective. Each of these branches can be expanded to display their sub-elements similarly to the hierarchical directory structure displayed in Windows Explorer.

The Site branch contains the components within what is referred to as the product model in Figure 1. The truncated example shown in Figure 4 includes one Building, one Building Storey, and five Spaces.

The Performance Objective branch is the key to Metracker. All performance objectives and performance metrics that have been defined for a project are contained within this branch, as discussed in the next sections.

Performance Objectives

A building project begins with a consideration of the various performance objectives of interest to building stakeholders (e.g., owners, designers, operators, occupants, etc.). While primary attention is generally given to space requirements and construction costs, a wide spectrum of objectives may be at least informally considered at this stage, including: life-cycle economics; energy-efficiency; environmental impact; occupant health, comfort and productivity; and building functionality, adaptability, durability, and sustainability. The process of identifying the objectives for a given building project is often referred to as programming. The outcome of programming is most commonly recorded in text that becomes part of design and construction documentation. This documentation may be frequently referenced during design, and occasionally referenced during construction, but then most often is lost from that time forward.

Metracker has been developed to address the issue of lost information by documenting both intended and actual building performance in an archive that is accessible across the entire life cycle of a building project. Within Metracker, the specification of building performance is documented using both qualitative Performance Objectives, and quantitative Performance Metrics. Performance Objectives are intended to document the goals for a building project, such as optimizing whole building energy usage. Performance Metrics are intended to document the criteria for evaluating the achievement of each of the stated Performance Objectives.

In most instances, a high-level Performance Objective will need to be delineated by multiple sub-objectives and metrics that influence its overall satisfaction. This delineation can be organized hierarchically as illustrated in Figure 5 for a hypothetical Performance Objective related to Whole Building Energy Use.

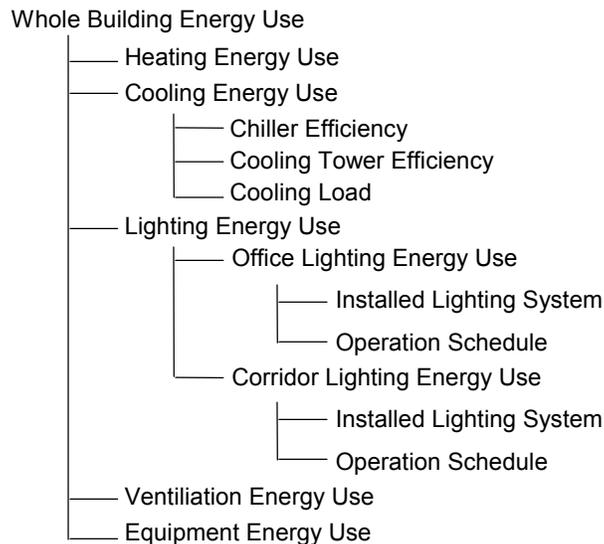


Figure 5: Hierarchical organization of Performance Objectives and Metrics.

Performance Objectives are organized hierarchically within Metracker beginning with a generic Root Performance Objective as shown in Figure 4. A new Performance Objective can be added to the hierarchy by first highlighting the desired parent Performance Objective and then choosing the Performance/New Objective menu option. A dialog box like the one shown in Figure 6 is displayed for user input. After the dialog box has been filled out and the OK button selected, the new Performance Objective will appear in the hierarchy under its selected parent.

A Performance Objective has the following attributes: Name, Date of Specification, Objective Type, Specifier, and Description. The Name provides a user-defined identification. The Date is automatically generated at the time of creation. Objective Type is eventually intended to be selected from a pre-defined set of key performance objectives agreed upon by industry consensus as being of common importance to all building projects. At the current time however, this attribute is user-defined. The Specifier is the person responsible for identifying this objective as important to the specific project. At the present time, this attribute is not used since BSPro does not yet support the Actor class type. The Description is simply a text statement describing the goal for a Performance Objective in more detail.

Figure 6: New Performance Objective dialog box.

Performance Metrics

Performance Metrics are intended to explicitly represent each Performance Objective using quantitative criteria in a format that provides value across the life cycle of a building project. A guiding principle in defining a Performance Metric is to identify a critical variable that measures, reflects, or significantly influences a particular Performance Objective. To be useful across the building project life cycle, each Performance Metric must also be capable of being either predicted or measured at various

stages of the project so that the achievement of each Performance Objective can be evaluated.

The Metracker data definition for a Performance Metric includes the following attributes: Name, Date of specification, Specifier, Metric Type, Description, Source, Data Type, and Data Value(s). The metric Name is a text identifier that is intended in the future to be supplemented with a standardized code for a predefined set of performance metrics. The Specifier and Date of specification document the building process participant concerned with each metric and its date of creation. Again, Specifier is not currently used since BSPro does not support the Actor class type. The Metric Type identifies whether the metric is a Benchmark or an Assessment. Benchmarks specify the intended level of performance, while Assessments record the estimated or measured level of performance. If the metric is of type Benchmark, then a benchmark type is selected from a list including the following: greater than, greater than or equal to, less than, less than or equal to, equal to, not equal to, target with tolerance, range, and distribution. The Source documents the origin of the metric value. For Benchmark type metrics, the Source might be a code, standard, benchmark database, manufacturer data set, or other source of benchmark values. For an Assessment type metric, the Source could be a simulation, monitoring measurement, or other assessment method. The Data Type of a metric is selected from a list including the following: scalar, vector (bar chart), time series, table (2D XY plot), graph (3D XYZ plot), and distribution.

Performance Metric Data Values occur in a variety of Data Types for which there is presently little standardization. For example, chiller efficiency can be specified in numerous ways including a single value parameter (e.g., coefficient of performance (COP) or integrated part load value (IPLV)), multiple data points representing a two-dimensional part load curve for specific operating conditions or a three-dimensional part load surface across the full operating regime, or a mathematical curve or surface function representing these same data. Moreover, the preferred Data Type used for archiving a Performance Metric may change over the life cycle of a project. Continuing with the chiller efficiency example, pre-design planning might specify a desired chiller IPLV. Detailed design simulation might employ a mathematical representation (e.g., a curve fit) of the performance of the selected chiller, based on manufacturer specifications. Commissioning and O&M measurements might subsequently collect multiple time series data points during the chiller's actual operation. The specification of a Performance Metric must therefore be flexible enough to accommodate this variety of Data Types. This issue is discussed in more detail below.

Performance Metrics are contained within the same hierarchy as Performance Objectives as illustrated in Figure 7. Conceptually, Performance Metrics could be delineated hierarchically in a fashion similar to that for Performance Objectives. However, to simplify the hierarchy structure within Metracker, the current implementation only allows Performance Metrics to be children of a Performance Objective. Also, Performance Metrics cannot be nested within each other, but can only be leaf nodes under a Performance Objective parent.

There are three Performance Metrics shown in the data view window for Version Three - AsOperated 1999 that is displayed in Figure 7. All of these metrics are children of the Whole Building Energy Use objective. The metrics appear here in reverse chronological

order and include Baseline, Schematic, and As-Operated 1999 data sets for Whole Building Monthly Electric EUI, as shown. The Baseline metric is a benchmark. The other two metrics are assessments.

A list of Performance Metrics containing both Benchmark and Assessment values may be archived for each Performance Objective over the life cycle of a building project. There may be an initial benchmark established in pre-design planning, updated benchmark values and simulated assessment values determined during design, short-term measurements from commissioning, and long-term monitored values. The list of archived Performance Metrics related to each Performance Objective will appear in the data view window.

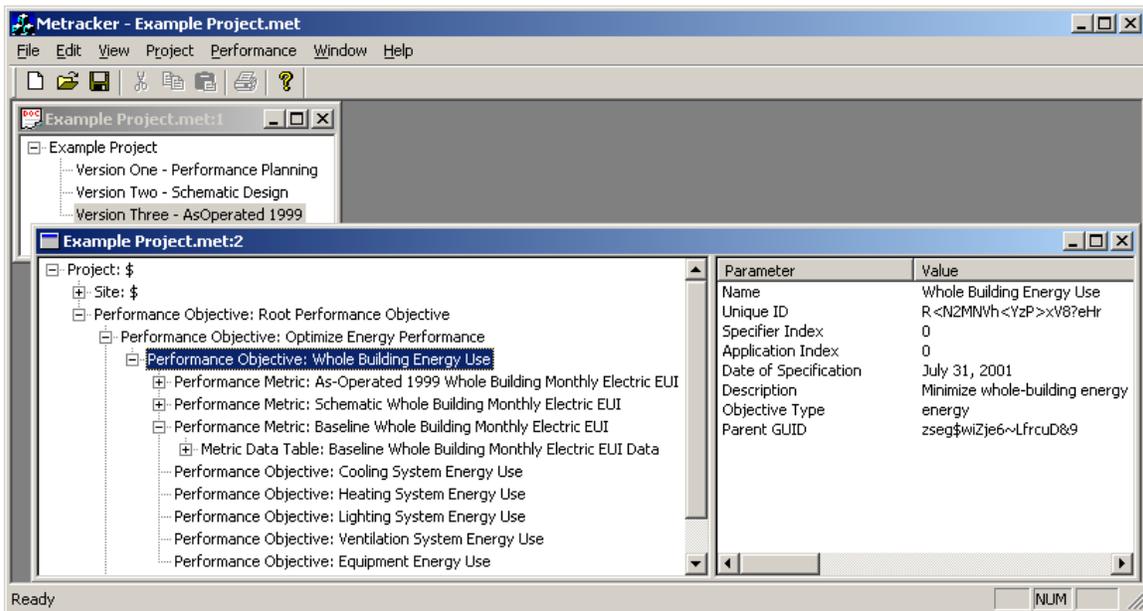


Figure 7: Performance Metrics within the Performance Objectives hierarchy.

A new Performance Metric can be added to the hierarchy by first highlighting the desired parent Performance Objective and then choosing the Performance/New Metric menu option. A dialog box like the one shown in Figure 8 is displayed for user input. After the dialog box has been filled out and the OK button selected, the new Performance Metric will appear in the hierarchy under its selected parent objective.

Performance Metric

Name: Baseline Whole Building Monthly Electric EUI

Date: August 01, 2001

Specifier: [Dropdown]

Metric Type

Benchmark Type: Less Than or Equal To [Dropdown]

Assessment

Description: Baseline whole building monthly electric energy use intensity

Data Type: Vector (Bar Chart) [Dropdown]

Source: Benchmark database

Import file: BaselineMonthlyElecEUI.csv Browse...

OK Cancel

Figure 8: New Performance Metric dialog box.

The Browse button on the Performance Metric dialog box can be used to search for the desired ASCII data file on your computer or network. Metracker assumes that these files of tabular data are named with a suffix of .csv, and will automatically display these files in the browse dialog. However, any appropriately formatted data file can be imported into Metracker.

Once the Performance Metric has been added to the hierarchy its parameters (attributes) can be displayed by highlighting the metric, as shown in Figure 9. Note that at the present time, several of the values for metric parameters appear as index values instead of more easily understood text values. Specifier Index is a zero-based index into the list of registered Actors. Application Index is an index into the list of registered software applications. The Is Benchmark value is 1 for TRUE (i.e., the metric is a benchmark), or 0 for FALSE (i.e., the metric is an assessment). The Benchmark Type is a zero-based index into the following enumerated list of types: Greater Than, Greater Than or Equal To, Less Than, Less Than or Equal To, Equal To, Not Equal To, Target with Tolerance, Range, and Distribution. The Data Type is a zero-based index into the following enumerated list of types: Scalar, Vector (Bar Chart), Time Series, Table (2D XY Plot), Graph (3D XYZ Plot), and Distribution. The Data Table ID is a reference to the IFC container for the actual data values associated with the metric. The non-readable strings of characters are unique identification strings that are used to maintain the hierarchy of these objects.

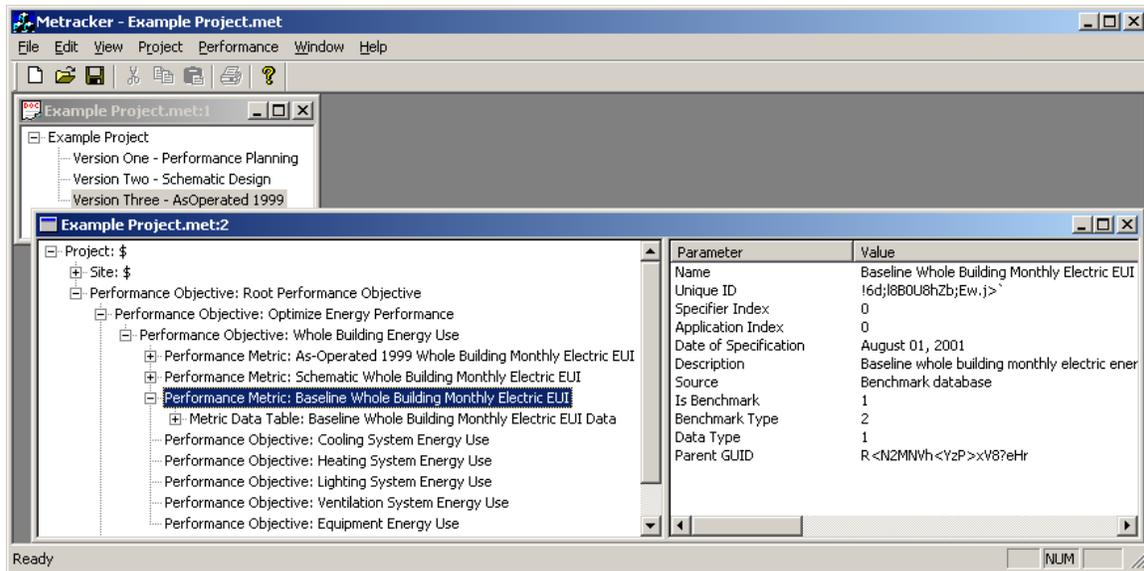


Figure 9: Performance Metric parameter values.

Performance Metric Data Types

A set of data values, which are the actual numeric value(s) for a Performance Metric, are archived according to the metric Data Type. The type that is manually entered by the user in the dialog box shown in Figure 8 may be overwritten when the data are entered/imported into Metracker, to assure that the data type matches the entered/imported data.

Note the Import File edit box at the bottom of the dialog box in Figure 8. The current method for input of Performance Metric Data Values is by importing the data from an external ASCII data file. The format of the ASCII data file is “comma separated values” (CSV), with the first line in the file containing a name and units of measurement heading for each column of data, and subsequent lines containing only numeric values. Each column heading should take the format of “Variable Name [unit of measurement]” where Variable Name can be any string that identifies the data within the column. The unit of measurement is currently any arbitrary string that specifies the data units enclosed within square brackets. In the future this convention will be replaced by a more standardized method of specifying the data unit of measurement following IFC conventions. The metric data file may contain one or more columns based on the metric Data Type. Metracker determines the metric Data Type based on the number of columns in the file, and the associated headings in the first row.

- Scalar and Vector Data Type files contain a single column.
- 2D XY Plot files contain two columns; the first column is the X-axis data value and the second column is the Y-axis data value.
- Time Series data type files also contain two columns where the first column is a timestamp and the second column is the associated data value. The heading of the first column must begin with the letters “Date” (not case sensitive). The timestamp

values must take one of two forms, either “MM/DD/YYYY HH:MM:SS” or “MM/DD HH:MM:SS” (month, day, 4 digit year, hour, minute, and second entries). The alternative with no year entry is allowed for simulation time series with no explicit year. NOTE: The IAI BS-8 project on extending the IFC data model to support energy simulation tools is addressing the topic of modeling time series data. The results of this work will be folded into Metracker.

- Distribution data type files also contain two columns; the first column is the X-axis data value and the second column is the frequency-of-occurrence of that data value. The heading of the second column must begin with the letters “Frequency” (not case sensitive).
- 3D XYZ Plot data type files contain three columns of data for the X, Y, and Z axes.

Metric Data Visualization

Metracker provides data visualization graphics for Performance Metric data that have been archived in a Project Version. The current version of Metracker graphs Vector, 2D XY, Time Series, and frequency Distribution data types. Vector data are graphed as bar charts. 2D XY, Time Series, and frequency Distribution data are graphed as two-dimensional plots, where time series data produce plots with time as the X-axis.

There are two options available for visualizing Performance Metric data. The data set associated with an individual Performance Metric can be graphed, or the data sets associated with all Performance Metrics that are children of a single Performance Objective can be simultaneously graphed on the same plot. The current version of Metracker can only graph multiple Performance Metric data sets if the Data Type of all data sets are identical. This will be enhanced in the future so that differing, but compatible data types can be graphed together for comparison (e.g., a Scalar benchmark along with a Time Series assessment).

A graph for a single Performance Metric is shown in Figure 10. This graph was displayed by first highlighting the desired metric, As-Operated 1999 Whole Building Monthly Electric EUI, in the Version data view hierarchy, and choosing the View/Metric Chart menu option.

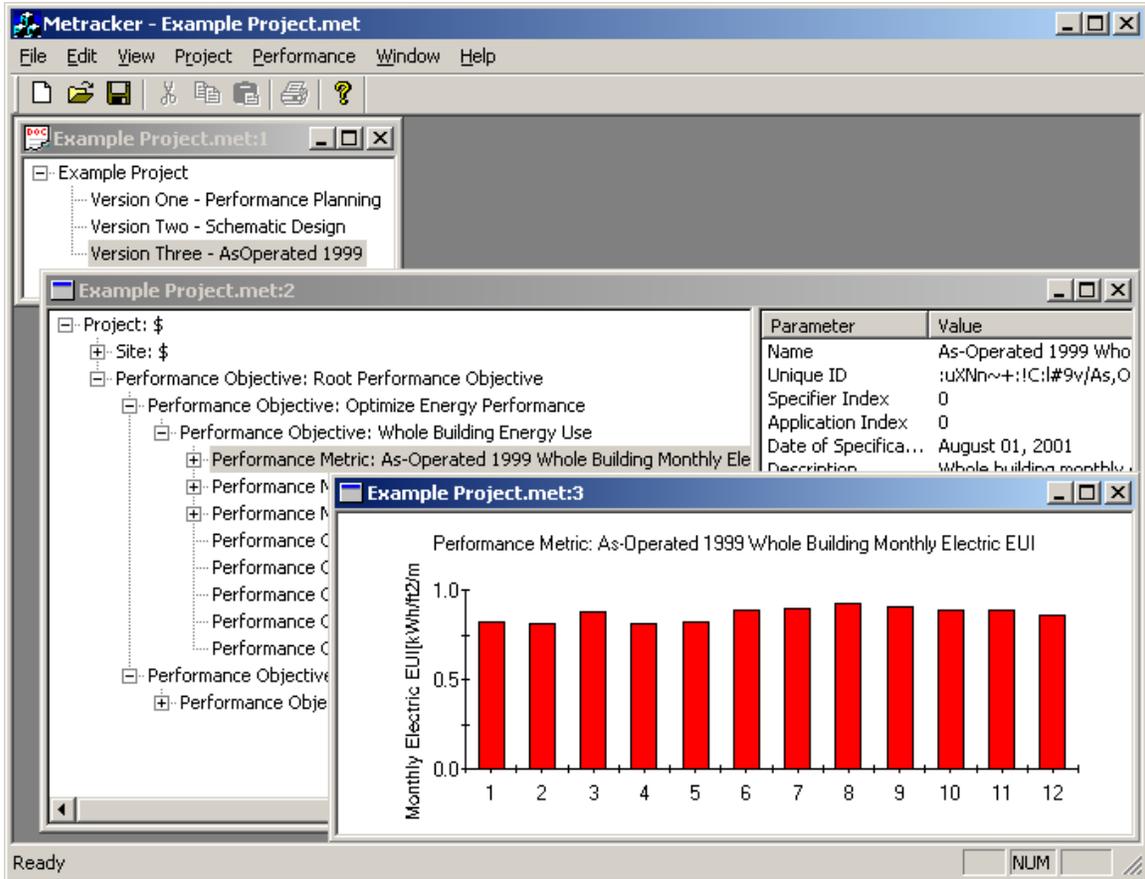


Figure 10: Graph of a single Performance Metric.

A graph of all Performance Metrics that are children of a single Performance Objective is shown in Figure 11. This graph was displayed by first highlighting the desired objective, Whole Building Energy Use, in the Version data view hierarchy, and choosing the View/Metric Chart menu option. As mentioned above, for this visualization option to work correctly, the Data Type for all metrics under a given objective must be identical.

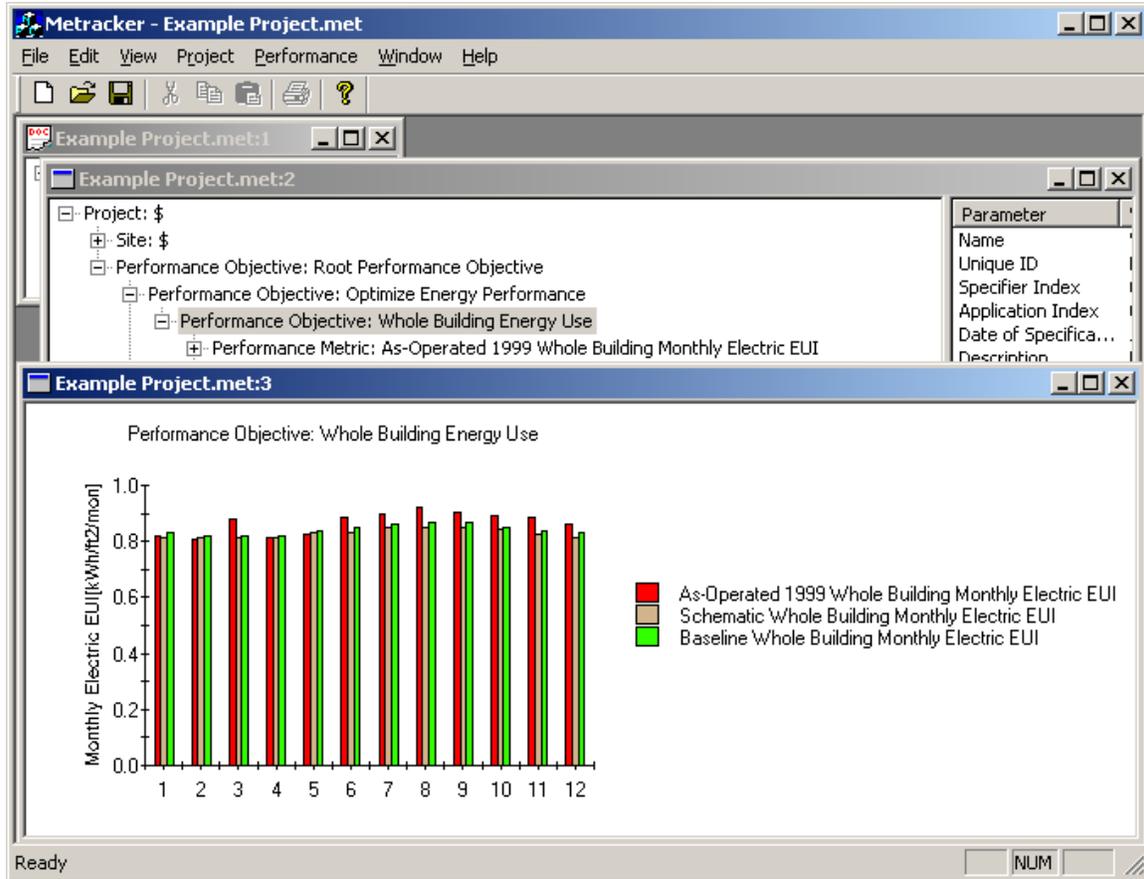


Figure 11: Graph of all metrics associated with a Performance Objective.

The data visualization window can be resized/repositioned by the user. Also, right clicking on the window causes a dialog box to appear that provides access to a variety of graph attributes for more user control.

Actors

The current version of BPro COM-Server used in developing Metracker does not support the Actor class type. This class type will be supported in the future.

Installing Metracker

This section gives instructions for installing the Metracker and BPro COM-Server software on a target computer. These instructions, along with a step-by-step demonstration scenario using a sample Metracker Project archive file, are included in the accompanying "Metracker Demonstration Scenario.doc" document.

1. Run the "SetupMetracker.EXE" installation program to begin installation of Metracker, and follow the onscreen prompts. It is highly recommended that you accept the default directory for installation (*C:\Metracker*) since the included example files must be placed in the proper subdirectory (*C:\Metracker\Archive Files*) to function correctly.

2. BSPro COM-Server must also be installed on the target computer in order to run Metracker. Run the “BSPro_e+.EXE” installation program to install BSPro.
3. The Metracker installation program creates a shortcut under the Windows Start menu in the user prescribed program group (Metracker by default). Select the *Metracker* menu shortcut from under the Windows Start menu to start the program.

Acknowledgements

This work was supported by the California Energy Commission, Public Interest Energy Research Program, under Contract No. 400-99-012 and by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology, State and Community Programs, Office of Building Research and Standards of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

- IAI. (2001) International Alliance for Interoperability. World Wide Website URL: www.iai-na.com
- ComponentOne. (2001) Olectra Chart. World Wide Website URL: <http://home.component1.com/>
- Olof Granlund. (2001) BSPro COM-Server. World Wide Website URL: www.bspro.net