Team Process* Data Warehouse – High-level Schema Plans

The core of the data warehouse will be a relational database schema that contains project metrics and metadata. The design of this schema will follow best practices for dimensional modeling. High level plans for this schema are described in this document.

Data Warehouse Bus

Conformed dimensions are a crucial part of a proper dimensional model. The table below shows the relationships between several of the primary fact and dimension tables in the warehouse:

Dimensions: / Facts:	Organization	Person	Team	Project	WBS Element	Task	Process	Metric
Time Log	X	X	X	X	X	X	X	
Defect Log	X	X	X	X	X	X	X	
Size Data	X	X	X	X	X	X		X
Process Metrics	X	X	X	X	X	X	X	X
Earned Value	X	X	X	X	X	X		

High-level descriptions of many of these dimensions and facts are included in the data warehouse requirements document.

Schema Design Challenges

The Team Process Data Warehouse initiative presents several interesting challenges that require special attention during the design process. Several of these challenges are described in the pages that follow, along with high-level descriptions of proposed solutions.

* TSP[SM] is used by teams working in a wide variety of problem domains (e.g. software, hardware, services). Since these activities are not limited to software, the name "Team Process Integrated" and the acronym "TPI" are used in this document to describe the full range of TSP-inspired high-maturity processes, and to avoid improper use of Carnegie Mellon service marks. TSP is a service mark of Carnegie Mellon University. Carnegie Mellon University has neither contributed to nor evaluated the contents of this document.

Generalizability across Multiple Organizations

Data warehouse initiatives are typically spearheaded by a particular organization. Within an organization, it is possible to identify stakeholders, data sources, analytical goals, and the relative priority of deliverables. For a generalized Team Process Data Warehouse initiative, these drivers are less clear.

Without question, a small number of stakeholders can be identified to drive the requirements. But if the resulting product does not meet the needs of the larger TSP community, this will result in a significant lost opportunity. If a warehouse can be developed that meets the needs of many organizations, this broader user base will enable:

- A larger group of stakeholders to fund future data warehouse enhancements
- A potential market for third-party vendors, who could provide value-added reports that work against the common data warehouse

These are substantial long-term benefits, even to the "small number of initial stakeholders" described in the paragraph above. Thus, it is in the interest of everyone involved to identify requirements and strive for a design that can accommodate the larger TSP community.

However, doing so will result in new design constraints that may be uncommon for data warehouse initiatives:

- The inability to precisely define certain warehouse constructs, leaving these to be defined by an organization at deployment time
- The need for some warehouse constructs to be general-purpose, allowing for different uses by different organizations
- The need for the core schema to support data streams from disparate data sources which may not be known in advance to the warehouse designers
- The need for the warehouse to be extensible, allowing the capture of custom data that is specific to a particular organization

To address these constraints, this data schema design effort should:

- Collect input from as many stakeholders as possible
- Seek to understand various data sources (especially various TSP/TPI tools) and strive to align the designs of core schema objects so they can accommodate data from these various sources
- Explicitly provide mechanisms for flexibility and extensibility

Input from the TSP community (and especially information about other TSP/TPI tools) has been gathered in support of the core schema designs.

Within the warehouse, the Taxonomy and Metadata concepts have been introduced to provide explicit support for customizability and extensibility.

Generalizability across Multiple TSP/TPI Tools

Data warehouse initiatives almost always pull data from a number of disparate data sources. To deliver maximum analytical value, it is important to provide coherent views that integrate this heterogeneous data.

In a TPI context, these disparate data source could include:

- Data collected in more than one TSP/TPI tool (for example, the Process Dashboard and the SEI Excel Workbook) and from many instances of those tools (for example, many different Team Dashboard and many different Excel Workbooks)
- Data collected in external corporate systems, such as defect trackers
- Data harvested from other COTS products, such as ALM or version control systems

At first glance, this data integration effort might appear to be a garden-variety integration effort, not unlike the integration efforts faced by other data warehouse initiatives. But upon closer inspection, the Team Process Data Warehouse initiative faces some significant inherent challenges:

- A traditional data warehouse initiative would be able to identify the set of data sources during the requirements and design process. For the Team Process Data Warehouse initiative, some of these data sources cannot be identified by the core developers, and will only be identified by a particular organization at deployment time.
- Because a traditional warehouse initiative can identify data sources during the design phase, a comprehensive set of ETL process can be specifically constructed to aggregate and normalize the data from those known sources. For the Team Process Data Warehouse, the need for data source flexibility means that ETL processes may need to run independently from one another.
- The independence of various ETL processes will implicitly require that:
    - Data in the warehouse must be tagged with information about the data source and/or ETL process that wrote it
    - The use of surrogate keys will be critical (as always for any data warehouse initiative), but flexibility will need to be provided allowing disparate ETL processes to record a variety of source system identifiers
- Core schema objects will need to include careful definitions of standard data elements, so the authors of various ETL tools will be able to write consistent information into these standard fields. (For example, it would be disastrous if one ETL tool wrote planned task time in minutes and another wrote this time in hours.)

Data Privacy

TPI teams collect a great deal of fine-grained information. This information is ideally suited for planning, tracking, and process improvement. But in the wrong hands, it is dangerously easy to misuse. A successful TPI initiative can be thwarted by a manager who decides to reward or punish individuals based on the metrics they collect.

Metrics can be abused at all levels, but personal metrics are particularly dangerous. Although many organizations have a decent respect for the sanctity of TSP data, there are many other organizations where managers seek to use TSP data to inappropriately micromanage individuals. If the data warehouse enables this, it could unwittingly become a "Weapon of Mass Process Improvement Destruction."

Unfortunately, the flexibility of a data warehouse means that it is trivially easy to slice and dice data in arbitrary ways. This means that a report of "productivity for each individual in the entire organization" would be just as easy to create as a report of "time by process phase." The only way to avoid this data privacy trap is to ensure that Personally Identifiable Information (PII) is never associated with sensitive metrics data.

Accordingly, the warehouse will not provide a way to tie personal metrics data (such as time, size, and defects) to a particular individual. Coaches and mentors who need to review personal data should perform that task in the original TSP/TPI tool instead. Of course, experienced TSP experts understand the sanctity of personal data, so they will not have any qualms about this restriction.

Still, it will not be possible for the warehouse to discard the concept of personal identities altogether. There are many legitimate analysis questions that will require knowledge of the individuals who collected a piece of data:

- During analysis of inspection data, it will be important to be able to count the number of individuals who participated in a particular review.
- A manager who is intent on viewing personal data should not be allowed to do so simply by drilling down (for example, by focusing on a single project component that was written by a known individual). Accordingly, the reporting mechanism should be able to count the number of individuals whose data contributed to a particular query, and redact data if this count falls below some configurable threshold.
- An organization might wish to group individuals into various categories, and compare the aggregate data from those categories. For example, they might wish to perform an analysis measuring the return on investment for sending individuals through a PSP[SM] Advanced course. Or they might want to perform an analysis that excludes individuals who are known to be willfully ignoring the process.

These requirements will require the warehouse to contain some measurement of personal identity. Accordingly, the ETL process will need to generate an arbitrary surrogate key for each real-life individual.

[SM] PSP is a registered service mark of Carnegie Mellon University.

This key can be numeric, as long as it does not reveal any clues about the individual's true identity. (For example, it should not be a corporate employee ID or other number which could be looked up in a directory.)

This arbitrary surrogate key will be used to tag the data collected by each individual. A simple "count distinct" clause can then be used to identify the number of individuals whose data contributed to a particular query.

To meet the analysis requirement described in the final bullet above, taxonomies can be used in the warehouse to classify individuals into various categories. These associations will need to be stored in an external system, outside the data warehouse. The ETL process will need to read this information and create the appropriate associations in the warehouse between an individual and a taxonomy category. With these pieces in place, a warehouse report could then apply a filter to show data from individuals in a specific taxonomy category such as "PSP Advanced."

Exceptions

The data privacy considerations described above are of utmost importance in a data warehouse that holds data for an entire organization. In such a warehouse, personally identifiable information (PII) must be encrypted in some way (or omitted from the warehouse entirely) to protect the privacy of personal data.

However, an organization-level warehouse is not the only possible deployment scenario. The data warehouse design can scale downwards as well, to hold the data for a single team project or even a single individual. When a tiny warehouse instance is created for such a purpose, it is quite feasible to limit physical access to that warehouse. For example, a team could create a tiny warehouse that only contained the data from their project, and they could limit warehouse access to project team members. These individuals already have access to the same data in the source system, so privacy concerns are significantly reduced. For these small deployments, it may be desirable to disable the encryption of PII. Doing so could enable the creation of powerful reports to assist the team with self-management.

Changes to the Structure of a Project Plan

It will be very important for the data warehouse to track history. Users will want to see the history of changes to both plans and data.

Within the scope of a TPI project, one item in particular introduces a special challenge in this regard: the hierarchical work breakdown structure for a project plan. This structure can change over the life of the project, and it will be critical to track those changes.

The traditional approach to this requirement would be to model the work breakdown structure as a Type 2 SCD. Of course, due to the hierarchical nature of this data, the work breakdown structure for a project can become a rapidly changing dimension. Consider, for example, the following change, in which Component A has been renamed to Component A':

Project
• Component A
  ○ Subcomponent 1
  ○ Subcomponent 2
  ○ Subcomponent 3
  ▪ Subcomponent 3a
• Component B
• Component C
	Project
	• Component A'
	  ○ Subcomponent 1
	  ○ Subcomponent 2
	  ○ Subcomponent 3
	  ▪ Subcomponent 3a
	• Component B
	• Component C

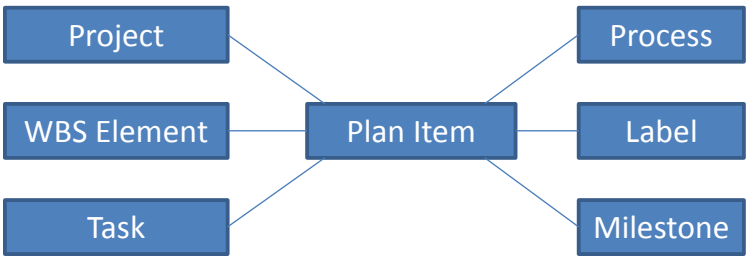
In this simple example, new dimension rows would need to be created for Component A and for all of its descendants. In the example above this would translate to 5 new table rows. But in real-world plans, the number of affected rows could be many times higher – in some cases, in the hundreds.

This in and of itself does not pose an immediate problem. Unfortunately, the true problem arises when fact table data is considered. Each node in a work breakdown structure will typically have hundreds of associated fact table rows in dozens of different fact tables, recording data such as time/defect log entries, planned and actual sizes, etc. When you multiple these hundreds of fact rows with potentially hundreds of affected WBS dimension rows, it becomes apparent that a small change to a WBS item could lead to the invalidation and regeneration of thousands of fact table rows.

Of course, users will also wish to track changes to those facts (for example, a change in the planned time for a given task). That change tracking would become more difficult, since true changes to a fact table row will be buried underneath the large number of fact table rows that were invalidated and regenerated as the result of an unrelated WBS change.

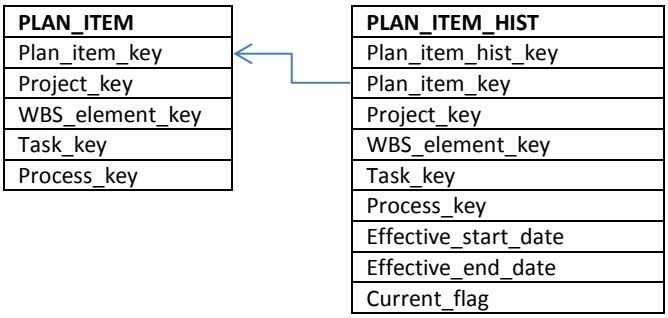
To resolve this problem, the data warehouse will include a secondary construct called a "Plan Item." In most cases, a plan item will correspond directly to a particular component or task within a team plan. However, this direct correspondence will not be strictly required if a particular ETL process has a compelling reason to create several "plan item" entities for a single component or task.

The Plan Item construct will pull together several closely related dimensions in the warehouse, as depicted below:



Conceptually, this means that a plan item is associated with a particular WBS element within a particular project. The Task association is optional; if missing, this item represents a component or assembly. If present, this element represents a task underneath the given component. When the task association is present, the process association can be provided to indicate the process and phase that were used to generate this task underneath this WBS element. Finally, label and milestone associations can be provided to attach additional information to the plan item.

Current and historical information about plan items will be recorded in two separate tables, following the design pattern for a Type 4 slowly changing dimension:



In accordance with the Type 4 pattern, the PLAN_ITEM table will be a Type 1 SCD. It will include a surrogate key (assigned to the plan item during the ETL process) along with the current values of project, WBS element, task, etc.

The PLAN_ITEM_HIST table will be a Type 2 SCD that tracks changes in a given plan item. It will include the surrogate key of the plan item, tying together the multiple history rows for that item. It will include start and end dates indicating the effective period of the row. And it will include the values of the project, WBS element, task, etc during that range of time.

Elsewhere throughout the data warehouse, fact tables will use the plan_item_key to indicate the project, component, and task that are associated with a given fact. (For example, a time log fact table would include a plan_item_key column to record the project/component/task that a given time log entry is associated with.) Since the PLAN_ITEM table is a Type 1 SCD, these fact table rows will not be invalidated by changes to the plan structure. For example, if someone were to rename a component (as

described in the example above), all of the historical time log entries for tasks under that component would still be attached to their respective tasks and the fact table rows would not need to be modified.

The vast majority of analyses in the warehouse will target the current hierarchical structure. The PLAN_ITEM Type 1 SCD makes this task simple. However, the PLAN_ITEM_HIST dimension will make it possible to:

- View the history of changes that have been made to the structure of the plan over time
- Reconstruct the exact appearance of planned and actual data at some historical point in time
- View current metrics data through the eyes of historical project structure

In a small number of cases, some fact tables may choose to include a foreign key directly to the PLAN_ITEM_HIST table. This would be appropriate, for example, in the case of a data mart that stores daily snapshots of precomputed values such as forecast completion date. Since the table row is inherently tied to a specific instant in time when the snapshot was computed, it is appropriate to tag the data with a plan_item_hist_key indicating what the plan looked like at that instant in time.

Process Definitions

In meeting #3 of the TSP launch process, teams define the processes they will use. These processes are typically described as a sequence of defined steps. Later in the planning process, processes are used to create tasks in a plan.

Once data has been collected, PSP and TSP high-maturity behaviors provide teams with a wealth of tools they can use to analyze data from a process. Accordingly, the warehouse needs to:

- Include information about the custom process definitions that have been created by organizations, teams, and individuals
- Tag the data that is collected (for example, time and defect data) with information about the process that was being enacted
- Allow detailed analyses to slice and dice data by process and by process phase.

However, many of those behaviors become challenging in the face of real-world process definitions, because:

- Every organization and team will have its own custom process definitions. In some tools, individuals could also have process definitions of their own.
- Process definitions will change over time. These changes could be rapid, and could occur in the middle of a project iteration.
- Although data was collected using a particular process, it will often be necessary to analyze it according to the structure of a different process. For example:
    - If an individual uses a personal process to collect data, their team might want to roll up the data according to a corresponding team process.
    - Different teams may have varying processes, but an organizational process mentor might want to analyze all of this data through the eyes of an organizational process.
    - When a team changes their process, they may want to view historical data through the eyes of their current process definition.

Because of these challenges, separate mechanisms will be needed to record the definition of a process, and to provide mappings between two processes. At a minimum, the warehouse will need to track:

- The various processes that have been created
- The phases in those processes, and information about those phases
- A history of how these process definitions have changed over time
- Optional mappings showing the correspondence between phases in related process definitions

Of these, change history has been the most likely to present special challenges. After significant analysis and review from experts within the TSP community, the decision has been made to:

- Capture the evolving states of a process as unique entities, each of which could be considered to be an unique "defined process"
- Use the "process mapping" mechanism to map "Process State N" to "Process State N+1"

Based on this design decision, the mapping between states in an evolving process can utilize the same mapping mechanism that is used by:

- The mapping of team processes to organizational processes
- The mapping of tailored processes to standard processes
- The mapping of personal processes to team processes

The use of a single "process mapping" mechanism for all of these scenarios is powerful. It means that a report which can handle one scenario can handle them all. A thoughtfully designed report can take data collected from any process, and slice/dice it according to a target process, as long as mappings between these two processes have been defined.

Of course, it will be within the purview of TSP/TPI tools to manage the definition of processes and the mapping of process phases. Once these items have been captured in a TSP tool, the data warehouse will support the succinct analysis of data by both direct and mapped process phases.