# VALIDATING THE TAME RESOURCE DATA MODEL*

D. Ross Jeffery (1) & Victor R. Basili (2)

(1) University of New South Wales, Australia
(2) University of Maryland, College Park, MD 20742

## Abstract

This paper presents a conceptual model of software development resource data and validates the model by reference to the published literature on necessary resource data for development support environments. The conceptual model presented here was developed using a top-down strategy. A resource data model is a prerequisite to the development of integrated project support environments which aim to assist in the processes of resource estimation, evaluation and control. The model proposed is a four dimensional view of resources which can be used for resource estimation, utilization, and review. The model is validated by reference to three publications on resource databases, and the implications of the model arising out of these comparisons is discussed.

**Keywords** : software process, methods, tools, conceptual model, resources, estimation, environments, software engineering database, validation

## INTRODUCTION

To date, the approach taken to the accumulation of knowledge concerning the software process has been largely bottom-up. Studies have been carried out to determine the existence and nature of project relationships. These studies, such as [Wolverton 74], [Nelson 67], [Chrysler 78], [Sackman et.al. 68], [Basili, Panlilio-Yap 85], [Basili, Freburger 81], [Basili, Selby, Phillips 83], [Walston, Felix 77]), [Jeffery 87a,87b], and [Jeffery, Lawrence 1979, 1985] have explored the relationships between project variables, searching for an understanding of the software process and product. For example, relationships between effort and size, errors and methods, and test strategy and bug identification, have been found.

*This research was funded in part by NASA Grant NSG-5123 to University of Maryland

This paper has two major aims:

1) To briefly present a top-down characterization (TDC) structure of software project resource data, which aims to facilitate :

1. Further accumulation of knowledge of project resource characteristics and metrics within a theoretical structure.

2. The storage of project resource data in a generalized structured way so that estimation, evaluation, and control can be facilitated using an organized quantitative and qualitative data base.

2) To validate this structure against published resource data models.

The characterization structure of resource data is a prerequisite to the development of an Integrated Project Support Environment (IPSE) in which it is possible to:

1. Objectively choose appropriate software processes.

2. Estimate the process characteristics such as time, cost, and quality

3. Evaluate the extent to which the resource aims are being met during development, and

4. Improve the software process and product.

The structure presented and validated here is a part of the TAME (Tailoring A Measurement Environment) project which seeks to develop an integrated software project measurement, analysis, and evaluation environment. This environment is based in part on the evolutionary improvement paradigm [discussed in Basili, Rombach 87]. It is also based on the "Goal-Question-Metric" paradigm outlined in [Basili 85] and [Basili, Weiss 84].

The aims of this paper are firstly to present the TDC structure or model for the perception of software development resources which will assist in the process of taking those aims of, say, a development manager and translating them into a set of questions and metrics which can be used to measure the software process. It is meant to be independent of the particular process model used



0270-5257/88/0000/0187$01.00 © 1988 IEEE

Recommended by: B. Boehm and J. Musa

for development and maintenance. A full description of the model, including its dynamic nature is described in [Jeffery, Basili 87a and 87b]. The paper secondly aims to validate the model by a comparison of the model with the resource data models presented in the literature.

## 2. THE PROJECT ENVIRONMENT CHARACTERISTICS

Resources are consumed during the software process in order to deliver a software product. The software process has overall characteristics which are super-ordinate to the resources consumed. Therefore, before resource data can be characterized it is necessary that a process characterization profile be established. This characterization includes data on factors such as:

project type
organizational development conventions
project manager preferences
target computer system
development computer system
project schedules or milestones
project deliverables

In this data the broad project and its environment characteristics are established. For example, is the process using evolutionary development or a waterfall method? Is the project to be developed by in-house staff or external contractors? What organizational constraints are being imposed on the project development time? What management constraints are being imposed, say on staffing levels?

These factors form the environment in which the software process must occur, and will therefore determine, in many ways, the nature of that software process. A simple example of this is the question of the process model - evolutionary or waterfall. This constraint establishes milestones and the pattern of resource use, and therefore partially determines the interpretation of the resource data collected.

## 3. THE RESOURCE CLASSIFICATION

At the level below the characterization of the project and its environment we are interested in classifying the resources consumed in the generation of the software product. In this section of the paper we present a structure for that classification. This structure covers only the resource aspect of the project and is therefore only concerned with the software process and the resources consumed or used in the process. The model is not concerned with the software product. As stated above, the resource model was first developed and presented in [Jeffery, Basili 87]

The model structure consists of a four dimensional view. This four dimensional view is divided into two segments:

1. resource type, and
2. resource use

In a software process the two segments being separated are (1) the nature and characteristics of the resource, and (2) the manner in which we look at or consider the consumption of that resource.

### 3.1 Resource Type

In the first segment we are concerned with classifying the nature of the resource; is it someone's time, or a physical object such as a computer, or a logical object such as a piece of software? We are also interested in describing the properties of those resources such as description, model number, and cost per unit of consumption.

By decomposing the resources into different types different views of the resources can be provided. For example, it may be important for operations personnel to know a breakdown of the hardware resources used on a project according to the different physical machines being used, whereas from a project manager's perspective at a point in time, the specific machine may not be of interest, but the availability of a certain class of machine may be critical. Resource managers will be interested in the types of resources available (for example, people) and the characteristics of those resources for project planning purposes. Thus the categorization provided here is the basis of the resource management environment, in that it is in this segment of the model that the resources are listed and described.

The resources of a software project can be classified as:

- hardware
- software
- human
- support (supplies, materials, communications facility costs, etc.)

These categories are meant to be mutually exclusive and exhaustive and therefore are able to contain each instance of resource data in one or other of the categories.

Hardware resources encompass all equipment used or potentially able to be used in the environment under consideration. (For example, target and development machines, terminals, workstations).

Software resources encompass all previously existing programs and software systems used or potentially able to be used in the environment under consideration. (For example, compilers, operating systems, utility routines, previously existing application software).

Human resources encompass all the people used or potentially able to be used for development, operations, and maintenance in the environment under consideration whether internal or external (subcontractors, consultants, etc)

Support resources encompass all of the additional facilities such as materials, communications, and supplies which are used or potentially able to be used in the environment under consideration.

The values associated with these resources may be stored in both price and volume measures, where volume means, for example, hours of use or availability, or the number of times a resource is needed, and price refers to the $ values associated with that resource. This may be a cost per unit measure or a cost per period of time.

This four-way classification provides an initial resource-type decomposition. The aim in this decomposition is to separate the major resource elements that are used in the software process in order to provide manageability. This initial separation is necessary because of the very different nature of each of these resource types and the consequent difference in attributes and management techniques which are necessary in the estimation, evaluation, and control of each of these resource categories.

Further decomposition within this segment may be desirable and will be dependent on the goals of the responsible persons. The number of different possibilities increase as the decomposition continues within each of the major resource categories. For example, the exact nature of the resource decomposition within the hardware category will vary significantly from one organization to another because of the different hardware utilized and the organizational structure surrounding that hardware utilization. For example, it may be desirable to decompose hardware into target and development hardware if there is a difference, and software into operating systems and languages/editors in order to model say the availability of cross-compilers.

### 3.2 Resource Use

Over the type segment we need to impose the second segment; the "use" structure. The categorization within this dimension allows the resources consumption to be associated with different perspectives of the software process. For example, it is through this use structure that we are able to distinguish, for example,

between prior-project expectations of consumption and resources actually consumed, or

between resources consumed in each phase of the project, or

between the utilization of a resource and the availability of that resource, or

between an ideal view of resource planning and the resources actually available.

The use structure consists of :

1. INCURRENCE
   - 1.1 Estimated
   - 1.2 Actual

2. AVAILABILITY
   - 2.1 Desirable
   - 2.2 Accessible
   - 2.3 Utilized

3. USE DESCRIPTORS
   - 3.1 Work type
   - 3.2 Point in Time
   - 3.3 Resources Utilized

#### 3.2.1 Incurrence

This category allows the resource information to be gathered and used in a manner suitable to the management of the resource. It is necessary, for example, to store data on *estimated* resource usage, resource requirements, and resource availability.

This data is necessarily kept separate from the *actual* resource incurrence or use, which is stored via the actual category.

These two categories then permit process tracking via comparisons between them and extrapolation from the actual data. At the project summary points, explanations and defined data accumulations on estimated and actual resource use provide feedback on the process. This *feedback* should contain reasons for variance between the estimated and actual so that a facility for corporate memory can be established and the necessary data stored to facilitate and explain any updates of the current resource values. It needs to be noted that the model proposed allows for different estimates and actuals at different points in time.

The two classifications are the basis for the structure proposed because they constitute significantly different viewpoints on the process, and again provide mutually exclusive categorization which will facilitate management estimation, evaluation, and control.

This structure requires that process data, as it changes in value during the project, will not be lost but will be stored in an accessible manner so that meaningful analysis of projects can be carried out using a database that provides complete details of the project history.

This philosophy specifically addresses the need for a corporate memory concerning past projects. By implementing such a structured project log the basic data for such a memory is available in numeric and text format.

### 3.2.2 Availability

This category allows storage of a resource use by :

- desirable
- accessible
- utilized

This categorization provides further refinement of the resource data. Through this, and say the incurrence category, it is possible to compare the actual resources utilized with the estimated utilization, and then trace possible reasons for variance through the desirable and accessible dimensions. That is, differences between planned availability and actual availability of a resource will be significant in understanding the software resource utilization that occurred during the process.

Desirable is defined as all the resources that are reasonably expected to be of value on the project.

Accessible is a subset of desirable (when considering the project resources only) and is used to define the resources which are able to be used on the project.

The difference between desirable and accessible is those resources seen as desirable for the project but which were not available for use during the project. This difference may occur, for example, because of budget constraints or inability to recruit staff. The desirable resource list permits an "ideal" planning view. When compared with accessible it allows management to see the compromises that were made in establishing the project, thus facilitating a very explicit basis for risk management within the resource database. The database is thereby able to hold views of not only the resources actually applied to the project but also those resources which were considered to be desirable along with the reasons for their use or non-use. In this way the resource trade-offs are made explicit.

Utilized is a subset of accessible and is defined as the resources which are used in a project.

The difference between accessible and utilized represents those resources available for the project but not used. This difference will arise because of three possible reasons:

1. The resources prove to be inappropriate for the project under consideration, or

2. The resources are appropriate but they are excess to those needed

3. The resources are appropriate, and their use is contingent on an uncertain future event.

The use of these storage categories is somewhat complex and is explained in detail further below in section 3.4.2.

Through this availability category we are able to distinguish between:

(1) the resources which are reasonably expected to be beneficial to the process (desirable),

(2) the resources which exist in the organization and are able to be used if needed (accessible), and

(3) the resources which are used in a project (utilized)

Through this categorization it is then possible to track resource usage and to pinpoint their use or non-use and to ascribe reasons particularly to their non-use as in the case of non-accessibility. As in the INCURRENCE category, the reasons for divergence between desirable, accessible, and utilized are stored in a *feedback* facility.

### 3.2.3 Use Descriptors

This category provides a description of the consumption of the resource item in terms of three essential characteristics of the consumption that item:

1. *The Nature of the Work* being done by the resource: (e.g. coding, inspecting, or designing) This category can be used in conjunction with other views to distinguish between process activities, such as human resources estimated to be desirable in *design* work, or machine resources actually utilized in *testing* , or elapsed time implications of *inspections*.

2. *Point in Calendar Time* : This category pinpoints the resource item by calendar time. In this way resource items (estimated or actual; desirable, accessible, or utilized) are associated with a specific point in time or period of time. This facilitates tracing of time dependent relationships and the comparison of resource values over time.

3. *Resources Utilized* : This category measures the extent of resource consumption in terms of hours, dollars, units, or whatever is the appropriate measure of use.

The Use Descriptors also provide the link to the work breakdown structure which is commonly embodied in process models. This link is established through the association of a particular piece of work being done at a point in time with the work package described in the work breakdown structure. This point is discussed further below in Section 8, Validating the Model.

### 3.3 COMBINING THE VIEWS

The structure suggested here can be viewed as a hierarchy for the purpose of explanation. Such a hierarchy is shown in Figure 1.

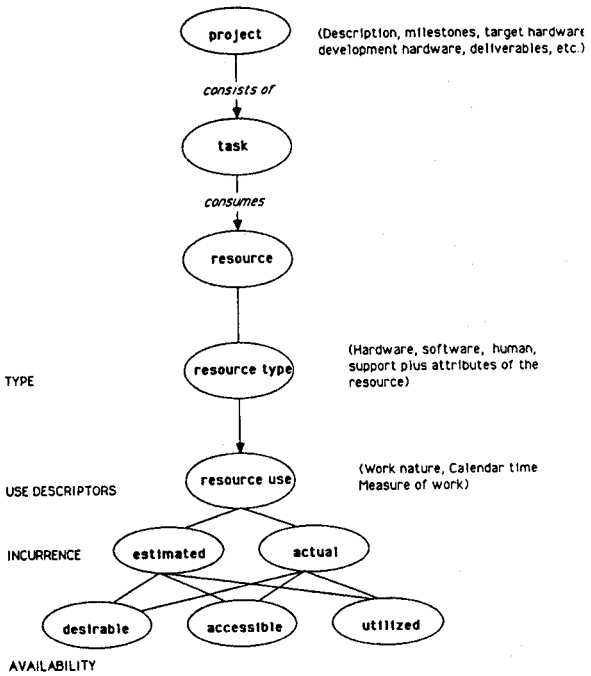


FIGURE I. THE STRUCTURE OF THE TDC MODEL

In this figure we see that the proposed structure views the software project (which has attributes describing that project) consuming resources. The resources are characterized as having four dimensions of interest (type, use, incurrence, and availability). At the resource type level we describe each resource as being one of hardware, software, human, or support, and having various attributes. The attributes for each of these four types will be different in nature. For example, the human attributes might include name, address, organizational unit, skills, pay rate, unit cost, age, and so forth. The attributes for hardware will be quite different, describing manufacturer, purchase date, memory capacity, network connections, or similar types of characteristics.

At the next level in the diagram we model the use of the resource. In the first instance this involves the type of work that the resource is performing, the point (or span) in calendar time at which the work is being done, and the measure of the amount of work done. This last measure (amount of work) might be expressed in person-time, execution-time, connect-time, or whatever is the relevant measure of work for the resource instance.

The use of the resource is then described as being either estimated or actual, and both of these may be desirable, accessible, or utilized. In this way the following concepts are supported :

1. *Estimated Desirable* : The resources considered "ideal" at various stages of the planning process.

2. *Estimated Accessible* : The resources which are expected to be available for use in the process, given the constraints imposed on the software process (a contingency plan).

3. *Estimated Utilized* : The resources which it is anticipated will be used in the software process.

4. *Actual Desirable* : With hindsight, the resources which proved to be the "ideal" considering the events that occurred in the software process. A part of the learning process.

5. *Actual Accessible* : Again with hindsight, the resources which were actually available and could have been utilized. A part of the learning process.

6. *Actual Utilized* : The resources actually used in the software process.

Categories one through three are used initially for planning purposes. The numeric and text values associated with each of these three categories may be derived from:

a. individual or group knowledge
b. a knowledge base
c. a database of prior projects, and/or
d. algorithmic models

At the very simplest level, the planning process might establish only numeric values in the estimated utilized category based on individual knowledge alone. In essence, this is the only form of estimation used in many organizations, wherein project schedules and budgets are established by an individual, based on that individuals experience. These estimates represent the expected project and resource characteristics for the duration of the project.

The extensions suggested here allow these estimates to be enlarged in the following dimensions :

The nature of the estimate
The source of the estimates
The timing of the estimates

1. *The nature of the estimate.* The model allows project and resource managers to distinguish between desirable, accessible, and utilized estimates as discussed above. The estimated desirable dimension would be used at a fairly high level in the project planning process to outline the hardware, software, people, and support resources that are considered to be desirable for the project. This may list specific pieces of hardware and software which are desirable at certain points in time. It might also be used to list characteristics of the people (such as skills) that would be ideal on the project. The accessible dimension would

then reflect the expected resources that will actually be available to be used. Again this could be at a fairly high level, indicating the resources available, the differences between these and those desirable, and the reasons why the two categories do not agree; reflecting cost constraints, or risk attitudes which have been adopted as part of the project management profile. The utilized category would normally extend to a lower level in terms of the project plan, detailing estimated resources perhaps down to the work package level and short periods of time.

2. *The source of the estimates.* It was suggested above that there are four major possible sources for these estimates; individuals or groups of people, a knowledge base, a database of prior projects, and algorithmic models of the process. Each of these should be supported in a measurement environment, and each has significant implications with respect to the design of such an environment. The current state of the art appears well equipped to support algorithmic models of some parts of the estimation process (for example, estimates of project effort based on one of the many available estimation packages such as COCOMO [Boehm 81], SLIM [Putnam 81], SPQR [Jones 86]). Similarly the tools available in the database environment allow the storage and retrieval of numeric data on past projects. However the storage and searching of large volumes of text data on prior projects, the use of a knowledge base, and the support of group decision support processes are all the subject of current research (see for example, [Bernstein 87], [Nunamaker, et.al. 86], [Barstow 87], [Valett 87]).

*The timing of the estimates.* In the structure suggested, all estimates may be made before the commencement of the software process and also at any point in time during the process. However there are certain points in time during the process at which estimates are more likely to be updated. These are:

1. at project milestones

2. at manager initiated points in time at which major divergence between estimate and actual is recognized by the manager

3. at system initiated points in time at which the measurement system recognizes a potentially significant divergence between estimate and actual

The third possibility implies that the measurement system is able to intelligently recognize the existence of a problem with respect to the comparison of actual and estimate. This facility is suggested as needed because one of the major management stumbling blocks is generally not concerned with taking action once a problem is identified, but the identification of the problem in the first place. This identification problem occurs because of the volume of data that needs to be processed in order to recognize a potential problem state. It is the measurement environment which is expert at processing the data volume. It is the manager who is expert at taking corrective action once the problem is highlighted.

Categories four (actual desirable) and five (actual accessible) of the structure exist to provide a feedback and learning dimension to the project database. These values would be determined after the project is complete. And in the comparison of the estimates made at various stages of the process and these two categories, a process is facilitated in which the organization can learn based on the variance of expectations and actual which have occurred in the past projects. As with the estimates, the categories of desirable and accessible are used in order to allow the comparison of "actual ideal" with "actual available" so that an ex-post view of the management of the process can be captured. The question being asked here is; "How could we have handled resources better?" It is a learning mechanism to generate explicit new knowledge for the knowledge and data bases, and also to improve individual and group knowledge.

Category six (actual utilized) will be the most active category within the structure, carrying all of the values associated with the resources of the project. These values will be updated on a regular basis throughout the software process, and will be the source of the triggering process mentioned in the discussion of updates to the estimates.

The data collected during the project should be able to:

1. increase individual and group knowledge
2. improve the knowledge base
3. add to the prior project database, and/or
4. support the algorithm determination process in the individual organization.

In summary, the model proposed is a four dimensional view of resource data. The four views in the data model are:

1. RESOURCE TYPE: which is a mutually exclusive and exhaustive categorization which captures the nature of the resource.

2. INCURRENCE: which is also mutually exclusive and exhaustive describing actual or estimated resources. It carries an additional feedback element to contain the corporate memory explaining the difference between the category values and differences over time.

3. AVAILABILITY: in which each category is a subset of the the higher category, allowing desirable, accessible, and utilized resources. Again feedback is used to explain the differences between categories and over time.

4. USE DESCRIPTORS: which categorizes specific elements in the nature of the resource use. These are the nature of the work done by the resource, the point in time of the work, and the amount of that work.

## 3.4 USING THE TDC STRUCTURE

### 3.4.1 At the project level

Discussion so far has applied the proposed 4D structure to resource classification. It is appropriate to also consider using this structure, or a part of it, for the Project Environment Characteristics outlined in section 2 above. In this way the constraints acting on the software process can be identified as applying:

to a particular type of resource,
either estimated or actual
with a stated availability
at a point in time,
concerning a particular type of work

An overall model of the software project is shown in Figure 2. In this figure the meta-entity project is decomposed into a number of tasks or contracts, each task consuming the meta-entity resource and producing the meta-entity product. In the implementation of this model the meta-entities will require many entities to characterize them.

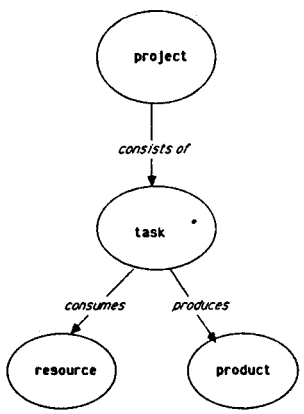


FIGURE 2. AN OVERVIEW OF THE SOFTWARE PROJECT

Thus the project has characteristics, as do the tasks and subtasks, the resources, and the products. Characteristics at all of these levels need to be stored.

Through the storage of the project characteristics, the constraints acting on the product or process determined at any time before or during the project can be tracked for consistency, and any changes noted to facilitate a relationship analysis between the project and the resource occurrence values accumulated during the process.

A simple example of the application of this structure would be where the process organization is changed during the development, say a change toward greater user involvement. This change would be reflected in a difference between the estimated project characteristic and those at the point in time at which the change occurred. This information is then used to explain variances that occur in the process data, such as a changed pattern in staff utilization.

Examples of the data stored at the project level would include:

- the type of project
  e.g.real time, business application
- the project elapsed time
- the total project effort
- the total project cost
- the type of development process
  e.g. evolutionary
- the target computer
- the development computer
- the project deliverables
- the project milestones
- the project risk profile

The application of the TDC model at this level provides a mechanism for storing estimates, accumulating actual values, and facilitating feedback and learning at the level of the project and its development environment.

If we take the project milestones as an example and assume that the milestones apply equally to all resource types, then the model suggests we store:

- *estimated desirable milestones.* This is an "ideal world" view of the project milestones; the dates at which we could deliver if we were not constrained.

- *estimated accessible milestones.* Given the constraints we will be working under, these are the dates at which we could deliver if it were necessary.

- *estimated utilized milestones.* These are the dates at which we expect to deliver, taking into account the dimensions of desirable and accessible.

These three views, in their values and difference, provide a perspective on the risk associated with the project; the smaller the difference between the categories, the higher the risk. More specifically, the difference between estimated desirable and estimated accessible shows the extent to which elapsed time could be changed if the constraints could be modified. For example, if the estimated final desirable milestone were June 30th and the

estimated final accessible milestone was August 30th, the difference of two months measures the estimate of the extent to which the project could be compressed if the restricting constraints could be be removed.

The difference between the estimated accessible and the estimated utilized provides a measure of the available slack in the milestones. This difference is the extent to which the milestones could be compressed, without modifying the project constraints. In the example above, the estimated utilized final milestone might be say November 30th. In this case the difference between accessible and utilized of three months reveals the amount of elapsed time compression that is possible on this project without changing constraints.

In these relationships we see some of the dynamic nature of the project characteristics. This suggests that for the TAME measurement environment, if a change in project characteristics such as the nature of the process occurs, then this should trigger the review of the project milestone and effort values, which will also be reflected at the lower level in the task and resource data values.

In the actual category we need to store the :

- *actual desirable milestones.* As explained above, this category is used for feedback and learning. It carries the values calculated after project completion based on the knowledge gained about the project during its completion. This value is again an "ideal world" value.

- *actual accessible milestones.* This is also a feedback and learning category which says, based on the constraints which did eventuate in the process what milestones could have been achieved?

- *actual utilized milestones.* This category stores the dates of the milestones achieved. Differences between actual and estimated are stored in a feedback facility to provide a mechanism for learning and a mechanism for calculating the actual desirable and accessible at project end.

### 3.4.2 At the resource level

The description of the use of the TDC structure at the resource level amounts to a process model of resource planning and use in software development. This process can be described as an interacting three-stage process involving the subprocesses of:

1. planning
2. actualization
3. review

The *planning* process establishes and records the resource expectations or estimates before and during the software project, and the *actualization* process tracks and records the actual use of resources during the software project. The *review* process compares actuals with estimates for the purposes of modifying the estimates and learning from experience. In this way the *feedback* referred to above provides information for an historic resource database for future planning and estimation. Details of this process model are given in [Jeffery, Basili 87].

### Application of the planning and review cycles

In any particular organization, it may be deemed sufficient to use only a part of the planning and review processes outlined here, and therefore only a part of the TDC structure presented in this paper.

For example organizations may not wish to use project reviews, or they may not consider it appropriate to carry out formal contingency planning or risk management. At the simplest level only the estimated utilized and the actual utilized may be used, perhaps providing input to an informal project learning process which occurs at the individual level.

Specifically, it is most likely that in software environments with very little uncertainty (say an implementation of the twentieth slightly different version of a well known system) there may be no need to explicitly consider the desirable or even accessible dimensions of the resource model. If uncertainty is very low, the utilized level of the model may capture all the necessary data. The advantage of the model in this case is that the data excluded is done so in the knowledge that there is no information in those levels not used.

In higher uncertainty environments, the model prompts the estimator to think explicitly of the resource risks and uncertainty of the development process, and to quantify or express that risk as a part of the resource database.

## 4. VALIDATING THE MODEL

Three significant pieces of work in the literature which provide definitions of the types of data needed to support the measurement of the software process are [Penedo, Stuckle 85], [Tausworthe 79], and [Data & Analysis Center for Software 84, STARS Measurement DID Review].

Penedo and Stuckle (P&S) provide an excellent structure and content of a project database for software engineering environments which can be used here to test whether the model resulting from the top-down methodology employed is able to encapsulate all of the process data suggested by them as needed in a project database. Table 1 lists the entities identified by Penedo and Stuckle and associates the particular model categories which would be used in the model derived here to describe them.

The first aspect which is noticed when mapping the 31 P&S entity types to the TDC model is that the broad structure presented in section 2 above (The Project Environment Characteristics) is an important link between the software process and product. The P&S list contains entities for the project, task, product, and resource categories of Figure 2. In table 1 the P&S entities such as the requirement and risk have been categorized as project characteristics, while entities such as data component, external component, document, interface, product description, product, and software component have been categorized as product instances.

But the focus of this paper is not on the project or the tasks which go together to make up that project. Rather the focus is the resources consumed by those tasks. In this respect we notice that only a subset of the available TDC categories are used in the P&S entities. For example, at the Resource Type level we see instances of all four categories (Hardware, Software, Human, and Support), but at the next level it appears that the P&S model concentrates on actual values. It is difficult to see how the P&S model stores values for estimates, and particularly how the information explaining divergence between estimate and actual can be stored. The same applies to the Availability level of the TDC structure. The P&S model appears to concentrate on the Utilized aspect and does not appear to model the other availability dimensions presented in the TDC structure. This may well be because these dimensions of resource data were considered not to be necessary in the environment of the P&S study.

Table 1. P&S Database Entities in The Model Structure

| Penedo & Stuckle Entities | Top Down Model Categories |
|---|---|
| Accountable Task and Contract | The task and contract are the convergence of process and product and subsets of the project. It is in a contract or task that resources are consumed to produce the product. They are not, therefore, resource entities. |
| Change Item | This item is generally associated with a product change. |
| Consumable Purchase | *Support resource, incurrence and availability not specified. |
| Data Component | Product Entity |
| Dictionary | *Software resource, or perhaps product entity |
| Document | Product Entity |
| Equipment Purchase | *Hardware resource |
| External Component | *Hardware resource or perhaps Product Entity |
| Hardware Architecture | *Hardware resource or perhaps product entity |
| Hardware Component | *Hardware resource or product entity |
| Interface | Product Entity |
| Milestone | *Project Entity |
| Operational Scenario | Product Entity |
| Person | *Human Resource |
| Problem Report | *Process as part of feedback or Product entity |
| Product | Product Entity |
| Product Description | Product Entity |
| Requirement | Project Entity |
| Resource | *Support resource |
| Risk | *Project Entity |
| Simulation | Product entity |
| Software Component | Product Entity |
| Software Configuration | Product Entity |
| Software Executable Task | Product Entity |
| Software Purchase | *Software resource |
| Test Case | *Software resource and/or product entity |
| Test Procedure | *Task or project characteristic |
| Tool | *Software resource |
| WBS Element | Project Decomposition Entity, may be the same as accountable task and contract |

It remains to be seen, of course, whether all of the categories available in the TDC structure are deemed necessary in any particular environment. However, the advantage of such a structure is that exclusion of certain categories of data occurs explicitly rather than implicitly.

The second model suggested as a means of testing the TDC model is that provided by [Tausworthe 79]. In this work the model's entities are not presented in a list form, but are included in text discussion and report forms. For this reason it has been necessary to convert the form to a list of entities. In doing so it is always possible that misconceptions of Tausworthe's ideas may be present. However, even if incomplete, it provides another test of the suitability of the TDC model.

The Tausworthe structure is very much oriented towards a decomposition of the project into tasks and the association of resources with those tasks. Thus the modelling approach used by Tausworthe is somewhat at a tangent to the modelling approach used here since once again our focus is on resources, not the activities which consume those resources. This is not to say, however, that it is not necessary to associate resources with tasks, but that it may be necessary to model resources apart from the tasks that consume them in order to better understand all of the dimensions of resource data.

The entities listed here are a partial list derived from the work breakdown structure, the software technical progress report, the software change analysis report, and the software change order of Tausworthe's model. From these sources the following resource data, among others, were identified as necessary to establish a resource database. Only some of the Tausworthe entities have been listed here. This has been done to the extent that is necessary to illustrate the conclusions drawn.

From Table 2 it is clear that the focus of attention in the Tausworthe work is the project and the decomposition of that project into its component parts. Thus we see that the resource data is associated with particular tasks and activities. In viewing the data in this way a structure is provided which is excellent for control purposes, in that it establishes units of accounting which are more easily estimated and controlled. What is not clear from the structure, however, is how questions of desired versus accessible resources can be modelled, nor exactly how actual versus estimated can be compared and conclusions stored for use in later project estimates. It is also difficult to see how the model proposed in the WBS can easily facilitate the analysis of resources consumed on a particular activity type (say inspections), regardless of the project phase in which the inspections were done or the project task in which they were done. Thus questions such as the value to the project of using a particular form of inspection may be difficult to answer because the data model may make this data difficult to isolate.

Table 2. Tausworthe Derived Entity List

| Tausworthe Entities | Top Down Model Categories |
|---|---|
| Staff:<br>Staff I.D.<br>Staff Name<br>Staff Phone | Human resource, estimated or actual |
| Task Activity:<br>Task I.D.<br>Task Activity I.D.<br>Budget $ | The dollar value may be a sum of all resources consumed on a task-activity, estimated or actual |
| Task:<br>Task I.D.<br>Task Name<br>Task Descr<br>Task M'ger<br>Task Budget $, ETC. | The value is a sum of all resources, estimated and/or actual |
| Software Change Order<br>S/ware ID<br>Change Order #<br>Activity ID<br>Person ID<br>Description<br>Start Date, etc. | The focus is again on the activity. The resources may be any type, estimated or actual. |

However, it is clear that the resource data suggested as necessary by Tausworthe are readily modelled in the TDC structure. The importance of the application of the TDC model to the project and task level is highlighted by Tausworthe and also Penedo & Stuckle, so that the association of resource data and project work breakdown structures can be facilitated.

Perhaps the most detailed resource data collection forms developed so far has been that of the STARS Measurement Data Item Descriptions.

The information which follows in Table 3 was derived from stars Software Development Environment Summary Reports DI-E-SWDESUM, DI-F-RESUM, DI-F-REDET, [06 JULY 1984]. These reports contained information most relevant to the task of validation of the TDC model. The data suggested as necessary by these reports concerned aspects of the project, the process, and the product. In this paper only those aspects concerning the project and the process have been listed. As with the Penedo and the Tausworthe models, the data model implied in the work appears not to have been developed on the basis of a theoretical structure, but rather from a pragmatic evaluation of those data items deemed necessary for project management. In addition, because the data items are
listed in the context of data capture forms, some rearrangement of these items has been carried out in the following data list in order to provide a clearer presentation of these items.

TABLE 3. STARS Measurement Data Items Descriptions

**A. PROJECT NAME**

Project Name
Contractor
Contract No.
Start date, Finish Date
Software Level (System, Subsystem, CSCI)
Application Type
Application description
Revision of current project (y/n)
  Revision -version no.
    % of software redeveloped
    Total no. lines of source code
Initial development (y/n)
  if y - Total no. lines source code
    no. of instructions
    no. of data words
System Structure-
  single overlay
  multiple overlay
    (# overlays, avg. size bytes
  independent subsystems
    (# subs, avg.size bytes
  virtual memory system
    (amount of addressable memory, size bytes
Progamming language and % used
Constraints -
  Execution Time, rating
  Main memory size, rating
  Product Complexity, rating
  Database size, rating
  Methodology, rating
  required reliability, rating
  Other, rating
Concurrent Hardware development (y/n)
Operational site development (y/n)
Multiple site development (y/n)
no. of development sites
no. of test sites (if different)
Other Constraints .(text).
cost estimation assumptions made
cost estimation methods used and supporting rationale
rationale for discrepencies between current estimates and all previous estimates

**B. SITE CONFIGURATION INFORMATION**

Site ID
Description (development, test)
Computer manufacturer
Model name
Model no.
no. of persons accessing site
no. of input terminals
Terminals in each programmers office (y/n)
Input terminals in central area (y/n)
no. of card readers
no. of printers
no. tape drives
no. disk drives
other peripherals.(specify).
no. documentation sets on hardware/software environment available
no. site support personnel
amount of storage in development computer
  main memory real
  main memory virtual
  aux memory

DEVELOPMENT SITE ACCESS

Site I.D.
Access type: % batch
% interactive
Average job turnaround time
no. hours per day development site available
no. days per week development site available
no. hours per day utilized
no. days per week utilized

TEST SITE ACCESS

Site I.D.
no. hours per day test site available
no. days per week test site available
no. hours per day test site utilized
no. days per week test site utilized

## C. PROJECT PHASE INFORMATION
[examples]

**requirements**

Development system used (y/n)
Documents maintained on the dev. system (y/n)
Methodology (formal spec., functional spec., procedural spec., english spec., none, other)
Tools/Formalisms (requirements analyzer, word processor, on-line editor, c.m.t., librarian, spec lang, PDL, none, other)
start and finish date
deliverables

**design**

Development system used (y/n)
Documents developed/maintained on system (y/n)
Methodology (top down, bottom up, hardest first, prototyping, iterative enhancement, none, other)
Tools/Formalisms ( software dev. folders, design reviews, walkthru's, flow charts, HIPO, etc.)
start and finish date
deliverables

**implementation**

Development system used (y/n)
Documents maintained on development system (y/n)
Unit testing performed on dev. system (y/n)
Methodology (top down, cpt, prototyping, etc.)
Tools/Formalisms ( code reading, pre-compiler, dbms, etc)
start and finish date
deliverables

**test and integration**

Testing performed on development system (y/n)
Documents maintained on system (y/n)
Level of testing performed on dev system
Methodology (spec driven, top down, none, etc)
Tools/Formalisms (.....)
start and finish date
deliverables

## D. PROJECT PERSONNEL INFORMATION

[these values can be derived from more detailed records]

Project Name
Job Classification (supervisor, consultant, analyst, programmer, site operator, librarian, other)
Avg. no. years application experience
Avg. no. years experience with software
Avg. no. yrs software training
Avg. no. yrs programming language experience
Avg. no. yrs hardware experience
Avg. capability rating

**communication**

Regular project status meetings (y/n)
How often?
Persons typically in attendance (classification, No.)

## E. RESOURCE EXPENDITURE ATTRIBUTES

**summary level**
[these values may be derived]

Project name
total system cost, estimated, actual
total software cost, estimated, actual
total labour cost $, estimated, actual
total software labour cost $, estimated, actual
total labour hours, estimated, actual
total software labour hours, estimated, actual
total staff size, start, finish, estimated, actual
total software staff size, start, finish, estimated, actual
total computer costs $, estimated, actual
total software computer costs $, estimated, actual
total computer hours, estimated, actual
total travel costs $
total material costs $
total miscellaneous costs $

[these may be divided by milestones or activities]

**labour costs**
[these values may be derived]

labour category id
total hours
no. of people, start, finish
cost $
computer hours
computer costs $

**computer costs**
[these values may be derived]

no. of computers used
no. of different types of computers
total computer hours

*** for each computer***

computer i.d.
number of hours
total computer costs $
cost of each computer $

**task costs**
[these values may be derived]

task i.d.
definition
personnel costs
software costs
hardware costs
supplies costs

****for each task****
****for each labour category****

total hours
no. of people, start - finish
cost $
computer hours
computer cost $
travel cost $

****for each task****

total cost of labour
total hours of labour
total cost of computer
total hours of computer
total cost of travel
total cost of materials
total cost of miscellaneous

The Table provides data items to describe the project, development and test site configurations and access, project phases, personnel assigned to a project, and resource expenditure summaries. The detail shown here has been selected to highlight the volume of data items which will be necessary in a measurement system.

In terms of the TDC model, the STARS list shows recognition of the need to store resource availability in that the development and test site access data includes an accessible and a utilized dimension. There appears, however, to be no facility for storing the desirable dimension suggested in the TDC model. The STARS list also shows extensive use of the incurrence dimension in section E - Resource Expenditure Attributes, wherein estimated and actual resource use is tracked. The USE DESCRIPTORS of work type, point in time, and resource utilized are also extensively used in the STARS list. It is not possible from the documentation, however, to determine the reasons that the availability dimension was not applied more extensively in the data model (for example accessability of personnel or specific hardware or software items are not modelled). It can be assumed that it was considered to be innappropriate for entities other than site access.

The STARS data list provides considerable support for the theoretical structure provided in the TDC model. It reveals a considered need for the storage of :

1. Project information
2. Resource type information
3. Incurrence information
4. Availability information and
5. Use descriptors

Of considerable significance is the fact that none of the three schemas considered here have suggested data entities or items which cannot be successfully modelled using the TDC structure. It appears that the schemas considered here may be incomplete when compared with the TDC structure, but the reasons for the apparent exclusion of data entities and items are not known, but may be based on purely pragmatic reasons.

## 5. CONCLUSIONS AND IMPLICATIONS AT THE RESOURCE DATA LEVEL

The model presented here is meant to be general and provide a perspective for project manager and organization in identifying and tracking resources. It should help in better understanding the compromises made in resource allocation. However, it is assumed that any project (or even organization) will work with a subset of this model. For example, one might limit the number of availability views, such as combining desirable and accessible, or track only a subset of the resource categories. The subsetting process provides feedback on what has not been tracked. The actual data collected is driven by the goal/question/metric paradign based upon the goals set by the project and the organization.

The conclusions to be drawn from this research can be divided into two categories: those concerning the model itself, and those concerning the validation of that model.

In terms of the model itself, the discussion has suggested storage of resource data of a type which has significant storage and access implications; that of numeric and non-numeric project and resource data. It has been assumed in the discussion that the resource database is able to store not only numeric resource values, but also reasons for those values along with the resource environment characteristics.

A system using these suggestions should be able to efficiently search the numeric and non-numeric data in a manner which will eventually enable the system to propose reasons for numeric variances which occur in the database. In this way the system must be able to not only highlight a significant variance, say between an estimated and an actual resource occurrence value, but it should also be able to search the project characteristic database and the numeric and non-numeric resource classification database in order to propose or associate reasons for the variance.

It can be said that the model presented here has four broad implications :

1. It proposes a resource categorization which will allow project database designers to explicitly consider the content of that database against a model of the resource environment. In this way, a particular individual's view of the resource data can be positioned in a context and compared with other external views of the same data. This model should motivate the resource data user to consider the measures that may be beneficial in seeking improvement in the particular process goals.

2. It suggests a project management system's environment which will be able to achieve far more in terms of management support than any known environment available today. It is able to do this because of the extent and dynamic nature of the model of the resource data proposed.

3. It provides a resource categorization which can be used when considering relationships between tasks or contracts and resources. Specifically it provides a focus for the consideration of the resources consumed within a task.

4. It provides assistance when applying the Goal/Question/Metric process paradigm, so that questions which answer the resource purpose of the study are highlighted and the measures appropriate to those questions are suggested.

In terms of the validation of the data model we

have seen by reference to three published models that the proposed theoretical structure for resource data is able to encompass all that has been suggested as necessary for resource management. Also of significance, is the fact that each of the publications used contains different views of the necessary data and that each one omits certain elements that the other appears to consider of benefit. This is, of course, the norm in comparing different external views in a database design exercise. One advantage of the TDC model is that it is able to act as a data model template, suggesting the data categories which need to be considered when designing a resource data schema. If it is used in this way the data items excluded from the particular resource model instance will have been excluded on the grounds that they are deemed unneccesary in the particular environment, rather than being excluded because the category of data ( for example, estimated desirable hardware for testing) was not noticed by the data base designers as necessary.

Thus we can be confident that the theoretical model proposed in the TDC structure can contain all of the project and resource data so far suggested in the literature as necessary in a resource management environment. In addition it appears that there may be project and resource information of use in resource management which has not been included in prior models. The practical need for this additional information has not been justified in this piece of research but is the subject of other current work by the authors.

We have begun to apply the model independent of TAME in a couple of industrial environments and have found it provides a useful framework for planning and tracking resources throughout a project. We have not yet reached the stage where we have been able to evaluate the feedback process, however.

## 6. REFERENCES


[Barstow 87] D. Barstow, "Artificial Intelligence and Software Engineering," Proc. 9th Intn'l. Conf. on S'ware.Eng. IEEE, Monterey, April, 1987, pp.200-211.

[Basili 85] V.R.Basili, "Quantitative Evaluation of Software Engineering Methodology," Proc. First Pan Pacific Computer Conference, Melbourne, Australia, September, 1985.

[Basili, Freburger 81] V.R.Basili, K.Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," The Journal of Systems and Software, 2, 1981, pp. 47-57.

[Basili, Panlilio-Yap 85] V.R.Basili, N.M.Panlilio-Yap, "Finding Relationships Between Effort and Other Variables in the SEL," Proc. 9th COMPSAC Computer Software & Applications Conference, Chicago, October, 1985, pp. 221-228.

[Basili, Rombach 87] V.R.Basili, H.D.Rombach, "Tailoring the Software Process to Project Goals and Environments," Proc. 9th Intn'l. Conf. on S'ware Eng. Monterey, April, 1987, pp. 345-357.

[Basili, Selby, Phillips 83] V.R.Basili, R.Selby, T.Y.Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects," IEEE Trans. on Software Eng. Vol. SE-9 No.6, November, 1983, pp.652-663.

[Basili, Weiss 84] V.R.Basili, D.M.Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, SE10,3, November,1984, pp.728-738.

[Bernstein 87] P.A.Bernstein, "Database System Support for Software Engineering," Proc. 9th Intn'l. Conf. on S'ware. Eng., Monterey, April, 1987, pp. 166-178.

[Boehm 81] B.W.Boehm, Software Engineering Economics, Prentice-Hall Englewood Cliffs, New Jersey, 1981.

[Chrysler 78] E.Chrysler, "Some Basic Determinants of Computer Programming Productivity," Comm. of the ACM, 21,6, June, 1978, pp. 472-483.

[Data & Analysis Center for Software 84] STARS Measurement Data Item Descriptions, Data & Analysis Center for Software, RADC/COED, Griffiss AFB, NY. July, 1984.

[Jeffery 87a] D.R.Jeffery, "The Relationship between Team Size, Experience, and Attitudes and Software Development Productivity," Proc. COMPSAC87, Tokyo, October, 1987.

[Jeffery 87b] D.R.Jeffery, "A Software Development Productivity Model for MIS Environments," Jnl. of Systems And Software, June, 1987.

[Jeffery, Basili 87] D.R.Jeffery, V.R.Basili, "Characterizing Resource Data: A Model for Logical Association of Software Data," Technical Report TR-1848, University of Maryland, May 1987, 35pp.

[Jeffery, Lawrence 79] D.R.Jeffery, M.J.Lawrence, "An Inter-Organizational Comparison of Programming Productivity," Proc. 4th Intn'l Conf. on S'ware. Eng. Munich, 1979, pp.369-377.

[Jeffery, Lawrence 85] D.R.Jeffery, M.J.Lawrence, "Managing Programming Productivity," The Journal of Systems & Software, 5,1, February, 1985, pp. 49-58.

[Jones 86] T.C.Jones, SPQR/20 User Guide V1.1, Software Productivity Research Inc. January, 1986.

[Nelson 67] E.A.Nelson, "Management Handbook for the Estimation of Computer Programming Costs," System Development Corporation, Santa Monica, March, 1967.

[Nunamaker, Applegate, Konsynski 86] J.F.Nunamaker, L.M.Applegate, B.R.Konsynski, "Facilitating Group Creativity: Experience with a Group Decision Support System," Proc. 20th Annual Hawaii Intn'l. Conf. on System Sciences, Hawaii, January, 1987, pp.422-430.

[Penedo, Stuckle 85] M.H.Penedo, E.D.Stuckle, "PMDB - A Project Master Database for Software Engineering Environments," Proc. 8th Intn'l. Conf. on S'ware. Eng., London, August, 1985, pp. 150-157.

[Putnam 81] L.H.Putnam, "SLIM A Quantitative Tool for Software Cost and Schedule Estimation," Proc. NBS/IEEE/ACM Software Tool Fair, San Diego, CA, March, 1981, pp. 49-57.

[Sackman, Erikson, Grant 68] H.Sackman, W.J.Erikson, E.E.Grant, "Exploratory Experimental Studies Comparing Online and Offline Programming Performance Comm. of the ACM, 11,1, 1968, pp. 3-11.

[Tausworthe 79] R.C.Tausworthe, Standardized Development of Computer Software: Part II Standards, Prentice-Hall, Englewood Cliffs, New Jersey, 1979.

[Valett 87] J.D.Valett, "The Dynamic Management Information Tool (DYNAMITE): Analysis of the Prototype, Requirements and Operational Scenarios," M.Sc. Thesis University of Maryland, 1987.

[Walston, Felix 77] C.E.Walston, C.P.Felix, "A Method of Programming Measurement and Estimation," IBM Systems Journal, 16,1, 1977, pp.54-73.

[Wolverton 74] R.Wolverton, "The Cost of Developing Large Scale Software," IEEE Transactions on Computers, 23,6, 1974.