Team Process Data Warehouse – Process Dashboard Notes for Core Data Schema

The "Core Schema" document describes various database constructs that are present within the data warehouse. Since the warehouse itself is tool-agnostic, that document remains abstract and does not dwell on the details for any particular tool.

Of course, developers who wish to write queries and reports against Process Dashboard data must understand how the data in the Process Dashboard maps to the objects in the warehouse. This document provides that information.

This document describes version 1.5.3 of the Data Warehouse logic, which is bundled with:

- Process Dashboard, version 2.3
- Process Dashboard Enterprise Server, version 3.6.1

Dimension: Organization

The Process Dashboard does not currently have any construct that would map to the "Organization" dimension. As a result, tools may eventually need to be provided that allow people to load the org chart into the database, and the ETL tool may need to allow the configuration of the organizations that should be used for various Process Dashboard datasets.

This functionality would take time to write. But for the initial iteration, the warehouse will most likely be used in small, trial deployments. As a result, the ability to analyze data by organizational divisions will not be critical during the initial iteration.

Because of this, the Organization dimension could potentially be excluded entirely from the scope of the initial iteration. However, since Organization will eventually play a central role in the management of permissions and security, it was deemed worthwhile to include the dimension in the first iteration.

To resolve this potential conflict in the initial iteration scope, the following approach will be used:

- Tables will be created for the Organization dimension in the warehouse.
- However, no tools will be written for managing the contents of this dimension, or for mapping Process Dashboard datasets to specific organizations.
- As a result, the Process Dashboard ETL logic written for the first iteration will simply create a single "global" organization with an ORGANIZATION_KEY of 1, and will use that organization for all data that is loaded.

This approach will allow downstream tools and reports to design against the schema for the organization dimension, even if the dimension tables do not contain detailed data.

Dimension: Team

The Team dimension will eventually be expanded to include information about the project teams within a program and the subteams (for example, Dev vs QA) within those projects. Unfortunately, those rich subteam descriptors are not currently captured within the Team Dashboard. New user interfaces will need to be added to the Team Dashboard to capture this information.

In the meantime, and in keeping with the minimal scope of the first warehouse iteration, the Process Dashboard ETL logic will create a single entry in the Team dimension for each Team Dashboard that is imported. So a dataset known as “XYZ Team Dashboard” might create an entry in the Team dimension with the team name “XYZ.”

When the warehouse is running in "lightweight embedded mode" within the Team Dashboard, this means that the Team dimension will contain exactly one row. The Enterprise Data Warehouse will have many rows in this dimension, one for each imported Team Dashboard.

Dimension: Person

An entry will be created in the Person dimension for each personal Process Dashboard instance. In the typical usage scenario, where each individual has a single Process Dashboard database that they use to capture data for all of their current and historical projects, this will result in a single row for each person.

Dimension: Data Block

On a team project, the data collected by individuals is exported to “PDASH” files on a shared network drive. The ETL process will import data from these PDASH files, and create a Data Block object representing each PDASH file.

Dimension: Project

Within the Team Dashboard, “Team Project” entities are created in the Hierarchy Editor to describe the collaborative effort of a group of individuals on one or more project iterations. Each “Team Project” entity will map directly to an entry in the data warehouse “Project” dimension.

In the initial iteration of the warehouse, “Master Projects” created in the Team Dashboard will not be mapped into any particular database construct.

Dimension: WBS Element and Task

The Process Dashboard’s WBS Editor captures a single hierarchical structure that describes both components and tasks. This differs from other TSP tools – like the SEI spreadsheet – which maintain separate data structures for hierarchy-vs-task.

In the interest of interoperability, the dashboard ETL logic will create a similar delineation:

- In the WBS Editor, items that have a “component” type will be recorded in the WBS Element dimension.
- In the WBS Editor, items that have a “task” type will be recorded in the Task dimension.

Dimension: Process

In the Team Dashboard, two major concepts merit consideration as high-maturity process definitions: Metrics Collection Frameworks (MCFs) and Common Team Workflows. Accordingly, both are captured in the warehouse:

- An entry is made in the Process dimension for each MCF. Corresponding entries are made in the Phase dimension for each MCF phase.
- An entry is made in the Process dimension for each Common Team Workflow in every team project. Corresponding entries are made in the Phase dimension for each of the steps in a workflow.

Each step in a workflow is associated with a particular MCF phase. (This association is defined by the user when they click the icon that appears in front of the workflow step name.) These mappings (from workflow step to MCF phase) are recorded in the Phase Mapping table.

Dimension: Plan Item

In the warehouse, a Plan Item object will be created for each node in the WBS (both components and tasks). This will be a one-to-one mapping.

If a task in the WBS was created by applying a workflow, its Plan Item will have a Phase object describing the relevant workflow step. If a WBS task was created manually (not from a workflow), the Plan Item will have a Phase object describing the MCF phase that was selected for the task. The Phase Mapping table can be used to convert all phases to MCF phases for any analysis that requires it.

If an individual has elements in their personal plan that are not present in the WBS (for example, because they created extra nodes with the Hierarchy Editor, or because they have a leftover node in their plan that was deleted from the WBS at some time in the past), Plan Item objects will be created for those personal plan elements as well.

Milestones

In the WBS Editor, milestones are edited in their own window. This differs from other common planning tools (such as Microsoft Project), where a milestone is just another item in the project plan. Data interoperability is an important architectural goal of the warehouse, so the database schema follows the industry standard paradigm. Accordingly, WBS milestones are recorded in the Plan Item dimension, as rows with the following distinct characteristics:

- The "task name" of the plan item will indicate the name of the milestone, and the "phase" of the plan item will point to a special "Milestone" pseudo-phase. A query can look for the phase with identifier "`*Unspecified*/Milestone`" to find all project milestones.
- Milestone plan items will be associated with the WBS Element that is the "root" of the project.
- The "ordinal" field will indicate the relative ordering of milestones, and can be used to sort the milestones in the order they appear in the WBS Editor's milestones window.
- Plan Item attributes will be created to record milestone "hidden" and "defer sync" flags, as well as the milestone commit date.

Dimension: EV Schedule

The dashboard does not require a 1-to-1 mapping between team projects and EV schedules. Individuals and teams are free to create multiple schedules for a particular project and/or schedules that span projects. The following schedules will be captured in the EV Schedule dimension:

- Any EV Rollups that are defined in the Team Dashboard (as seen via “C > Task & Schedule”)
- Any personal schedules that have been added to those rollups

This means that if an individual has a personal schedule in their dashboard, but that personal schedule has not been added to any team rollup schedule, the resulting data will not be included in the warehouse.

Dimension: Baseline

In the dashboard, baselines are saved in the Task & Schedule window, and associated with a particular earned value schedule. These baselines will be listed in the Baseline dimension, with a baseline target type of “EvSchedule” and a target key pointing to the EvSchedule object that the baseline was saved for.

Baselined estimates and completion dates are saved in various fact tables, as described later in this document. Baseline facts have the following characteristics:

- They have a false value in the row_current_flag column
- The row_eff_start_date and row_eff_end_date columns both contain the date the baseline was saved.
- The baseline_key column contains the unique key of a row in the Baseline dimension, indicating the baseline this fact is associated with.

People wishing to work with baseline data should follow this query pattern:

1. Identify a particular EV schedule that you wish to report against.
2. Find that EV schedule in the EV Schedule dimension, and retrieve its numeric key.
3. Search the Baseline dimension for rows whose target type is “EvSchedule” and whose target key is the number found in step 2. This will return a row for each baseline that has been saved for the schedule; if you are interested in the active baseline, search for the Baseline whose “active” flag is true.
4. Retrieve the numeric key of the baseline of interest.
5. Search fact tables (e.g. Task Date Fact and Task Status Fact) for rows with that baseline key.

Dimension: Defect Type Standard

When teams create their own defect type standards, these will be written into the Defect Type dimension. The regular PSP^{SM} type standards will be loaded as well.

Due to the way defect types are recorded in the dashboard, it is not uncommon for defects to have "legacy" types which were assigned before the team established a common standard. Those values may be written into the Defect Type dimension with a DEFECT_TYPE_STANDARD_NAME of "Unrecognized Defect Type."

Also in the dashboard, Defect Type Standards are identified solely by their name, and not by any unique internal identifier. As a result, if two different teams create a defect type standard with the exact same name, the warehouse will contain a single standard that includes the union of all defect types defined by both teams.

Dimension: Size Metric

The Size Metric dimension will include entries for each of the size metrics that were defined by a metrics collection framework. Ad-hoc size metrics (such as the ones that can be attached to tasks in the WBS) will not be recorded in the warehouse at this time.

Fact: Task Dates

Task Date rows will be created for each of the tasks that appears in the "Flat View" of the team EV report. In practical terms, this means that there will be generally be one Task Date row for each combination of [Leaf Task, Assigned Individual, Measurement Type]. This table will use the Measurement Types Plan, Replan, Forecast, and Actual. Level of effort tasks and "manually removed" tasks will not be included in this table.

Baseline dates can also be found in this fact table. When a baseline is saved, all of the task date fact rows are copied and marked with a baseline_key pointing to the baseline in question. As a result, you can query to see what the Plan, Replan, Forecast, and Actual dates looked like at the moment the baseline was saved. If you are interested in the value that appears in the "Baseline" column of dashboard reports, you should look for the baselined version of the Plan date.

Fact: Task Status

Task Status rows will be created for each combination of [Leaf Task, Assigned Individual]. In contrast to the Task Dates table, this table will include level of effort tasks and "manually removed" tasks.

Task status rows are also created for saved baselines, as well. When a baseline is saved, all of the task status rows are copied and marked with a baseline_key pointing to the baseline in question. If you are interested in the value that appears in the "BT (Baseline Time)" column of dashboard reports, you can sum the planned times for all task status rows that have the appropriate baseline_key.

^{SM} PSP is a registered service mark of Carnegie Mellon University.

Fact: Time Log

Time log entries from individuals will be recorded in the fact table in a straightforward manner.

Fact: Defect Log

Defect log entries from individuals will be recorded in the fact table in a straightforward manner.

Each defect has an entry representing the phases were the defect was injected and removed. Currently, these will always be recorded as MCF phases. A future release of the dashboard and the warehouse logic may make it possible to analyze defect injection and removal patterns by workflow phase instead.

Fact: Size

On a team project, individuals capture size in two places.

When using a PSP task or a PROBE-enabled Team Workflow, individuals enter size data on the Size Estimating Template and the Project Plan Summary form. During the ETL process, exactly two rows (one Plan and one Actual) will be created in the Size Fact table to represent each PSP task or PROBE-process enactment. The ETL process will pull the aggregate size accounting metrics (Base, Added, Deleted, etc) from the Project Plan Summary, and copy them into the two fact table rows. (The warehouse will not contain granular data of individual rows from the Size Estimating Template.)

Individuals also capture size data on the Size Inventory Form. That form allows multiple rows to be created for a particular node in the WBS. The ETL process will:

- Look at each WBS component and find the rows that are attached
- Discard any rows that have an “Inspected” size type. (These are artificial rows created within the dashboard to facilitate the calculation of review rate for the inspectors of a work product.)
- Group the remaining rows by size metric (e.g. LOC, Reqts Pages, etc)
- For each group, create a sum of the planned numbers and a sum of the actual numbers
- Record these numbers into two rows in the Size Fact table
    - Base/Deleted/Modified/Reused will be zero
    - Added/Added & Modified/Total will all hold the summed value

Fact: Plan Item Attribute

The Plan Item Attribute fact table will be used to capture the following types of information about the items in the WBS:

- The labels associated with each WBS item. A row will be created in the Attribute dimension with an ATTRIBUTE_IDENTIFIER of `plan_item.label`, and a row will be created in the Attribute Value dimension containing the text of a single label. With these in place, rows will be created in the Plan Item Attribute Fact table to document the labels that are associated with each plan item.
- Values entered into custom WBS columns. These values are stored in the same way labels are stored, but with an ATTRIBUTE_IDENTIFIER derived from the name of the custom WBS column.

Fact: Plan Item Dependency

The WBS Editor allows teams to declare dependencies between components and tasks. These will be recorded as rows in the Plan Item Dependency table, with a type of “Finish-to-Start” and a lag time of 0.

As described above, milestones are recorded in the warehouse as Plan Items with special characteristics. Accordingly, the association of Plan Items with Milestones will be recorded as just another Plan Item Dependency. Specifically, a row will be created in the Plan Item Dependency table for each leaf node in the WBS that has been assigned to a milestone. (Rows will not be created for parent WBS nodes that have children.) These rows will be Finish-to-Start dependencies (based on the industry-standard paradigm that a milestone is an instant in time when the given collection of work will be done) with the WBS node being the predecessor, and the milestone being the successor. Since the WBS Editor restricts each node to point to a single milestone, queries can rely upon finding no more than one milestone dependency for any given Plan Item.

In addition, since milestones have an explicit ordering within the WBS Editor, Dependency rows will be created to describe dependencies between adjacent milestones (for example, to indicate that Milestone 1 precedes Milestone 2).