USING A WORKLOAD INFORMATION REPOSITORY

Mapping Businesses and Applications to Servers and Processes

Dr. Tim R. Norton
Inovant LLC, A Visa Solutions Company, PO Box 266001, Highlands Ranch, CO 80163, USA
Email: TNorton@visa .com

Keywords: Workload characterization, modeling, application analysis, business metrics, business drivers.

Abstract: Workloads are often defined differently within an organization, depending on the purpose of the analysis, making it difficult to compare analysis from different points-of-view. WIRAM (Workload Information Repository for Analysis and Modeling) is a preliminary implementation of a database repository to collect application and system information about workload groupings and their relationships. This information can then be used as a component of the ITIL CDB to define consistent workloads from business products to computer systems, regardless of the analysis or modeling tools used or the objectives of the analysis.

1. INTRODUCTION

The term “Workload Characterization” has been defined, redefined, discussed, explored, categorized, investigated and researched until it seems like there would be little mystery left. Why then are there still so many papers, articles and new techniques on the subject? The initial reaction is to say that it’s because the “right” definition hasn’t been found. But what if it has? What if ALL the definitions are “right” definitions? Maybe the question shouldn’t be “Which is the right definition?” but rather “When should each definition be used?” Consider that each of the proposed definitions is absolutely correct given the perspective and point-of-view of its creator. Therefore, instead of addressing the term “Workload Characterization” let’s look at the definition of the workload itself and the relationships between workloads.

To begin, we must ask what "workload" means? It's a common term so everyone assumes they know what it means, but are they using the same meaning? The American Heritage Dictionary (1994) defines “workload” as:

> ***work·load*** *(*wûrk′lōd′*) n. 1. The amount of work assigned to or expected from a worker in a specified time period. 2. The amount of work that a machine produces or can produce in a specified time period.*

Modifying the above definition to fit the needs of the computer systems capacity planner at the most general level, the term "workload" means the work (processor time, I/O’s, etc.) expected from a ‘computer worker’ (a collection of processes, users, applications, etc.) that can be thought of as a single entity. Using such a general definition means a number of problem areas need to be considered. The first is that different people have different objectives so they tend to look at what things belong together differently (Norton, 2002). Another is that the availability of measurements also influences what can or should be grouped. The third is the meaningfulness of any analysis to the business. The use of management tools, such as Balanced Scorecard, that connect all aspects of the business (company financial performance, customer satisfaction, employee performance, Information Technology resources, etc.) (Lebsack, 2002) require a consistent set of definitions across the entire company. One source of such definitions is the ITIL (Information Technology Infrastructure Library) framework (OGC, 2003). The ITIL Service Delivery book includes a section that describes the general requirements of the CDB (Capacity Database) (OGC, 2003, Section 6.5 Planning and Implementation). A detailed analysis of all these areas is beyond the scope of this paper, but we can touch on all of them by looking at a practical approach to identifying and tracking workload information.

Given these constraints, what’s really needed is

a way to capture all the important information about these entities and their relationships so that views of what's going on can be generated from different perspectives. Some initial work was done at MCI to develop a system called AITS (Application Information Tracking System) that would collect information about applications into a single repository (Boyd, 1996). AITS was intended to interconnect existing information sources, such as the security database, naming standards and the inventory control system, with system configurations and application details. AITS was never completed due to organizational changes and the eventual outsourcing of most of the IT staff.

2. WIRAM OVERVIEW

This paper presents a preliminary implementation of WIRAM (Workload Information Repository for Analysis and Modeling), a database repository of definitions and relationships that provides workload consistency from business products to computer systems. While inspired by AITS, WIRAM has a much narrower scope with focus on just the information necessary to understand the workload definitions used in performance, capacity and modeling reports. The high-level objectives of WIRAM are:

1. Define what is in workloads.
2. Define what drives workloads.
3. Define what measures workloads.

WIRAM provides consistency across a company whenever computing resources are discussed. For example, a business level report might show that an application is extremely profitable because it uses so few computer resources but managers may not agree with the assessment of what resources are used. In addition, if there is disagreement as to what constitutes the application, or even worse, widespread confusion because other presentations have used the same application name but shown vastly different resource usage values, then the focus shifts from presenting a positive business situation to bickering about the meaning of data points. By creating a WIRAM, a company has a central repository of record that defines what resources each application uses and how applications are combined into business products.

The ITIL CDB requirements identify a number of general areas for data collection that include "business data from the business plans and strategy" (OGC, 2003, Section 6.5), but do not address a mechanism to connect the business information with actual resource consumption data. The WIRAM provides that connection.

3. WIRAM DESIGN

The design of a WIRAM is based on the concept that everything should support the business.[1] That means that all of the computing resources used in a company should be accounted for as part of what is required to implement the objectives of the business. The purpose is simply to report on current usage and predict future usage based on business forecasts instead of system measurements. What's the difference? Consider that the product line business managers have a better view of growth projections based on their sales objectives. After all, those are the numbers the sales staff is trying to make happen because that's what their commission (or whatever performance review measurements the company use) is based on. The problem is that product projections cannot be directly correlated to resource usage. The WIRAM defines what the business is called, what drives the volume and what components that make it work (applications, systems, computers, etc.) should be included. The repository does not contain the actual business forecast or performance data but does define the source of that data and the relationship to the business. This allows everyone in the company the opportunity to provide input into the definitions. The end result is that everyone agrees about the meaning of a business or product, how it's projected to grow and what gets rolled into it.

Of course the implementation of a WIRAM will depend on the specifics of each individual company. The author has chosen to use the Business Metric of Interest (BMI) approach (Kaminski, 2003) but other techniques for quantifying business activity could be used. How to identify and measure appropriate business metrics is beyond the scope of this paper. Techniques for actually measuring resource usage and reporting it are beyond the scope of this paper as well, but there are many excellent papers that address these areas (Domanski, 2002; Johnson, 2003; Kaminski, 2003).

---

[1] Throughout this paper the term "business" is used to mean any structured organization with defined objectives. While commonly used to mean a for-profit company, such as Visa USA, it can also mean a non-profit company, a military organization or a service club. Each of these has "products" (goods, services, or bombs) that they deliver. If computing resources are used in the creation and delivery of the products then a WIRAM could be used.

4. WIRAM LAYERS

As stated above, the design of a WIRAM should be tailored to the requirements of the company. However, there seems to be a general design that could provide a starting point for customization and it may work as is in many cases. This design is based on a layered approach that puts a business product at the top. The term 'layer' is used because each is built on, or layered on top of, what's below it. Figure 1 shows an entity relationship diagram with these layers, called Major Categories, down the center (the other entities, called Modifier Categories, will be discussed in the following subsection).

The Major Categories are:

1. Business
2. Product
3. Application
4. Application Component
5. Workload

4.1 Major Categories

Each business is composed of individual related products. Products are implemented with applications, which are composed of application components. Notice that the top three levers are totally business focused. They describe things that everyone in the company should be able to relate to because they are what the company does. These levels allow us to talk about resource usage and capacity requirements in broad business terms but still have enough granularity to address idiosyncrasies unique to each application. The *Application Component* layer is where we start mapping the business to the computing infrastructure. *Application Components* are associated with workloads in the more traditional sense. It isn't until this level that we start looking at things we can measure with all of the wonderful performance and modeling tools at our disposal.

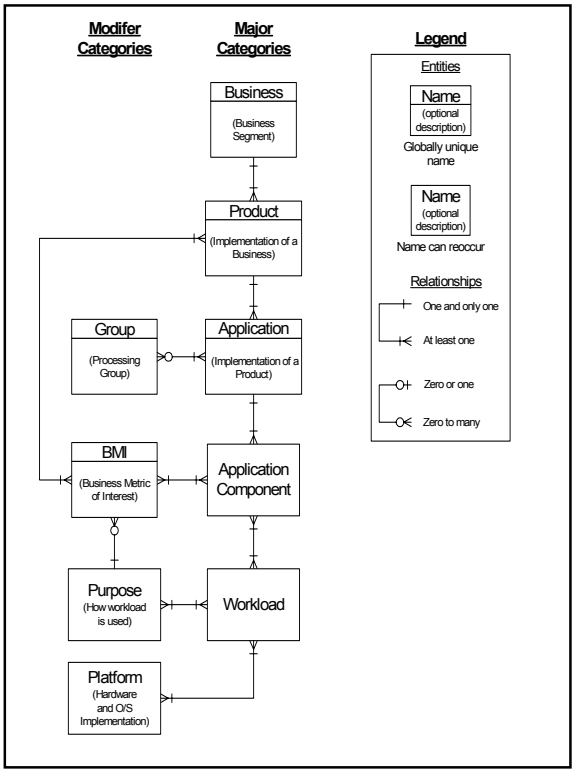

Figure 1: WIRAM Entity Relationship Diagram

4.1.1 Business Layer

A *Business* defines an overall, or strategic, objective of the company. A *Business* is a set of one or more related products that are available to customers and address a common set of customer needs and requirements. For example, a company may be considered in the automotive business because it sells new cars, used cars and car parts. A company does not have to include all of the possible or usual products in a *Business* (as the above example does not include car repair or manufacture) and in fact may have only a single product (a company only selling new cars would still be in the automotive business). The name used for a *Business* is unique throughout the company and usually the industry.

4.1.2 Product Layer

A *Product* is conceptual customer deliverable and is associated with something a customer can buy or subscribe to. A *Product* is generally what the marketing or sales staff offers to customers (a car or a part or a service contract). The name used for a *Product* is unique throughout the company and possibly the industry. A *Product* name is often trademarked and identified with both the business and the company.

4.1.3 Application Layer

An *Application* is the implementation of a *Product* that provides a tangible customer deliverable. An *Application* can be a product in itself (a used car sold as is) or one of several different deliverables that may be either required by the nature of the *Product* (the new car) or provide optional "extras" that enhance the *Product* (such as a specific trim package). An *Application* name is unique throughout the company because it identifies a unique im-

plementation within the company. Even if a COTS (Commercial Off-the-Shelf Software) product is used, the implementation name (which may or may not be the same as the COTS product name) only identifies something internal to the company. The *Application* name is generally not unique in the industry, and may or may not be known to customers.

4.1.4 Application Component Layer

An *Application Component* is one of the parts of an *Application* that has meaning only within the context of a specific application. It is more about the implementation and working of the *Application* than about the *Product* objectives. Even though an *Application Component* is associated to only a single *Application*, the name of the *Application Component* may be used as the name of a *Component* of another *Application*. Two *Application Components* having the same name means that the two components perform the same type of function for their respective *Applications*, not that they are the same component. Because one of the objectives of a WIRAM is to provide aggregation of low-level workloads into business functions, an *Application Component* can be part of only one *Application*. If it appears that an *Application Component* needs to be associated with two *Applications* then those *Applications* are probably defined too granular and should be merged into a single *Application*. Following on the automotive analogy, an *Application Component* could be a financing package that would have the same name and function for both new and used cars but the term details would be different (shorter for a used car because it isn't expected to last as long). An *Application Component* name may or may not be unique throughout the company, is almost certainly not unique in the industry, and is seldom known to customers.

4.1.5 Workload Layer

A *Workload* identifies how resources are used, or consumed, by an *Application Component*. The exact meaning of a *Workload* is dependent on the associated *Application Component* but it can also be meaningful to refer to the *Workload* as a general category. Each of the required forms to buy a car can be considered a workload; each car needs the same forms but how they are completed will differ with each type of car (*Application*) and each specific car (*Application Component*). We can note how long it takes to fill out the form for a car in general, for a sedan vs. an SUV, or for a specific car. We can also note how much time is spent each month filling out a specific form. All are valid views into how the resource (the salesman's time) is used. A *Workload* name may or may not be unique throughout the company. While a *Workload* name is almost certainly not unique in the industry there are many standard names that have generally the same meaning throughout the industry, such as 'database' and 'OLTP' (on-line transaction processing) and may be known to customers in that context. Care should be taken when assigning these names so that they do not conflict with the implicit meaning of the name or that they are not so general that they are not useful. A *Workload* name of "Database" would be so general, and used by so many *Application Components*, that it would loose its meaning. A *Workload* name of "OE-DB" would be more meaningful, even if it was used by several *Application Components*, because it would be more specific to the processes, user-ids, etc. associated with the ONLINE-ORDER *Application Component*. Because a *Workload Name* is not unique, it requires the associated *Application* and *Application Component Names* for identification. Using "OE-DB" does not identify a specific *Workload* because that name can be used in several *Application Components*. Examples of fully qualified *Workload Names* could be names like ORDER-ENTRY.ONLINE-ORDER.OE-DB or ORDER-ENTRY.INVENTORY.OE-DB, both of which are part of the same *Application* (ORDER-ENTRY) but in a different *Application Component* (ONLINE-ORDER and INVENTORY). They are different workloads because they contain different processes, or different user-ids, or different terminals, or different something. If both *Application Components* actually use the same workload, then there would have to be a proration amount at the *Application Component* level. While the WIRAM does not require or enforce uniqueness of names, it is suggested that unique names be used wherever possible. It is the responsibility of the implementer to insure name uniqueness and to derive proration rules.

A workload is defined by anything that can be identified, such as process names, user-ids, terminal addresses, accounting codes, system names, etc. Therefore, the definition can become quite large and hard to maintain if everything is explicitly identified. Wildcards or patterns can be used but care must then be taken to avoid double counting (i.e., if workload X includes process=ABC and workload Y includes process=*, then process=ABC is really included in both workloads). There is deliberately a great deal of flexibility in these workload definitions because real applications are seldom very orderly. Shared databases and reuse efforts can lead to a workload being spread across many application components. The proration field allows multiple *Application Components* to reference the same

workload definition (although it would look like different definitions because the name in part of the key and therefore in multiple records). However, there is nothing in the design of the WIRAM to prevent the sum of these proration from exceeding 100% and the implementer should use this feature with care.

4.2 Modifier Categories

Modifier Categories apply an additional level of constraint to one of the Major Categories so that it can be viewed differently for a specific purpose.

The Modifier Categories are:

1. Group
2. Purpose
3. Platform
4. BMI

4.2.1 Group Modifier

The *Group Modifier Category* provides granularity at the *Application* level that is independent of how the application works. A *Group Modifier* could be used for business reasons (car rental Premier clients versus normal clients) or for load balancing (east region versus west region) or anything else that makes sense. An *Application* can be view as a whole (all new cars) or as a modified group (all four-door red sedans). It is generally more meaningful if the groups are such that an Application belonging to one Group cannot belong to another Group, but this is not a requirement. It is also possible to have multiple Group level modifiers, in which case each would have a unique name instead of "Group" (such as the Color Group and the Body Style Group).

4.2.2 Purpose Modifier

The *Purpose Modifier Category* provides granularity at the *Workload* level that is independent of how the workload is defined; instead, it addresses how a workload is used (a four-door red sedan could be used as personal car, a taxi or the fire chief's official car). When looking at computer systems, common *Purpose Modifiers* might be production, test, development, etc. The objective of the *Purpose Modifier* is to allow an application to be viewed from different business perspectives. For example, when looking at the total cost of the ORDER-ENTRY application it doesn't matter if it's production or test, it all goes into the cost. On the other hand, when looking at the resource requirements for an increase in the number of orders processed, we only want to look at the part of the application (production) that applies. The same can be said for adding new features (development) or tracking down a major problem (test). Each of these uses the same definition of the workload but distinguishes what should be included in the study by how it is used. A BMI is associated with a *Purpose* to provide the meaningful context for the BMI. As in the above example, the number of orders placed by customers would be a good indicator for only the part of the application associated with the *Purpose* PRODUCTION. Workloads associated with other Purposes would not be affected by changes in the number of orders placed but might change with some other BMI, such as the number of developers.

4.2.3 Platform Modifier

The *Platform Modifier Category* provides granularity at the *Workload* level that is dependent on how the workload is implemented at the hardware and operating system level. For example, a component of the OE-DB workload might be an entity called DATABASE that is defined as service class DB01 on MVS and process DATAB-MAIN1 (with all child processes) on Unix. The *Platform Modifier Category* fills two key requirements: first, it allows for platform specific definitions of workload entities and, second, it allows analysis to be done at the platform level ("How much will the MVS part of ORDER-ENTRY grow?").

4.2.4 BMI Modifier

The *BMI (Business Metrics of Interest) Modifier Category* connects the *Application Component* level to the business by identifying what part of the business affects the workloads that apply to the *Application*. In the example of the ORDER-ENTRY application, by tracking the Business Metric of ORDERS-ENTERED against the resources consumed for the ONLINE-ORDER workloads (the workloads defined by the ONLINE-ORDER *Application Component*) we can see the resources required for each order. In addition, other organizations within the company can often provide very accurate forecasts of these business metrics because they are related to sales goals and quotas. If we know what it takes to process an order, and we know how many orders the sales staff expects to enter, then it is a relatively easy calculation to determine the resources that will be needed on each platform. (Kaminski, 2003)

The *BMI Modifier Category* is not a modifier in the same way as the other modifier categories. Notice in Figure 1 that the BMI entity is connected to both the *Application Component* entity and to the

Product entity. While the *Application* entity connects the *Product* entity to the *Application Component* entity (and through it to the *Workload* entity) in a functional or procedural sense, the *BMI Modifier Category* connects them in a behavioral sense. The BMI modifier is a more formalized than the traditional workload driver because of this connection to products.

5. WIRAM IMPLEMENTATION

WIRAM is currently implemented as a Microsoft Access database because that was the tool the author had. There is no requirement that Access be used but it was an obvious choice because of the relatively small size of the WIRAM database, the flexibility of using Access with an IIS web server and the fact that Access is standard in the company's desktop software installation. Appendix A, at the end of the paper, lists the SQL statements to create each of the tables in our implementation of WIRAM.

6. CONCLUSION

This conceptual description of WIRAM (Workload Information Repository for Analysis and Modeling) shows the preliminary implementation of a database repository of workload definitions and relationships. The objective of WIRAM is to provide workload consistency from business products to computer systems by defining each of the workloads, what about the business changes resource usage and what metrics can be used to measures the workloads. The WIRAM is designed in layers from Businesses at the top to Workloads at the bottom. Modifiers Categories allow views of the relationships for specific purposes. This database design provides a starting point for customization specific to a company's needs and objectives. Once implemented, a WIRAM will provide a consistent set of definitions for performance and capacity planning studies. Presentations can simply refer to the WIRAM without having to spend time on workload explanations and arguments. Studies can be meaningfully compared. Using WIRAM to map businesses and applications to servers and processes improves the efficiency and effectiveness of application analysis and system modeling.

7. REFERENCES

American Heritage Dictionary, Third Edition, (Software) Version 3.6p, 1994, SoftKey International Inc.

Boyd, Linda; Norton, Tim R., 1996. *Application Information Tracking System (AITS)*, 1996, MCI Resource Modeling Group, Internal Presentation.

Baldwin, Ian C.E., 2002. *UNIX Capacity Planning for Mainframe Capacity Planners, A Rapid Development, Low Cost, Case Study*, Proceedings of the Computer Measurement Group, CMG2002, Reno, NV, December 8-13, 2002.

Domanski, Bernard, 2002. *System Performance Management and Capacity Planning Tutorial*, Proceedings of the Computer Measurement Group, CMG2002, Reno, NV, December 8-13, 2002.

Johnson, Adrian; Spellmann, Amy; Gimarc, Richard, 2002. *How do you eat an elephant? or How to digest application performance in bite-size chunks*, Proceedings of the Computer Measurement Group, CMG2002, Reno, NV, December 8-13, 2002.

Kaminski, Ron; Ding, Yiping, 2003. *Business Metrics and Capacity Planning*, Proceedings of the Computer Measurement Group, CMG2003, Dallas, TX, December 7-12, 2003.

Lebsack, Rick, 2002. *The Value Of It: A Strategic Perspective*, Proceedings of the Computer Measurement Group, CMG2002, Reno, NV, December 8-13, 2002.

Norton, Tim R., 2002. *The Effect of Workload Groupings on Distributed Transaction Capacity Models*, Proceedings of the Computer Measurement Group, CMG2002, Reno, NV, December 8-13, 2002.

OGC, 2003. Proprietary commercial electronic republication of the ITIL (Information Technology Infrastructure Library) documentation (www.itil.co.uk) by HMSO, St Clements House, 2-16 Colegate, Norwich, United Kingdom, in consultation with the Office of Government Commerce (OGC), Rosebery Court, St Andrews Business Park, Norwich, United Kingdom (www.ogc.gov.uk).

Smith, Connie; 2003. *Best Practices for Software Performance Engineering*, Proceedings of the Computer Measurement Group, CMG2003, Dallas, TX, December 7-12, 2003.