Technical Report TR-1848 May 1987
NSG-5123

CHARACTERIZING RESOURCE DATA: A MODEL FOR LOGICAL ASSOCIATION OF SOFTWARE DATA*

D.Ross Jeffery[1] & Victor R.Basili[2]

[1] Department of Information Systems
University of New South Wales
P.O.Box 1, Kensington NSW 2033
Australia
(02)697-4449

[2] Department of Computer Science
University of MAryland
College Park MD 20742
U.S.A
(301)454-2002

*This research was funded in part by the National Aeronautical and Space Administration Grant NSG-5123 to the University of Maryland.

ABSTRACT

This paper presents a conceptual model of software development resource data. A conceptual model, such as this, is a pre-requisite 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. A process model is presented showing the use of the data model, and instances of the goal, question, metric paradigm are presented to show the applicability of the models to the measurement task. The model is validated by reference to published literature on resource databases and the implications of the model in these database environments is discussed.

KEYWORDS: software process, methods, tools, conceptual model, process model, resources, estimation, environments database,

1. INTRODUCTION

We know too little about the software process. Our understanding of the relationships between inputs and outputs is fuzzy and at times erroneous. Evidence of the difficulties encountered in managing the software process abounds in the number of consulting organizations who aim to help management come to grips with the problems of software estimation, evaluation, and control.

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. 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]), 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. In this type of research we are trying to fit the pieces into a puzzle without a knowledge of the boundaries of that puzzle and with little direction as to the entire picture being represented.

Exploratory research, such as this, is the norm in a newer discipline where theories to direct the research are few and even the metrics of the discipline are undergoing development.

This paper provides a top-down characterization (TDC) structure of software project resource data, with the aim of facilitating:

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.

This characterization structure 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 developed 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 paper establishes a model of the resource estimation, utilization and review processes providing conceptual process and data models which are validated against existing data models.

2. THE MEASUREMENT ENVIRONMENT

One paradigm which can be adopted to improve the software process and resulting product is that of evolutionary development. This paradigm, which is commonly used

in software development itself, provides a low risk method by which one is able to make improvements to the object of interest based on the current characteristics of that object, the existing knowledge concerning those characteristics, and the goals appropriate to that object.

One project measurement environment which is based in part on this evolutionary improvement paradigm is TAME. This IPSE is discussed in [Basili, Rombach 87]. It is also based on the "Goal-Question-Metric" paradigm outlined in [Basili 85].

The thrust of the Basili argument is that to have a successful measurement environment it is necessary to connect that measurement with the goals which are being pursued in the development and which give rise to the purpose of data collection. There is little point in collecting data if there is no goal to be satisfied through that collection. Perhaps more importantly, without a goal it is not possible to establish the appropriate measures.

[Basili, Weiss 84] provide a sequence which can be used to establish the link between the goals and the metrics collected.

> 1.Generate a set of goals based upon the needs of the organization.
> 2. Derive a set of questions of interest or hypotheses which quantify the abstractions of the goals.
> 3. Develop a set of metrics and data distributions that provide the information to answer the questions.
> 4. Define a mechanism for collecting the data as accurately as possible.
> 5. Validate the data as it is collected.
> 6. Analyze the data to answer the questions.

To illustrate, we could measure project size using the metrics of lines of code or function points. Both are common metrics of size, but they can serve different purposes. If we knew that:

1. we could estimate function points sooner in the lifecycle than lines of code, and

2. the correlation between effort and function points was consistently higher than between effort and lines of code (see [Jeffery, Loo 87] for some evidence of this), and

3. our goal was to forecast the project effort based on estimated size,

then the function point metric would be better suited to our goal provided that we could estimate the function points as accurately as we could estimate the lines of code.

In another example, the goal might be:

> To reduce project effort by 3% over that forecast by the organizational estimation technique.

In order to achieve this goal actions must be taken in the development environment with the potential of achieving that goal, such as increasing staff skill, or applying a different development process. The GQM paradigm cannot select the method to be used, but it allows the change to be monitored through measurement so that the success of the changes in staff or process instigated can be determined, and then further future actions taken on the basis of the knowledge gained. The purpose of the measurement then would be:

To evaluate the process in order to understand and assess the benefit of the change made.

Questions of interest would then be:

What is the normal or standard effort?
What is the effort due to the change?

A knowledge of these goals and questions is necessary before decisions can made on the appropriate metric(s) to be used in a measurement environment.

A further aspect of the GQM paradigm is the process of goal setting. In Decision Support Systems (DSS) research [see for example [Liang 86] it is argued that for a system to be successful it must satisfy in terms of information concerning those factors which are critical to the organization's performance in the area supported by the system. But in this approach it is not the question of measurement which is being addressed, but rather the question of the object to be measured: measurement is assumed.

For example, if it is stated that accuracy of the model within the DSS is critical to the success of the system, then the metrics for accuracy are generally assumed. There exists within the domain of most of the DSS research:

1. Well established models of the activities (or process) performed and the results (or product) of those activities. These involve concepts such as profit, cost, or production levels.

2. Well established links between the activity models or results and the metrics used for those activities or results. These might include gross profit, net profit, average cost, or units produced per day.

The link between these two is the question being asked concerning the process or product which is determined by the goal being pursued. For example, the motor vehicle production manager might have a goal to increase production by 2% over the next month. Through the production model this goal might immediately trigger measurement of:

1. the existing production level measured in units per day
2. the rate of production over the target month, again measured in units per day.

This example appears trivial because the process and product models are so well understood and the metrics so well developed that the GQM sequence becomes self evident.

In the software process however, we do not have well established models of that process, nor do we have well established links with the metrics that might be used to characterize that process or product.

If for example a software manager is asked to identify his major aims he might indicate:

1. To complete projects on time
2. To complete projects within budget
3. To deliver a quality software product

and 4. To maintain development staff morale.

Based on this, systems can be designed which focus their attention on those aspects of the manager's task, but the problem of determining the appropriate metrics for product "quality", "staff morale", "on time", and "within budget" are more complex than if the goals are stated in terms of well known metrics such as net profit or average cost.

The aim of this paper is to develop a 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.

The paper is directed towards the first three levels of the Basili & Weiss list. This is less common in software engineering research which has been predominantly directed at levels three to six; defining metrics and analyzing relationships between them.

If we aim to construct a measurement environment suitable for different organizations and management, then this generalized environment must be able to include all of the six levels suggested by [Basili, Weiss 84], and to do this, a model of resource use is needed on which to build this measurement environment.

3. 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.

4. 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.

The model structure consists of a four dimensional view of the process. The four dimensional view presented here 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.

4.1 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 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, work stations).

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.

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.

4.2 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

4.2.1 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.

4.2.2 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 4.4.2.

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.

4.2.3 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.

4.3 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.

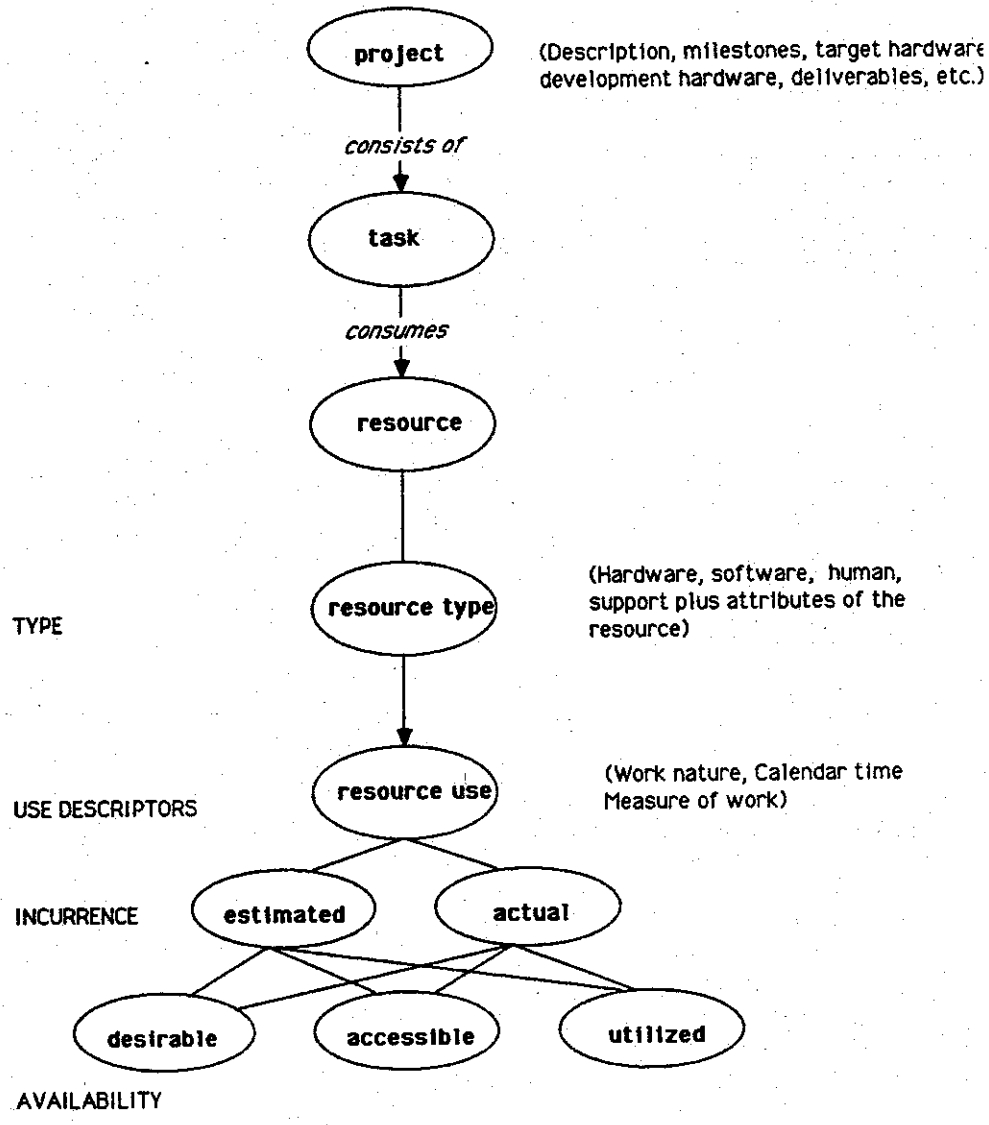

FIGURE 1. THE STRUCTURE OF THE TDC MODEL

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 78], 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.

4.4 USING THE TDC STRUCTURE

4.4.1 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 3 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.

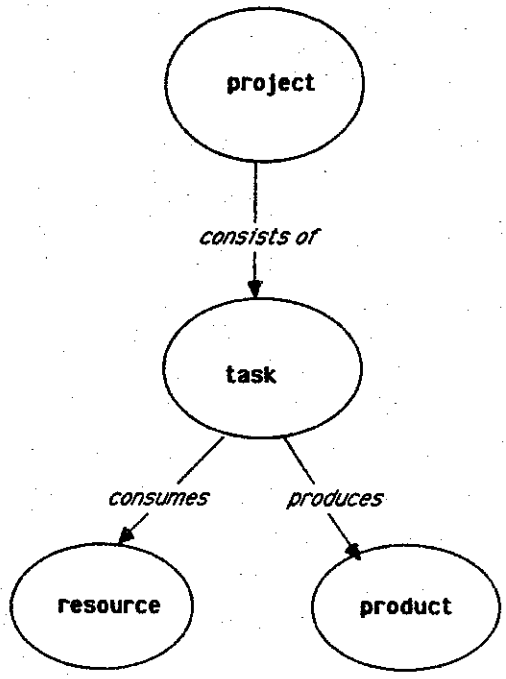

FIGURE 2. AN OVERVIEW OF THE SOFTWARE PROJECT

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 were to 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.

4.4.2 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 sub-processes of:

1. planning
2. actualization

and 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.

THE PROJECT CYCLE:

PLANNING:

Figure 3 shows the data flows, stores, and processes involved in planning, actualization, and modification. This data flow diagram shows three types of estimates being made; desirable, accessible, and utilized. The desirable resources are estimated (in process 1) by the project estimator on the basis of information concerning the project and the environment in which the project is to occur along with any project histories and/or knowledgebase which may be available. The project history data contains feedback on prior projects (or corporate memory) which can be used in the future planning processes. It therefore describes information which has been gained in reviewing prior projects which was considered to be of relevance to future projects. The project characteristics and environment information is described above in sections 3 and 4.4.1. The accessible resources for this project are estimated in process 2, again by the estimator, using the desirable resources and the corporate resource database as input along with any history or knowledgebase information. The corporate resource database lists and describes the resources which can be called on by the organization along with any commitments for those resources. Thus it may be that for the project in planning it is deemed desirable to have an expert in a particular field, but the corporate resource database shows that the only expert in the organization is committed for the period in question. This would raise a number of options including:

1. obtaining a further resource (update the corporate resource database)
2. committing to development without the expert
3. negotiating for full or partial de-commitment of the expert.
4. re-analyzing the project characteristics to determine whether the project can be modified in such a way as to remove the need for this expertise.

If it is assumed that the desirable resources remain unchanged, the outcome will be that the project in planning either does or does not have access to a desirable expert. If the expert is not available then this should be highlighted as it constitutes a project risk which needs to be closely controlled as a part of the project risk management plan. In this way the differences between the desirable and accessible resources form a significant database in the process of risk management, since this database reveals those aspects of risk resulting from decisions to develop the system with something less than the resources considered desirable.

The process of detailed project planning then continues in process 3, using the project accessible resource database as one of the inputs to this process, and generating the project resource plan which contains details of the estimated utilized resources. Once again the difference between the outputs has meaning. Just as the difference between desirable and accessible represents a database resource for risk management, the difference between accessible and utilized forms a basis for contingency planning. Resources can be "committed" in two ways. The first is when an available corporate resource is both accessible and utilized. In this case the resource can be considered as a hard commitment to the particular project. Contingency plans are also permitted in this system where corporate resources are available for the project but rather than entering the utilized list they are entered as a contingent commitment to a particular project. In that way the planning process can allow contingency planning for individual projects and for corporate resources as a whole. Thus the case may arise,

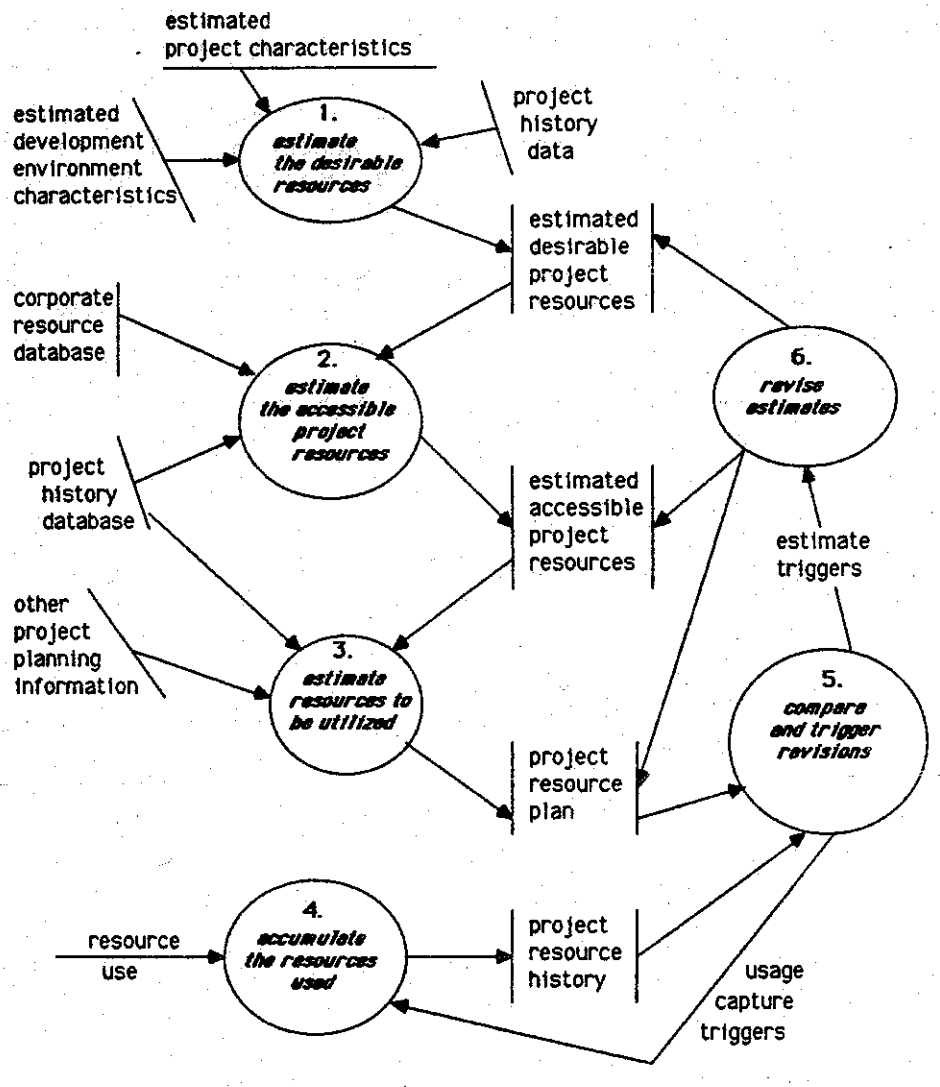

FIGURE 3. THE PROJECT CYCLE

for example, where say four projects have indicated a contingent commitment for a particular type of computer but there is only one available. A decision may be made to buy another machine based on the perceived probability that the resource will be needed; not by any one project, but by the four projects together.

ACTUALIZATION

As the software process continues, resources used are accumulated via process 4, resulting in the project resource history data store.

REVIEW

The actual's data is used in process 5 to monitor the estimates and thereby facilitate project control. If major divergences between estimates and actuals occur, this may trigger re-estimation of the needed resources which may result in:

1. a modification of the project resource plan, by allocating contingent resources to the plan

2. revised estimates of the accessible resources needed resulting in changes in both the accessible store and the plan, or

3. a revision of the desirable resources. This would trigger a major re-estimation since some of the perceptions of the nature of the project or its environment have proven to be inaccurate. The impact of these will need to pursued through all dimensions of the planning process.

This project review process can provide significant feedback for use in future planning and estimation.

THE POST-PROJECT REVIEW CYCLE

Figure 4 shows an overview of the use of the proposed structure in project reviews. The data accumulated during the project are used to review the project and generate learning based on the experience with the project. This new data consists of:

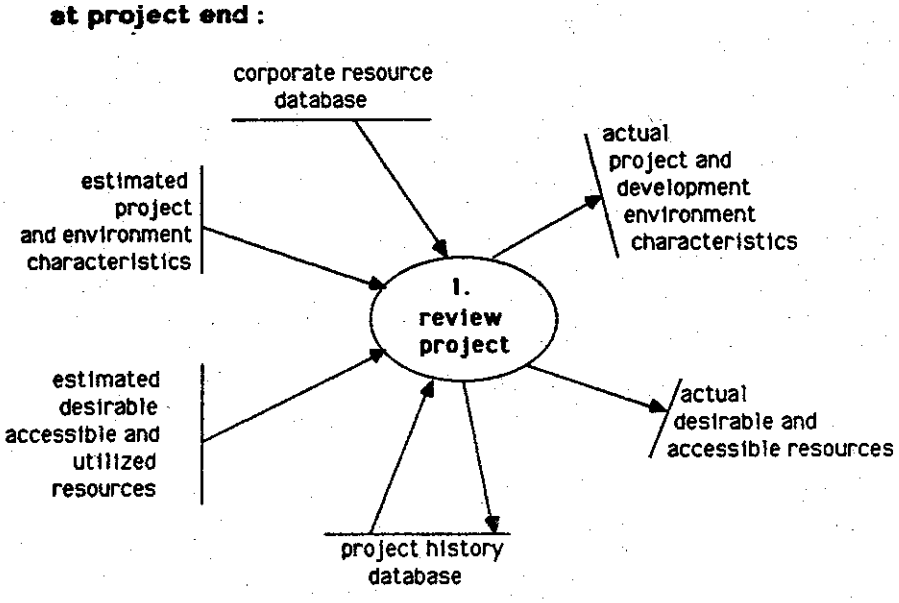

FIGURE 4. THE PROJECT REVIEW CYCLE

1. additions to the project history database

2. changes to the perceptions of the project and its environment, based on comparisons between estimates and actuals. For example, the comparison between actual and estimated milestones may reveal that shorter milestones were possible, or that desirable milestones should have been set longer. By making these comparisons it is possible to establish a project learning environment.

3. changes to the perceptions of desirable and accessible resources. Again learning is facilitated by the comparison of estimated and actual. In this way it may be learnt, for example, that aspects of the algorithmic effort estimation equations consistently over-estimated for this project. The reasons for this can be explored and, if necessary, adjustments made to the algorithm.

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.

5. THE DYNAMIC NATURE OF THE TDC STRUCTURE

In the discussion so far only brief mention has been made of the dynamic characteristic of the proposed structure. It has been mentioned on several occasions that resource use occurs at a point in time and that estimates and actuals will differ over time. However little explanation has been provided of the workings of the structure in a dynamic sense.

In order to illustrate the workings of the structure over time consider several points in time:

t_0: the point in time of the initial estimates of the project and resources
t_1: a point in time during the software process
t_2: a later point in time during the software process

At t_0 there are no actual resource values and the project characteristics are also estimates only. At point in time t_1, when work has begun on the project, the database will have actual values as well as the estimates. In order to illustrate the dynamic nature of the model consider Figure 5 in which estimates have been changed, and figure 6 in which actuals have been placed.

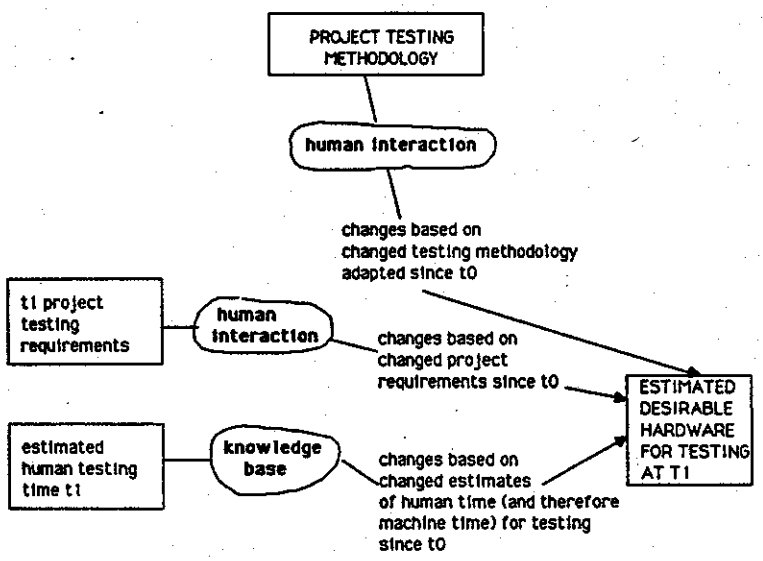

FIGURE 5. DYNAMIC ASPECTS OF THE TDC STRUCTURE AT T_1

Figure 5 shows:

1. the way in which estimates will change over time (in this example it is the estimated desirable hardware for testing purposes), and
2. the manner in which feedback can be used to provide a record of the reasons for the changes over time, and
3. the types of processing (e.g. human interaction, knowledge base) that may be involved in the estimate change.

In this example three reasons for a changed estimate are presented. Firstly there has been a revision in the estimated amount of human time which will be required for testing. This change has been processed through a knowledge base which contains knowledge of the relationships between human time and machine time in the testing process. On the basis of this, the estimate is revised. Another possibility is presented where the project testing requirements have changed due to a change in the reliability requirements of the project say. This change is shown as being processed through human interaction to derive a new estimate of the desirable hardware. The third example provided is a case where the testing methodology has been changed at the project level, and again human interaction has derived a revised estimate of the testing hardware requirements.

This example shows how changes in values in one part of the TDC structure will give rise to changes in values of other parts of the TDC structure. The processing of these changes is a complex process in which all of the assistance of human interaction, knowledge base, and algorithmic relationships will be needed to realize an accurate revision of the estimate taking into account the interactions and interdependencies which occur in the process environment [see Abdel-Hamid, Madnick 86].

The next example, shown in Figure 6, illustrates the relationship between values of the one resource instance at different points in time, without considering other TDC category changes. Here we see the actual human resource utilized in design being updated to take account of the work done in the period t_1 to t_2. At t_2 the new actual is compared with the estimate for t_2 and the resultant picture (t_1 actual, t_2 actual, and t_2 estimate) used as input to the process of updating the estimates for t_3 through t_n. The figure shows this process occurring though the facilities of a knowledge base containing information on inter-period human resource relationships as well as human interaction. The changes made to the estimates are logged and explained via the feedback mechanism of the TDC structure.

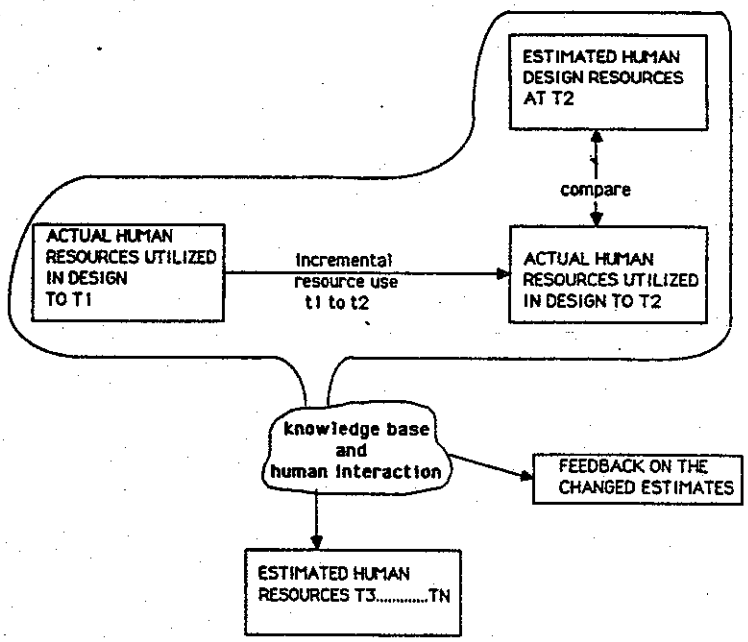

FIGURE 6. DYNAMIC ASPECTS OF THE TDC STRUCTURE - USING T_1 AND T_2 TO MODIFY T_3 THROUGH T_N

In the design of a measurement environment database it will be necessary to define all of these inter-period and inter-category relationships so that the environment is able to receive input and provide output in line with the environment goals.

6. APPLICATION OF THE MODEL IN EVALUATION

To illustrate the use of the model a simple management goal is proposed and a set of questions listed which flow out of that goal. The model is then used to provide a structure for the metrics which are relevant to answering the questions posed. A more detailed illustration is given below in section 7, in which the TDC structure is used to suggest question of interest within the GQM paradigm.

GOAL:

To evaluate the effect of a particular software engineering methodology on project resources

PURPOSE OF THE STUDY

To evaluate the development process used in order to improve it.

PERSPECTIVE

Examine the effectiveness and cost of the methodology from the point of view of the development manager.

POSSIBLE QUESTIONS

Project Level

(1) Did the use of the software engineering methodology improve adherence to the project schedule?

Adherence to schedule is a question which is not directly concerned with the expenditure of resources. At the project environment characteristics level, details of the schedule or milestones for the project are stored as outlined above. Comparison of estimated and actual utilized can be carried out and details from the feedback details concerning the milestones used to determine whether the methodology was observed to have any impact on the schedule. To establish the improvement (if any) in adherence to schedule, comparisons need to be made with the INCURRENCE feedback values for other projects having similar characteristics but not using the particular software engineering methodology.

Resource Level

A. Hardware

(1) Did the use of the software engineering methodology increase or decrease hardware resource requirements for this project as compared with those for similar projects?

This question signals access via the hardware resource type with comparisons between estimated and actual utilized (if the estimate was based on prior similar projects) and/or between actual utilized for this project and actual utilized for similar projects (based on the project environment characteristics) if a different estimation process was used. This assumes a database of similar data from prior projects.

B. Software

(1) Did the use of the software engineering methodology result in increased or decreased software resource requirements?

This question indicates a search via the software; estimated and/or actual; utilized domains making relevant comparisons. e.g. Was a compiler needed for a greater period of time than was estimated utilized? Had the question been phrased in another way, say: Did the methodology result in unexpected resource requirements? The search triggered would be via the software; estimated and/or actual; accessible/desirable dimensions of the model, using the point in time to pinpoint the stage at which it became obvious that changed resources were needed, if this were the case. e.g. Was a compiler shown as actual accessible which was not also shown as estimated accessible, or even estimated desirable? Or at what point in time did the compiler enter the list of estimated accessible, signalling a change in the understanding of the requirements of the process.

C. Human

(1) Did the use of the methodology increase or decrease the requirement for specific types of personnel?

This question requires a search using the human; estimated and/or actual; desirable; point in time dimensions to find a change over time in the human resources needed on the project. e.g. Did the estimated desirable, accessible, or utilized human resources change during the process with the feedback dimension showing a changed perception in the human requirements as a result of an insight into the methodology and its human resource implications. Another possibility suggested by the model is a search of the project environment characteristics and the resource level comparing this project with prior similar projects in a project database in terms of the human resources judged desirable at project end.

(2) How much additional training was required?

Once again this will involve a search of the human; actual and estimated; utilized; work nature dimensions comparing actual training with estimated on this project and comparing actual on this project with other similar projects in a project database.

(3) Was there a reduction in total project effort, and how was this distributed across each phase and activity?

This question is again very similar, requiring access via the human; actual and estimated; utilized; work nature and resources utilized dimensions. e.g. If the estimates were based on the use of a prior methodology, then the divergence of actual and estimate may reflect in part the impact of the methodology. The extent of the methodology impact should be stored in the feedback dimension of the comparison between actual and estimated effort if it can be estimated. Alternatively the information may be gained from a search of a prior project database comparing this project with other similar projects.

7. USING THE MODEL IN THE GQM PARADIGM

One relationship between a GQM Process Template and the TDC model is that of interaction during the application of the template. In applying the process template to a particular goal area it is relatively easy for someone to specify the first four categories: the goal type, purpose of study, perspective, and environment. It is much more difficult, however:

1. to develop the questions which are capable of answering the concern expressed in the first four categories.
2. to realize the data relevant to those questions
3. to determine the source of that data, and
4. to design the method of data analysis and presentation.

It is in these four areas that the TDC model can provide assistance

If we take an example where the purpose is to evaluate an estimation model. The TDC model suggests that there are questions of interest at both the project and process level. For example, question concerning the fit between the estimation model and the project, and the project milestones are answered at the project level. Questions concerning the resources of the process are answered at the resource level.

The TDC model suggests that:

1. There are project characteristics which need to be considered in the evaluation of the estimation model.

2. The estimation model needs to be evaluated in terms of its applicability to the four resource types.

3. The estimates made using the model should be compared with the actual resources used.

4. The desirable and accessible dimensions of the TDC structure may provide insights into the use of the model in the particular environment.

5. The extent of the fit between the estimated and actual use descriptors of the TDC model and their relationship to the actual work breakdown structure, can be used as a measure of the extent to which the estimation model is integrated with the actual process model used.

Consider this problem of an organization applying a new model for project resource and schedule estimation. One purpose might be to determine if the model is effective in that it does a better job of estimating resource and schedule than prior models or is accurate within some predetermined tolerance. For such a goal, the TDC model as presented so far and the GQM paradigm can be brought together as follows.

GOAL:

PURPOSE OF THE STUDY: To evaluate a project resource and schedule estimation model in order to assess it.

PERSPECTIVE: Examine the predictive capability from the manager's viewpoint.

ENVIRONMENT:

PROCESS QUESTIONS

QUALITY OF USE:

(in order to ensure a thorough understanding of the nature of the process being evaluated, and the use of that process)

Characterize the estimation model.

What resources does it track? (e.g. hardware, software, human, support)

What activities does it track? (e.g. design, coding, review,...)

What milestones are used in the model?

Was the estimation model designed for the project class?

Was the estimation model designed for the project deliverables used in the organization?

How well was the estimation model followed?

What desirable items were not available? (compare the desirable lists with the available lists.)

What accessible items were not utilized? (Compare the available lists with the utilized lists.)

How well was the model communicated to the developers?

How well was the model applied?

DOMAIN OF USE:

(in order to understand the environment in which the process is being applied)

How well was the software problem understood?

How well were the methods to be used understood?

How well were the tools understood?

How well was the hardware understood?

How well was the software environment understood?

EFFORT OF USE:

(in order to understand the cost of the application of the process)

What was the cost of applying the estimation model?

What was the relationship to the total project cost?

What was the cost of each activity?

What were the dates for the completion of each milestone?

EFFECT OF USE:

(in order to assess the impact of the application of the process)

What were the differences between estimated and actuals for each phase, activity, and in total?

What are the resources (by type) expended on this project versus the resources expended on the normal software development of the same type?

FEEDBACK FROM USE:

(in order to learn from the evaluation process)

For each difference between estimated and actuals, what were the reasons recorded?

What changes need to be made to the model to improve its predictive power?

To what extent is it felt that the estimates drove the actuals in terms of values incurred?

8. VALIDATING THE MODEL

Two 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 [Tausworthe 79] and [Penedo, Stuckle 85].

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 3 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.

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

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

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 only 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. 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.

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

Table 2. Tausworthe Derived Entity List

9. CONCLUSIONS AND IMPLICATIONS AT THE RESOURCE DATA LEVEL

The discussion above 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 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 proposed 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 GQM process paradigm, so that questions which answer the resource purpose of the study are highlighted and the measures appropriate to those questions are suggested.

10. REFERENCES

[Abdel-Hamid, Madnick 86]
T.K.Abdel-Hamid, S.E.Madnick, "Impact of Schedule Estimation on Software Project Behavior," IEEE Software, 3,4, June 1986, pp. 70-75

[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.

[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.

[Jeffery, Loo 87]
D.R.Jeffery, C.C.Loo, "Metrics of Project Size in the Measurement of Programming Productivity: An Evaluation of Function Points," Technical Report, Department of Information Systems, University of New South Wales, January, 1987.

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

[Liang 86]
T.P.Liang, "Critical Success Factors of Decision Support Systems: An Experimental Study," Data Base, Winter, 1986, pp. 3-16.

[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.

[Selby, Basili, Baker 85]
R.W.Selby, V.R.Basili, F.T.Baker, "Cleanroom Software Development: An Empirical Evaluation," Technical Report TR-1415, Dept. of Computer Science, University of Maryland, February, 1985.

[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.