Autonomic Logistic Analysis

(Revised Edition – 2011)

Robert A. Butler
TFD Group

Précis

Imagine a world in which the rivulets of small transactions occurring throughout a running system's operating and support environment flowed together into a tumbling river of insight into the health of the system. This river of data, translated into measures of effectiveness, logistic policy decisions, budget estimates and the like would be at the fingertips of decision-makers, allowing them to answer every question and foresee all onrushing problems quickly and at little cost. This is the aim of autonomic logistic analysis. Just as the human physiology uses the autonomic nervous system to provide warnings and automatic responses, we can integrate analytical processes and their data to provide warnings and answers before the decision-maker has formulated the questions.

The current practice of logistic and cost analysis comes nowhere near this ideal vision. Although practiced widely for almost half a century, analysis in this arena has generally been insufficient to play its intended role in crucial decision processes. These processes, spanning the life cycle of hardware systems from early specification and design to operation, maintenance and the final disposal of systems, have significant implications for social welfare. Accordingly, the failure of the analytical community to deliver adequate and timely results presents us with an opportunity, by fixing these problems, to improve social benefits significantly.

Of the three elements required to deliver analytical product – analysts, models and data – there are supply problems with all three, but the most significant barrier to efficient production appears to be a number of problems with data. Data are generally ubiquitous, but somehow inappropriate for the needs of analysis. Data elements are often defined incorrectly, stored incorrectly, not updated, and the data of interest hidden among masses of irrelevant data. The result is that typically, 90% of an analyst's time is spent organizing data and only 10% performing analysis with it[1]. The consequence is to make analytical product late, inadequate, naïve and, frequently, just incorrect. That is, shoddy goods at high prices. A secondary effect results from the fact that the use of analysis by decision-makers follows the law of demand and supply: the higher the cost of a good (in this sense, inconvenience and low quality imply high prices), the less of it will be taken from the market. Thus, good analysts and models are also scarce.

This paper explores the nature of logistic data and why it has been such a bottleneck to progress, then suggests an architectural approach for data systems that could solve the problem. The appropriate structure of databases intended for traditional (acquisition) logistic analysis is discussed. The data requirements for analysis of running systems are considered as an additional complication. The goal of integrating the solutions for acquisition analysis and analysis of running systems then presents us with an interesting proposition: the transactions data stream lying at the heart of running system management provides a rich resource for the continuous updating of the static analytical database. Such a continuously updated database, in turn, makes continuous update of analytical outputs feasible. These can then be compared to flag changes that require the attention of managers. This is the essence of autonomic logistic analysis.

[1] This is, of course, nothing more than armchair empiricism. Working analysts make such statements to each other whenever the topic arises, however, and the author does not recall hearing anything that would stand as a refutation. The thesis herein would be no less significant if the proportions were reversed.

The Potential Benefits of Analysis and Our Failure to Achieve Them

The social benefit to be derived from investment in complex systems is significantly enhanced by the application of sound, effective and efficient decision-making. This is true throughout the system's entire life cycle, from acquisition to operation and support and thence to disposal. It is also true that the quality of decisions can be significantly enhanced by the timely application of analysis.

The function of analysis is to provide the decision-maker with both foresight and insight; foresight in the sense of anticipating outcomes of decisions without having to take them first, and insight into the "physics of the problem"[2] or the fundamental relationships that will drive all outcomes. The benefit of better decisions and the ability to improve decisions with better analysis makes the tools of analysis of great importance and their method of design and use of crucial importance to society.

Motivated by this logic, formal requirements for cost and logistic analysis of systems as an aid to planning and decision making have been common during most of the last half century. The best one can say, however, is that value of the resulting analytical effort has been uneven. Fulfilment of such requirements has been costly, naïve, incomplete, inaccurate, inappropriate, late and as often as not, appropriately, ignored. On the other hand, the promise of analysis – its potential to deliver crucial information to decision-makers at just the right moment – makes all of us, both producers and consumers of analysis, continue trying.

What accounts for this state of affairs and what might be done to improve the situation?

The low state of analytical effort in logistics appears to be due to deficiencies in all three of the essential resources required to produce good analysis: skilled analysts, good models and data. These resources, like the legs of a three-legged stool, must all be present and in good condition to produce useful analysis. The lack of any of the three will make the others all but useless. All three of these resource groups are demonstrably insufficient to one degree or another in most logistics establishments. Among the three, however, it seems that the most problematic is data.

Deficiencies in the way data elements are defined in our community, the design of the compendia[3] in which they are stored and the methods of update are all so glaring that even the most skilled analyst finds himself spending most of his time fixing the data and very little of it on the analysis itself. This misallocation of time and resources, even in the presence of top quality analysts and models, leads to a poor record of delivering analyses (i.e., information) to decisionmakers. Before long, the decision-maker's frustration leads him to believe that the analyst himself is incompetent and he forms the habit of making his decisions without benefit of analytical guidance.

The situation in which we find ourselves can be thought of as a marketplace for analysis. The ramifications of the deficiencies noted have raised the cost of (useful) analysis so high that decision-makers demand less and less of it. The result is a certain poverty in decision processes.

We do not suffer so much from the lack of good analysts or models, but from the intractability of the data problem. Furthermore, if these data problems are conquered, whatever deficiencies we may

[2] This is Dr. Craig Sherbrooke's phrase. He means by it, the relationships between the phenomena that drive outcomes and how they are codified in the variables and equations of a mathematical model. While a model user need not understand the underlying math, he soon begins to understand the phenomena modelled in a deeper way than mere experience allows. This is because you only get to do experience once, whereas you can run the model repeatedly, "experiencing" as many different variations on what reality might bring as you wish.

[3] I hesitate to call such specifications as MIL-STD-1388-2A/B, DEF STAN 00-60 and other similar publications "databases." They are, in fact, no more than data exchange standards, although MIL-STD-1388-2B was briefly hailed as a database design – presumably to gain wider acceptance of it prior to its publication. By the time it was published, however (actually, six months before), its principal author denied in conversation that it was ever intended to be anything but a data exchange standard. Somehow, this change of view failed to receive the same publicity as the earlier assertion that it was a database specification and we are still saddled with the consequences.

have in the supply of either of the other two resources will quickly be overcome. That is, once the data cease to be such a problem, analysts will be able to produce analytical output so much more quickly and cheaply, that demand for their product will increase. In turn, as in any healthy market, the derived demand for the other inputs, skilled analysts and good models, will also make itself known in the form of more and better analysts and models.

What follows begins with a brief look at the market for analytical product. Next is a discussion of the idea of the three-legged stool and the crucial role played by data. Having focused on data, we next turn to a discussion of data and data management as these concepts apply to logistics. There are some surprising differences, we argue, between the treatment of these subjects in logistics and the way they are handled in other fields. After dealing with the general concepts of data management for logistics, we turn our attention to the idea of autonomic logistic analysis and how it might be accomplished. The final section uses the design of a supply management system called Supply Chain Optimization to illustrate how an autonomic analytical system might be put into practice.

The Market for Analytical Product

Analytical practice is an outgrowth of many influences. Most of them can be thought of as either the character of demands or the conditions of supply. By demands, we mean consumers' perceptions of what questions need analytical answers. By conditions of supply, we refer to the data, models, computing machinery and analytical talent available to provide those answers. That is, members of the analytical community are driven to what they do by the character and strength of demands for their product on the one side and the cost of providing something like what is wanted on the other. To speak of demand and supply, in other words, is also to indicate that a market-clearing price is somehow involved; a hidden hand guiding the parties to a degree of effort balanced by the value of the outcome.

It is the function of the price in any market to attenuate the processes of supply when their output's value at the margin finally exhausts the interest of the buyer. This probably explains why many analytical efforts, initiated with great hope as ambitious projects, are ended before they have reached their full flower – because at some point the buyer loses confidence in the value of the expected outcome, compared to its increasingly evident price.

Nevertheless, the conditions of supply have improved for analytical effort just as they have for many other endeavours. Not least among these advances are the development of awesome personal computing power and the parallel development of the technology embedded in database management systems. These two advances, confined to the last quarter century, have had a profound impact on the conditions of supply of analysis. I argue here, however, that they have not had a commensurate influence on the organization of the market for analytical product. All the pejorative adjectives marshalled at the outset (costly, naïve, incomplete…) yet remain true.

The persistence of poor performance in the production of analytical output must be explained by something other than computer and database technology, since those have improved immeasurably. Other places to look are the character of demands and the remaining conditions of supply – analysts, models and data.

While analytical talent is probably more or less constant[4], advances in computing machinery and database technology don't address the data that reside in these more advanced database management systems. The data, any analyst will point out, are the crucial problem, the central issue in the failure of analysis to bear adequate fruit.

[4] It is tempting to believe, however, that the field of logistic analysis might start attracting more of the best analysts if the other conditions were improved – better data and models. There is a chicken-and-egg problem with such speculation, of course.

Data, however, are the handmaiden of the analytical process, which itself is defined by the models used by analysts. So we must examine the models for clues to the failure of analysis. Here we will find a rich source for speculation.

The models betray every flaw in the analytical market place, from poorly defined demands, to naïve formulation of answers, to the dependence, no longer true, on slow computational machinery and absence of data management tools and so on. It is not the models alone, however. For example, the tendency to define compendia of data addressing logistics in general, without a proper understanding of what data might be needed to run the models that provide the answers, has led to enormous waste. The result is the unwarranted assumption that the data must be available, because so much wealth has been squandered on compiling them.[5]

The problem alluded to above – data collections that don't serve the needs of the models – is paramount even though an attempt is made to design the models to react to the data available. They also attempt to react to the character of the questions asked. Early models of life cycle cost were seen as decision tools used to determine whether an acquisition program should or should not be pursued. Once the question was answered, the model and its answers were considered overcome by events. A similar phenomenon can be noted regarding the use of level of repair analysis tools. The assumption was that this question[6], once answered, would never arise again. This despite the quite obvious fact that, in a 20-year life cycle, virtually every condition assumed to be true in the model – the cost of labor and parts, the reliability of the components, the cost and conditions of maintenance, the location and rate of effort of the system – would change completely. The same thing happens with spare stock planning in most organizations. Initial recommendations are made based on faulty or theoretical data and assumptions about operational tempo and geographic disposition and the resulting stock level recommendations are honored forever after, as if they were sacred.

For some reason, no one seemed to notice that the same questions (least costly repair policy, optimal stock levels etc.) arose repeatedly until the system was retired – and that the inputs and therefore the answers produced by these analytical processes would be different each time.

What these illustrations bring us to is the central thesis of this paper. For many reasons, analytical effort has been carried out as a static, one-time exercise, even though the subject of analysis has been dynamic and the policy decision that gave rise to the analysis remains in question after the exercise is finished. This error, in turn, has influenced the character of the models brought to bear, the view of, and consequent design of, logistic data compendia and, indeed, the way in which analysts and those who employ them see their mission.

The truth is that the systems we wish to understand and manage are like living organisms. Their health is dynamic, always subject to change. To manage these systems to achieve and sustain high levels of support effectiveness for reasonable cost requires a new view of the formulation and application of analysis. In this view, analysis must also be a continuous or living endeavour. Questions worth asking must invariably be asked over and over again as the organism, whose health they seek to measure acts on and is acted upon by its environment. Those changes signal the need for changing policy and technique, whose formulation and implementation require a continuously revisited analytical landscape.

Much as the autonomic nervous system protects the human organism, adapting our physical reactions to the dynamic world we live in, the logistic health of running systems requires something like an autonomic decision process to insure its health through a dynamic life cycle. We neither

[5] This is all but definitive of the importance *and* failure of the LSA data collections. The author recalls an astonishing conversation with an RAF Group Captain sometime in the late '90s in which the Group Captain announced proudly that he and his colleagues had understood the problem with the American MIL-STD-1388-2B data standard, which was that it placed too much emphasis on the analytical processes that produced the data. He and his colleagues had determined that the British database should be free of such encumbrance!

[6] Should I repair or discard this item when it breaks, and if repair, at which echelon (level) of the support organization?

advocate nor believe in the efficacy of automated decision-making. However, the more readily the decision-maker can be provided with pertinent information (the output of the analytical process) the more efficiently he can navigate through the changing circumstances influencing the health of his systems. Such a capability rests entirely on the timely availability of all the resources required provide useful and accurate analysis.

Thus, autonomic logistic analysis.

The Three Legged Stool

To perform useful analysis any organization needs three fundamental resources: analysts, models and data. If any of these resources is missing, it will be impossible to derive benefit from the other two. This seems a straightforward proposition. Nevertheless, one often sees organizations that lack one or another of them. The consequence is that their "analytical cell" has never produced satisfactory inputs to the decision making process. A related proposition is that quality of analytical product is probably regulated by the least capable, suitable or well trained of the resources, not the best. Let us consider each leg of this analytical stool.

Analysts

True analytical capability **may** be one of the rarest commodities in the logistics world[7]. All firms and agencies engaged in logistic and cost analysis struggle to find qualified candidates for our work. A central problem appears to be that the educational process, which produces able practitioners in other fields, has nowhere been adapted properly to the needs of our profession.[8] An additional problem is the hopeless and worsening state of the literature of logistics. Having been driven almost exclusively by the US Department of Defense, once that agency lost interest in the system of military standards there seems to be no one willing to fill the resulting gap. Beyond this, there is little to say, at least in this essay, which is more concerned with the tools of the analyst, rather than the analyst himself.

Models

The models available to logisticians and cost analysts have improved greatly over the past quarter century. At the beginning of that time there were nothing more than extremely naïve methods, most commissioned by government agencies and built by model makers of no particular skill. The lack of sophistication, indeed, the simple incorrectness of many of them, led to an over long era during which their lack of impact on decision processes was a blessing. Meanwhile, a few souls struggled to found a small commercial logistic model-building industry in Israel, Sweden, the UK and the US. Today, following the defense realignments of the 1990s, these providers are beginning to make their contributions felt in an increasingly economy minded industrial sector.

The fruits of this development can be seen in two broad generations of models, with a third beginning to emerge. The first wave of models developed, for the most part, by the military departments of the US and Sweden, took two directions. The first was the development of extremely comprehensive, although naïvely-formulated models of level of repair and life cycle cost. The second, led by Dr. Craig Sherbrooke at RAND, was the very sophisticated modelling of spare stock requirements that introduced the idea of system optimization rather than single-item modelling[9]. Virtually all those

[7] Anecdotal evidence for this comes from the author's own work environment. After almost ten years attempting to be a "pure" software company, our firm has become a solutions provider, combining consulting with software to solve a variety of problems. This has been largely in reaction to noticing how much most – not all – of our customers struggle to make complete and productive use of our models.

[8] It pleases me to say that the MIRCE Akademy, for whom this monograph was originally written, is, perhaps, the only exception to this otherwise iron rule. Dr. Knezevic, both while at Exeter University and with the MIRCE Akademy, has managed to assemble both a curriculum and staff capable of delivering the basic intellectual requirements for an able analyst. Since he concentrates on mid-career professionals from industry and government as his student body, the combination has been wonderfully successful.

[9] See Sherbrooke, Craig C., "METRIC: A Multi-Echelon Technique for Recoverable Item Control,"

models, driven by the immensity of the computational problem compared to the availability of computational resources, saw operating bases as identical and hardware structures as abstract. A standard of the day was a 2-echelon, 2-indenture model in which all the bases were assumed to be identical.

The second wave of models, led by the efforts of Hans Ebenfeldt in the development of the OPUS spares model, began to distinguish between bases and be less tied to abstractly echeloned support systems. This was motivated largely be the European reality that forces were relatively small and could be modelled in detail – a virtual impossibility when modelling the USAF, which possessed huge fleets of aircraft at more than 100 bases, both in the US and overseas. The later development in the early '90s at TFD Group (fostered in part by competition, but more by the emergent computational power of desktop computers) of n-echelon and n-indenture models with different base cases and multiple system types in a single run provided the fullest expression of this kind of detailed modelling. The third wave, really only getting underway in the new century, is the introduction of simulation models led by the work of Professor Dubi (SPAR), John Millhouse and Chris Hampson (VMetric and SCO) at TFD Group and staff members of Systecon (SIMLOX).

One might also think of a fourth wave involving two specific ideas – tactical optimization in supply chain management and the introduction of the economics of time. Boeing first introduced the idea of tactical optimization, which has been implemented in SPO by MCA Solutions and by TFD Group for Northrop Grumman in SCO. The economics of time, first became explicit in the algorithmic formulations of SCO and is currently being developed further as the basis for TFD Group's multi-period spares optimization tool named TEMPO.

There is one thing to note regarding many of these commercial models as well as every one of the government produced and sanctioned models. The model builders' attitude toward data management was invariably to view it as an unimportant afterthought. The two fields, operations research and database management have never been particularly close, intellectually or emotionally. Practitioners in each field tend to view their colleagues in the other as intellectually challenged. With that as background, it is unsurprising that most software implementations of logistic and cost models lack sensitivity to the true needs of data management for their inputs and outputs. This fact has regulated much of the development effort at TFD Group for the past ten years. It will be significant in what follows.

Data

The normal situation with data is similar to the old lament, "Water, water everywhere (enough to drown in), but not a drop to drink." Most enterprises and government agencies are literally drowning in data, without the ability to make much use of it. This situation arises from a paradoxical set of attitudes. By and large, those responsible for collecting the vast storehouses of data are driven by an absolute belief that knowledge will lead to power and data must lead to knowledge. At the same time, those responsible for determining what data will be collected, how defined and how stored are almost invariably database experts, with help from decision-makers (those with the wherewithal to invest in such services). Unfortunately, the two groups together still lack the right qualifications to design and implement databases used to house data *required to run mathematical models*.

The missing expertise is that of the model-builder, who understands how the variables included in the database relate to each other mathematically. This knowledge allows him to define the content of the database correctly so that the implicit assumption of database normalization – independence of the attributes stored in the database – can be assured. This understanding is crucial to avoid implicit failure of data integrity, the lack of which causes allied models to produce conflicting results.

Operations Research, Vol. 34, pp 311-319, 1968. This was essentially a re-publication of the same article published in 1966 as a RAND memorandum.

Therefore, we logistic analysts find ourselves in an awkward position. Every client possesses a large and usually costly collection of data, yet we must plead with him that it will take most of the resources available for an analysis just to gather the data.

Even if the analysts and the models are available, they struggle to provide even a small amount of useful information to the decision-maker because of the data problem – too much, but in the wrong form, defined incorrectly, stored under the wrong keys in the wrong computers and of uneven quality.

The question, then, is whether one can change this situation for the better and if so, will it be a trivial or significant change. We believe that a profound change can be wrought in the role played by data in the analytical troika, but it requires a deeper understanding of all the elements that might contribute to a solution – and in whose absence, the solution will elude our best efforts.

Data Management

There are three concepts often confused by people when they talk about data and databases. It is crucial that these be understood as the very different things that they are. The first concept is that of a database management system or *DBMS*. This is a software package that provides a language and associated utilities that allow a user to implement the design of a database as a set of physical tables related to each other according to a set of rules. It then performs a range of useful functions that allow the user to input, view, edit and retrieve data for any purpose the user imposes on it. The second concept, the database *schema*, is the particular way the tables, relationships and attributes (table columns) are defined. For relational databases, this is normally communicated in the form of an entity-relationship model or a data model. The third element is the actual *data* stored in the database. For example, a column or attribute in the database, such as Unit Price, will have an “instance” or entry for each row, which would normally correspond with a type of part.

Taken all together, the DBMS, schema and data comprise what we normally think of as a database.

Database Management Systems

Comprehensive data management requires a number of tools. There are three basic database management functions that require different schemas. These are management information systems (MIS), data warehouses (DW) and analytical databases containing what we will call static data. A fourth concept is the Decision Support System (DSS) that combines some of these forms as well as the tools required to learn something from the data. Another distinct system type is called a Data Repository (DR). This can be a superset of the other forms, containing, for example, multiple databases, which, while each of them is itself normalized, are not related in any way to each other.

Any database management system (DBMS) is capable of performing the middle-ware data management function for any of the three forms, although typical DBMS products (Oracle®, Sybase®, SQL Server®, Access® etc.) tend to specialize in one or another of these functions.

Management Information Systems

The main function of MIS databases is to *do the business* of an organization. In logistics, we generally find these systems performing the roles of inventory management systems (IMS) and maintenance management systems (MMS) and more recently, enterprise resource planning (ERP) systems. While they play a number of significant roles, the most significant and central role played by any MIS is that of initiating and capturing data about transactions. Transactions might be a part order, receipt of a part, receipt of a demand for repair, completion of a repair, withdrawal of a repair part from stock and so on. Each transaction identifies the part(s) involved, the date and time, the type of transaction, the location and perhaps a number of other variables. Instead of parts, transactions might also involve correspondence, airline reservations, retail stock and other things.

Inherently, MIS databases are client-server. That is, the transactions may occur at widely separated locations, tied together by a local or wide area network and in recent years over the internet or even

served by the "cloud," or remote server farm. Despite their separation, however, the transactions are interdependent. For example, demands for the same part from a single depot, registered at different operating bases may have an impact on each other – the first demand empties a bin so the second generates a back order. This would not actually be known if the widely separated acts were not tied to a single central database.

More recently, certain kinds of transactions (those involving relatively small amounts of data) have been performed over the Internet. The idea here is that the only crucially important data values are the numbers of items (e.g., part # 123 or seat 3D on a particular airline flight) currently accounted in each category. That importance results from the fact that such systems allow people in different places to share resources – such as the sum of all the coach seats available on flight X. By being tied to a central database, their transactions are prevented from conflicting – usually. From time to time, you will find someone else holding a boarding card for your seat, but not very often.

Each transaction alters the state of the stock of something, raising it in one place, lowering it in another. Each transaction either initiates or closes a time-bound concept as well, initiating or bringing to a close some time-definable interval. For example, the issuance and receipt of an order are the two transactions required to identify an order and ship time (OST) or procurement lead time (PLT). The more sophisticated the transaction system, the more detail will be available. For example, OST might actually be subdivided into several components such as order processing, administrative delays, actual node-to-node shipping time, processing at each node and other elements.

The point of tracking all these transactions is very straightforward: what's crucial is to know, at any instant, just how many of everything the business still has, where it is and what condition it's in. Accordingly, as each transaction occurs, these values change and the previous values are no longer crucial to *doing the business*. A pure MIS, therefore, lacks the property of memory. That function is given over to a data warehouse.

Data Warehouses

A data warehouse captures each transaction and time-stamps it, making it possible to *remember the business*[10]. Much has been written about data warehouses. They are often confused with something called a decision support system or DSS. Because it remembers, the DW can be asked questions with time subscripts, so to speak. In retail sales, for example, the resulting data are a storehouse of interesting and not very difficult to interpret data. By simply running queries on the data, one can see such useful things as the comparative sales performance over time of different stores, regions or sales representatives. Financial data can be accessed directly and translated into the ledger entries from which financial accounts are built. These also can be interpreted in a reasonably straightforward fashion.

In terms relevant to logistic analysis, the example introduced in the previous section would be extended to data warehousing by storing the datum that on a given date, the quantity of part # 123 was reduced by 2 at location X. Each day, similar transactions are catalogued from all over the logistic system, enabling us to build a time series, one day at a time, of the quantity of part # 123 being added to or subtracted to storage bins distributed throughout the system. Depending on the form of analysis and the granularity of the models we intend to feed, these values can be aggregated and summarized by any time interval desired. Nevertheless, the granularity of the content of the data warehouse is whatever is provided by the underlying source processes – if they support minute-by-minute recording, then this is what is entered into the warehouse compendium.

[10] This short-hand description is intended to be symmetric with the similarly-formulated definitions of MIS and DSS types of database systems. Data warehouses are more conventionally thought of as places at which widely distributed data from MIS databases are brought together and organized for presentation to decision-makers.

Nevertheless, data warehousing failed to achieve its potential for many years (and still often fails) because of the dearth of "end-user tools" or tools that convert raw data into information[11]. That is, masses of data, without considerable filtering, combining and algorithmic processing have rarely provided the insight required by decision-makers. With the exception of retail sales and financial matters, these tools have largely been absent. Statistical analysis packages have provided estimation of standard statistics to the data. More recently, one hears from every side about major investment in OLAP tools (on-line analytical processing) by the database management system publishers. These tools are intended to make it possible to perform a variety of statistical and graphical analyses on large collections of data. The forms of analysis, however, are still restricted to the generic functions of statistical analysis.

In logistic analysis, the significant function performed by the data warehouse is to remember quantities and events by time period. The net result (after a good bit of analysis) is the development of many time series showing, for example, stock transactions by time, or completed time-cycles like order and ship time as each delivery is registered by a receipt transaction and compared back to its initiating demand transaction. The potential value of the data warehouse concept in the field of logistics stems from the utility of measures of this kind as inputs to our familiar analytical processes – level of repair, life cycle cost, spares optimization and the like.

It has always been a problem for logisticians to bridge the gap between the systems that handle these transactions and the models that address the end-user concerns for repair policy, cost and resource distribution.

Static Data and Analytical Databases

You will note that the focus of attention in a transaction-based data system is on the current quantities of things, whereas in a data warehouse, it is on the history of the transactions that change those quantities. In the static database, we record the values that are implied by how these transactions arrange themselves over time. For example, the analysis of a demand stream – the time series telling us how many of a part have been demanded at each location in each time period – provides part of the raw data necessary to perform demand forecasting.[12]

The demand rate (or arising rate) is, perhaps the most often looked at and studied data element among all those we classify as static. It is by no means the only one, however. What we mean by static is a variable which either has no time dimension or whose time dimension is assumed to be explained by a mathematical expression with fixed parameters, such as a Weibull distribution with fixed mean and shape parameters. The fact that we continue to agonize about what these values may be at any moment in time makes them no less static. Our never-ending concern about their true value amounts to faith that they do, in fact, have a single value (or at least a temporally stable value), but that we may not have found it; or that we may not have portrayed the underlying (fixed) complex of explanatory values correctly.

We recognize these values as the inputs to static equilibrium models. These are models whose computations assume that the state of the world described to them will persist throughout the system's remaining history. This may well entail (and in the most sophisticated models it does) dynamically changing values of some variable. On closer inspection, however, you will discover that

[11] Additional problems such as time to development, high maintenance costs, complication of business processes, data for data's sake and so on is discussed in Greenfield, Larry "The Case Against Data Warehousing," LGI Systems Incorporated, URL: http://www.dwinfocenter.org/about.html**.** This is one of a number of articles by this author and the web site is a good source for people interested in knowing more about data warehousing in general.

[12] We also need to know what has been happening with the stimuli to which we ascribe the raising of a demand. These will include various measures of system operations during the same and previous periods, the numbers and geographic distribution of systems, ambient conditions such as temperature, prevalence of dust and the like. With these elements in hand, demand forecasting attempts to predict the number of demands that will occur in the coming periods based both on inherent properties of the failing hardware and changes in planned conditions of operation and deployment.

these changes are driven by another equation in which the changing value is the dependent variable. That equation, finally, will be found to be driven by other static data elements.

The point of all this is that the increasing summarization of data that started life as nothing more than the spoor of elemental transactions ends, at the highest level of abstraction, in a static measurement of the phenomena thought to explain the attributes of those transactions (frequency, length, size etc.). Our models use these static measurements as inputs. When they are, in this final step, converted to outputs, the extent of data consolidation is extraordinary – thousands, even millions of bytes of raw data have been converted by perhaps ten or more processes, each of which may consist of tens or hundreds of thousands of computations, into the summary measures that represent information to the decision-maker.

The analytical processes defined by model makers in the logistics arena require data to be organized in a particular way. Despite all the processing just described, it is quite feasible to end the process capturing the result in a manner that is unfriendly to modelling. A single example, the conversion factor versus the duty cycle, will suffice.

One of the logistic model builder's most deeply held beliefs is that most hardware elements have an inherent propensity to fail, which is closely linked to the amount of time they are operated[13]. We note that not all the parts of a complex system operate the same amount as the system itself, being turned on and off as the system operates. Therefore, the number of hours of system operation during some calendar interval must be related to the time of operation of a given part by some variable. This measure, known as the *conversion factor* in MIL-STD-1388, allows us to convert the operating time of the system, known or planned, into operating time of individual components of the system.

However, is the conversion factor the appropriate measure to record in a database intended for logistic analysis? If we were only interested in the computation, we might say yes. We are also concerned about maintenance of the data, however. Think about an alternative measure, duty cycle, that captures the time a child operates as a proportion of the time that its immediate parent

(rather than its parent's parent's parent) operates. Imagine a part at four hardware indenture levels below that of the system. Any change, by error or design, in any of the four duty cycle values up the chain of parents of the part in question will cause its conversion factor to change. Alternatively, only the one duty cycle in error would have to be changed. This decomposition of a summary measure into its incremental components is essential to a maintainable database. We will discuss this rather innocuous sounding, but quite crucial matter at greater length below.

Decision Support Systems

A decision support system (DSS) consists of a data warehouse and end user tools intended to convert the data into information. The data warehouse brings together in a single place a concentration and summarization of data from sources that may be widely distributed in several transactional databases. Ideally, these data components would be filtered and organized in such a way as to provide insight directly or, through interpretation by end-user tools, would yield measures and graphical representations of information[14] of use to decision-makers. This ideal is rarely achieved as noted above. Perhaps the addition of OLAP tools will improve the situation to some degree, but

[13] Interestingly, evidence from the air campaign of the Gulf War suggests strongly that this is not strictly true. Multiple regression models of aircraft failure causes indicated that only 16% of the variation in failures was explained by operating hours and the remaining 84% was explained by sorties (cycles). This unexpected result explains why, despite having predicted the campaign's total flying hours with great accuracy, part consumption estimates predicted the need for twice the stock actually required: sortie lengths were, on average, twice as long as those on which the estimates were based.

[14] The distinction between information and data is crucial. The latter are available in great profusion to just about anyone in the decision making business. Unfortunately, they provide very little of the stuff needed to make decisions more wisely. The criteria for information are that 1) it is true, 2) it is pertinent and 3) it is presented in a form that is reasonably efficient for absorption into the decision process. See Butler, Robert A., "The Impact of

Technology and Organization on Command and Control," RD-104, The Assessment Group, Santa Monica, 1976.

not for the kind of specialized and complex decision processes to which logistic analysis is intended to provide support.

The role of the logistic analyst is to provide support to decision-makers concerned with systems whose complex reactions to operating and support stimuli can only be estimated through massive processing of raw data. Any analyst who has spent significant time trying to understand the significance of a MIL-STD-1388 data compendium "straight out of the box," as it were, will quickly agree that gazing at raw data is an activity more likely to produce a headache than analytical insight, much less foresight.

Luckily, the field of logistic and cost analysis is dense with mathematical models addressing many of the significant issues that arise from changes in the underlying data. These go by such names as level of repair analysis, life cycle cost analysis, failure modes and effects analysis, spares optimization and so on. They are all familiar to anyone involved in what our community calls logistic analysis. These are the end-user tools so lacking in implementations of the DSS idea in other arenas.

Since virtually all of these tools are static equilibrium models, it follows that database used to feed them must be an analytical database, populated with variables whose definition is static. A logistic decision support system (LDSS) would then consist of an appropriately constructed data warehouse connected to an analytical database to which analytical models of logistic phenomena were attached.

Data Repositories

A data repository (DR) may contain multiple normalized databases within a single DBMS instance. The purpose is to combine in a single place, disparate databases and independent tables that are, perhaps, internally consistent and deal with many of the same attributes, but come from different sources. This may be done so that they can be compared to each other, combined together for computational purposes or satisfy the needs of different communities who, for some reason, are served by the same compendium. In reality, an LDSS must be based on a data repository since the warehouse portion of the database will often hold multiple attributes defined the same, but from different sources or observed at different times.

An instance of this applied to a logistics setting, would be a database that stores MIL-STD-1388 data in its own format as well as a strictly-compliant version of the 1388 tables. This is done in order to facilitate exchange of data between the 1388 source and the alternative database. For example, the TFD Data Repository uses a strictly-compliant table set, which can be thought of as a database, to hold the content of a data exchange file and a second table set (another database) in which the 1388 keys are replaced by those of the TFD Database. In addition, the repository keeps the actual TFD Database, used as a common source database to serve analytical models and another distinct database, serving the needs of day-to-day asset management.

Another example of a data repository is that used in the JCAPS (Joint Cost and Performance System) implementation developed by Northrop Grumman Corporation for the Joint STARS aircraft. In this instance, the data repository was developed to house a wide variety of data from original USAF sources[15], time stamp and archive them, and then display these reports in the form of time series. The DR also allows *ad hoc* queries into much of its content. Many of these tables and databases are used to update the TFD Database, which is also resident within the data repository. While the latter capability probably has the greatest significance for the long-term health of the aircraft system, the ability of the repository to bring together so many disparate sources and display them in a convenient format for the USAF end user has proved a very attractive element.

[15] These include SBSS (the USAF base supply reporting system), CAMS (the USAF maintenance management reporting system), wing performance reports and internal Northrop Grumman Corporation reports.

Data Management for Logistic Analysis

In what follows, we build up a picture of a properly structured analytical database and then add on its interactions with analytical tools to form a Logistic Decision Support System or LDSS. First, we discuss the broad requirements for an analytical database, and how it is different from convention al databases. Then we discuss the detailed elements of analytical database design.

Requisites of a Logistic Analytical Database

We might ask for several capabilities in a logistic analytical database. At the most basic level, the data elements (attributes in database parlance) should be defined correctly from the viewpoint of modelling. While this seems a simple enough idea, some aspects of it must be explored. Second, the keys used to find rows or groups of rows in the tables must be well designed to cover all the ways in which fundamental ideas or metrics such as failure rates might differ between usages. Third, the way in which attribute definitions are matched to the tables in which they reside (defined by their keys) is of crucial importance to the quality of data reusability. Finally, if there are aspects of the database that can be implemented in such a way as to facilitate the maintenance of the data content (there are) we would also like to see this done.

To summarize, the database contains the data required to run analytical models. The data are collected under keys that enhance their re-usability. The data elements are defined appropriately for the modelling tasks and to ease their maintenance and analytical manipulation.

We would also like to see that the data can be moved between the database and outside data sources and uses[16], whether outside means in a foreign source not under the control of the database designer/owner or in another part of a single data repository. This process is supported by import and export functions. Finally, the database, to achieve true utility, must be capable of regular update from outside sources. This is generally achieved through the medium of a data warehouse and interface elements capable of loading the warehouse either automatically or on command.

The following discussion looks at each of these characteristics and examines how they are imparted to an analytical database.

General Database Structure

Data used for logistic analysis falls into relatively clear groupings, which require different key configurations to store correctly. One broad objective of the database design must be to distinguish between slowly changing and rapidly changing, or ephemeral, data. In the simplest terms, the former might be thought of as inputs and the latter as outputs. The reason we are concerned with this distinction is the quality of re-usability.

Figure 1 shows the broad regions of the database as blue boxes and notes that the transforming function is performed by models that manipulate the permanent data (regions above) according to a specific scenario, producing the ephemeral outputs we call project data. The content of standard LSAR databases is an unfortunate and incomplete mixture of all three of these regions.

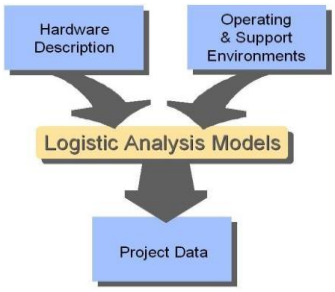

Figure 1 Regions of a Logistic

The term “ephemeral data” may be puzzling to some. We use this term for those attributes of systems that flow, not from the inherent attributes of the systems themselves, but from the way in which those systems are deployed, used and

[16] The “outside world” consists of other models or compendia that keep their data in everything from an alternatively defined database such as one based on MIL-STD-1388 or DEF STAN 00-60, to an Excel® spreadsheet or flat ASCII file.

supported. These matters – deployment, usage and support – are always subject to rapid and unpredictable change, whereas the way the systems are built and the way they react to these stimuli are relatively permanent[17].

We begin our exploration of these concepts by looking at the degrees of permanence of the attributes of a hardware system and its components. Before we begin that discussion, however, the following short section introduces a number of database terms that will be useful in understanding what follows.

Database Terms and Definitions

Relational database technology includes a number of concepts that are commonly understood, but a terminology that often sounds strange. What follows is a compact reference for the main terms that we will find useful in the rest of this paper. Anyone already well-versed in the terminology of relational database design might skip to the next section.

All the data kept in a database are kept in *tables.* A table consists of rows and columns. The columns are the *attributes* of the things identified in each row. Identification of the row is done with a *key*. A key is a special attribute (or ordered group of attributes), which makes that particular row unique among all the rows in the table. Therefore, if we know the value of the key attribute, since the key is also *indexed*, we can find the specific row we want without any searching. An index is an alphabetic and/or numerically ordered list of the content of a column in a table. While many attributes may be indexed, only one attribute (or one ordered group of attributes) may be the key.

To be able to store each datum only once in a database is to maintain the *integrity* of the data. The idea is that, if stored more than once, there is always a possibility that the two instances will have different values, leading to ambiguity or uncertainty about the attribute's true value. The desire to preserve data integrity leads, among other things, to the use of one-to-many *relationships* between tables. An example is the structure of a telephone number database. Motivated by the fact that, over time we have realized that we will never know how many phone numbers some of our friends may have, we've decided not to reserve a fixed number of fields in the table keyed by their names. Instead, we've created a second table, in which to store their numbers. There may be many numbers for a given name in the first table. The name key in table one must be *inherited* by all of the rows in table two that contain telephone numbers belonging to that name. We use an additional key (perhaps a telephone type attribute or nothing more than a sequence number) to identify which number we want to retrieve.

In the example above, we suggested that table two might be keyed by the type of telephone number – fax, cell, home, office etc. A moment's reflection illustrates why this is actually a bad idea. What if our friend decides to put in two faxes at home, or decides to buy a second cell phone? What if they move and their home number changes? This is an illustration of the technical arguments against the use of *intelligent keys*. An intelligent key is one that contains information other than that required to find the single row in which its data reside. Database designers try to stay away from intelligent keys whenever possible because of these pitfalls. The most prominent such error in the logistics community is the LCN (logistic control number). Used throughout the community as the most significant key in all logistic data compendia, it causes endless troubles because of the fact that it also carries indenture information.

[17] "Relatively permanent" because almost nothing fails to change. In particular, once a system reaches the field, it begins to grow apart from all the other systems defined, initially, in the same way. Slowly, the bill of materials changes as the system undergoes modifications. As different parts fail at different random intervals (despite sharing a common mean expected interval), the resulting renewal process engendered by repairs causes each system to have a cumulatively different effective age with the result that its operating and support characteristics also grow distinct from other members of its class.

Characterization of Parts

We start from the bias that all description of hardware systems will be simplified and made more intelligible by representing them as strictly arborescent hierarchies[18]. The system or end item is at the top of the structure and everything else is underneath it, arranged in a manner to illustrate what goes into what, into what, *ad infinitum*. The practical implementation of a hierarchy, where many layers can occur and unknown numbers of members reside at each layer, is an indentured parts list. The alternative forms are a parts list, which has only one entry for each part type in a system and an un-indented parts list with indenture information in it[19].

An indentured parts list is nothing more than a parts list in which a parent-child relationship is *illustrated* through the device of indenting the child under the parent. Creating such a list is a matter of preserving the correct sequence of parts, together with the number of indents appropriate at each point in the list. Provisioners will note that such information is contained in the PLISN (sequence number) and indenture code of the LSAR 036 report. They will also note the opportunity for error or ambiguity introduced by the inclusion, in the same report, of the LCN, which is intended to provide the same information, but often doesn't – exactly the kind of data integrity failure the use of relational databases is intended to avoid.

Suppose the attributes that appear next to items in a table whose key denotes position in an indentured parts list describe component identification, reliability, cost, maintainability and so on. Two things should be noted about such a table: 1) the list may well contain different instances (usually called appearances) of the same part and 2) adjacent rows may hold data for very different types of components (for example, the system and an LRU). These observations raise two corresponding issues: 1) how to preserve data integrity in the identification attribute, since it may occur more than once and 2) how to describe very different types of parts in the same table.

The first issue, data integrity of part identification information, leads to the first departure from standard logistic practice as described by documents such as MIL-STD-1388-2A/B. Since, if we wish to preserve data integrity, we cannot define the identifying characteristics more than once, in fact, *we cannot use them at all in an indentured parts list*[20]. Accordingly, part identification material – in fact, everything about the part that is inherent to it and does not change from application to application – must be stored in a simpler table called a parts list. This table contains only one entry for every part type, along with all its other attributes that are otherwise invariant. The key assigned to each part (remember that we are not going to use intelligent keys, so this one is nothing but a sequence number attached to the name of the table itself) is passed on to, or inherited by, the indentured parts list table, possibly many times. When a view is assembled for the database user, the readable part identification data are retrieved from the parts list and shown in the indentured parts list at each application of the part in question.

The second issue, how to describe very different types of parts in the same table, yields to similar logic. Adjacent rows in an indentured parts list may contain the system, a subsystem, a line replaceable unit, a shop replaceable unit or a piece part like a bolt or gasket. While parts of these different classes may have different attributes because of the role they play in a system, they also have a common pool of equivalent attributes, which are subject to being stored in the same table.

---

[18] This is, in fact, nothing more than the author's prejudice. It appears to be shared widely in the logistics community, but has only recently gained a place of prominence in software applications. Note, for example, the apparent reluctance of the LSAR software community (1388 and 00-60) to introduce methods of actually viewing the parts in their hierarchical positions, rather than as long lists of parts whose logical relationship to each other is hard to decipher.

[19] Indenturing information can come in three basic forms. The first is parent-child data, using identification of the next higher assembly as the means of building the bill of materials. The second is a part sequence with indenture information for each sequence number, which is the form taken by PLISN and indenture code information. The third method is a code number that effectively lists the entire sequence of parents from the part in question all the way to the top of the hardware hierarchy, the best known example of which is the Logistic Control Number.

[20] Recall that we are describing the means of data storage, not data viewing.

Much of the goodness of design of a logistic database has to do with emphasizing the sameness of attributes (limiting the number of key collections and tables necessary to hold the data) and redefining variables so as to minimize the number of keys needed to retrieve them.

By extension, the cataloging of different part-type attributes in the same table, no matter where they reside in a hierarchy means that few distinctions between parts are appropriate. We might summarize this in the statement, *a part is a part is a part.* Many modelling efforts have been weakened by intense concentration on naming hierarchies, with the associated attempt to draw hard and fast distinctions between these named elements[21]. For example, what is the difference between an SRU (shop-replaceable unit) and a sub-SRU? Nothing inherent, to be sure.

By contrast, there are certainly inherent distinctions to be made in the *role* played by a part in a certain appearance in a system. For example, an LRU (removed and replaced directly on the system or, for airplanes, at the flight line, hence the L in LRU) in one instance may well be an SRU in another, or indeed in the same appearance, but different maintenance event. The role may well be significant in the model builder's treatment of the part even though *the part's attributes are always the same whatever the role*.

The role names for parts and what distinguishes their behaviour from the viewpoint of the model builder are as shown in Table 1. There are other names that indicate groups of part attributes (or lack of them).

Role	Identification	Behaviour	Measure of Effectiveness
System	Top item in hierarchy	Operated by a crew	A_o
Equipment	Provides a measurable stream of utility	Operated by crew or system according to operating profile	A_o
Logical Assembly	Functional name	None	Reliability, cost
LRU	First removable (physical) assembly in a branch of the hierarchy	Operated by system or equipment according to duty cycle	Fill or other stock MOE, reliability, cost and other engineering parameters
SRU	Below the first removable assembly in a branch of the hierarchy	Operated by system or equipment according to duty cycle	Fill or other stock MOE, reliability, cost and other engineering parameters

Table 1: Classes of Parts

For example, a consumable is understood to be something for which no maintainability data are required. But any of the groups distinguished below could be a consumable. Therefore, adopting that particular taxonomy compels us to treat "consumability" as an attribute, rather than a part-type.

These hardware component classes are all that we need to build complex systems. It is also useful to note that the same naming rules can be used for virtually all the other resources dealt with by the logistician. These include tools[22], skilled labor, facilities, software modules, commodities such as

[21] Not excepting the author's own efforts. The first several versions of the EDCAS model drew hard and fast distinctions between LRUs (line-replaceable units) and SRUs, ascribing different attributes to them. This caused endless troubles in the organization, maintenance and retrieval of data describing such assemblies. Once the similarities were recognized, the database, organized along those lines, became quite easy to manipulate and the model itself became more rational.

[22] The generic name, tool, can be subdivided into a number of very different classes. The main ones are common tools, support equipment, aerospace ground equipment (AGE or "yellow gear") and what we call "big tools." The latter include such things as test stands, anechoic chambers, clean rooms and the like. The model builder will also want to know if a given tool

paint, solvents, POL and rags, training courses and data. With the exception of piece parts, commodities and skilled labor, all these resources share the characteristic of parts that they can be portrayed in terms of hierarchies. A database that treats them in the same manner can therefore also portray their hierarchies in the same manner.

For any given part, some of its attributes never change no matter what, others are a function of its membership in an assembly and still others are a function of its position in an entire system[23]. We will call these attributes part, configuration and system attributes, respectively, all of them referring to different characteristics of the same part.

To illustrate, a *part attribute* might be its weight or reference number. A *configuration attribute* of the part might be its duty cycle when a member of a specific parent (different when a member of a different parent). A *system attribute* of the part might be its inherent failure rate at a specific appearance in a system, different from its failure rate at other locations in the system.

From the discussion of keys above, it will be clear that the key structures of each of these groups of attributes are quite different. Part attributes have a single key representing the part itself. This is the simple parts list referred to in the discussion above. Configuration attributes carry the key of the part as well as the key of the parent or next higher assembly. System attributes have the key of the part together with a complex key representing the entire sequence of parents from its present location all the way to the key of the system itself[24].

You will appreciate that the more keys, the more columns in the database and hence the more data required to fill each row, imposing a heavier burden of database maintenance as well as a threat to data integrity. Hence, the first important rule of database construction (for logistic analysis to be sure, but perhaps as a more general rule) is to *push as much data as possible to the tables with the least number of keys.* We call that pushing the data *uphill* in the key structure.

"Pushing the data," means correctly classifying data elements as well as correctly defining them.

The example we have been using, part identification, is often tempting because doing so renders it unnecessary to look up the reference number elsewhere when looking at the system table[25]. Unfortunately, as we have seen, it also destroys data integrity by making it possible to have different values in each of those locations – and imposes the additional burden of making the user enter the value over and over again. Of course, the argument can be made that these problems can be conquered in a de-normalized database, but doing so relies on a *de facto* normalized database and the use of triggers and rollback mechanisms that may not always work correctly. The correct approach is to push the reference number up to the part table containing the single key and then display that attribute wherever needed by the use of an SQL query that draws it from that single

---

is part-specific, but this can be determined, within a system or database, by the existence, or lack thereof, of links to maintenance events of which other parts are the subject. In other words, part specificity need not be an inherent property of a tool, but can be determined by inspection. In acquisition analysis, on the other hand, when tools may well be the object of design, it is often useful to include part specificity as an attribute, since it can have a significant influence on costs and support policy assignment (level of repair results).

[23] Later, we will also see that some of these attributes are subject to further variation as a result of their deployment and usage conditions.

[24] Either as a sequence number or as an intelligent key. The sequence number would make it necessary to visit the configuration table and then the part table over and over again until the entire sequence of parents is recovered. In practice, this is done with a moderately simple SQL query. The alternative is an intelligent key like the LCN, the disadvantages of which have already been discussed.

[25] Some years ago the author learned, in conversation with the author of the original MIL-STD-1388-2A tables, that one of the objectives of this design was to make it unnecessary for program managers who purchased LSAR data collections to also purchase software to make their content comprehensible. Hence, each table had re-entered into it the human-perceptible identification of the parts it described. Oddly enough, this was not too bad an idea until computers entered the picture. Once we started reading these tables digitally, however, typographical errors (which abound in any such compendium) are raised to a new level of significance since the computing machinery cannot exercise so high a level of fuzzy logic as a human being does, routinely, in deciphering what is correct and what is not.

source. By doing so, a change or correction in the reference number corrects every use of the value throughout the database and software applications without further effort.

The most compelling illustration of the value of pushing the data uphill actually requires the redefinition of a very common data element – the *conversion factor* of MIL-STD-1388. The conversion factor is the proportion of the system's operating time during which a component operates. In Figure 2, we illustrate a simple hardware breakdown structure in which the assembly ABC-123 occurs twice. Conversion factors are illustrated by the lines to the upper right linking the two appearances of the component ABC-123-X directly to the system. Duty cycles (defined as the proportion of a parent's operating time that the child operates) are illustrated on the lines at left linking each parent and child. The chain product of all duty cycles up to the system equals the conversion factor. The issue is which of these values should a database store? The answer should become clear if one considers the problem that the duty cycle between the System and part ABC, .9, may have been changed or found to be incorrect. In a database that stores conversion factors, all the values in the tree of ABC (say, 2,000 of them corresponding to all the parts in ABC not shown in the illustration) will have to be

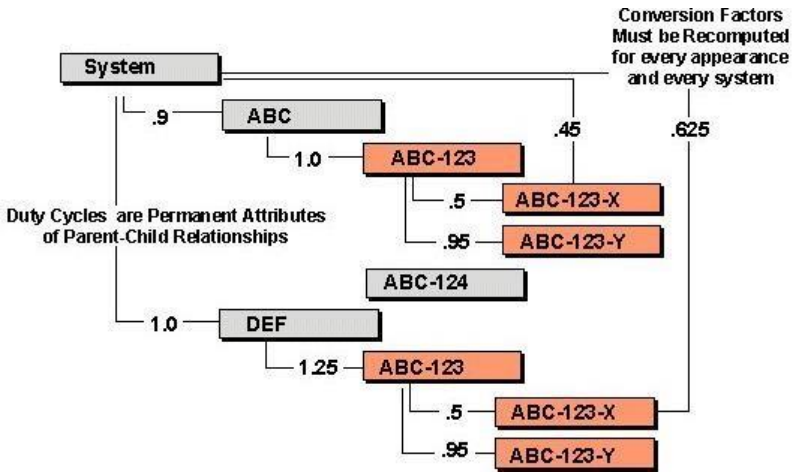

Figure 2: Conversion Factor and Duty Cycle

recomputed and re-entered. In a database containing duty cycles,

only the .9 would have to be changed. Notice also, an additional labour-saving value of the configuration table. A multiple appearance assembly like ABC-123 needs only be built once. If it changes, it needs only to be corrected once also. Without a configuration table it would have to be re-built every time it occurred and corrected everywhere it occurred.

Data Integrity and Attribute Independence

The assurance that a datum is correct and that nowhere else in the database will it have a different value is of some considerable value.[26] Without that characteristic, called data integrity, the user can easily find himself in a situation where complementary analytical processes provide conflicting answers. This leads to loss of confidence in the results, in the database and in the analytical tools. But preserving data integrity in an analytical environment is not so easy. The problem is that data integrity, as dealt with by professional database designers, is only half of the story. The other half of the story concerns the implicit assumption, rarely addressed, of independence among the attributes held by the database. This characteristic is quite properly assumed for most business databases. In an analytical database, however, it is inevitable that many of the data elements are numeric and that they are related to one another through equation systems.

The relationships between mathematically related variables are resolved (i.e., normalized) in the "reduced form" of an equation system. The reduced form is a set of equations that separates or

[26] A non-key data element or attribute. Key values are repeated from table to table as necessary to provide the relations that are so crucial to the relational database. This repetition of key values gives rise to the rule, no intelligent keys (keys containing data of interest to the user). If keys are nothing more than auto-generated sequence numbers, they will not be subject to being given different values in their multiple appearances in the database.

factors all the basic variables and produces the simplest expression of their relationship. Unless the database designer has developed something akin to the reduced form of all the equations that relate the various attributes stored in the database, his design will suffer from lack of data integrity.

The issue of mathematical independence is problematic because logistic analysis databases will always feature a rich mixture of inputs and outputs. In fact, because the modelling processes we identify stretch from the very conception of a new system's design all the way to the point of disposal, many of the attributes in any database are both inputs to one process and outputs of another.

It should be noted that the physical implementation of most commercial or business databases are carefully de-normalized. This is because observance of strict data integrity (called the third normal form of the database) imposes such burdens of inconvenience on the user that the same value will, in fact, be stored in a number of places. Internal triggers and a process called rollback are used to protect against the dangers of this type of de-normalization.

The sacrifice of data integrity in exchange for performance improvements in the operation of the database is a considered trade-off performed by highly specialized database experts. It is also unjustified in the case of logistic analytical databases. The advantage we have, that tips the balance of the trade-off in favour of strictly enforcing data integrity, is that the analytical database does not lie at the heart of an operational (do the business) scenario. It is used for planning and analysis; activities that are normally pursued off-line. The only database performance issues that arise in an analytical database have to do with making the operation of models fast enough to be convenient for the analyst. This problem can be handled in a variety of ways, none of which require alteration of the fundamental storage design of the database.[27]

Data integrity, then, is a necessary, but not sufficient condition for a sound analytical database. The need for data independence is illustrated by the following situation.

> *A database is designed in which the groups of variables **A** and **B** are stored. The design is carried out carefully and all the members of **A** and **B** are recorded only one time, preserving data integrity. Unfortunately, when analysts are asked to update the calculated values of **A**, they appear to be unable to do so, or their work takes far longer than one would expect, now that they have the database at their disposal. The explanation lies in the fact that a set of relationships **A = B + C** exist, but were not taken account of in the design of the database. Two problems arise. First, the values in **C** were never included in the database because the designers were unfamiliar with the modelling requirements of the discipline[28] Second, while on the surface, data integrity was achieved, it was lost in reality because **B** is stored twice – once as **B** and once as part of **A**. The ambiguity we tried to avoid by using the relational model remained because of a technical relationship that made the assumption of attribute independence untrue.*

[27] The data required by an analytical process is typically extracted from the database in the form of a query. If a part of the analytical processing will require a particularly slow query, it can be extracted beforehand and stored as a separate table – to which the user has no direct access. If the user takes an action that would cause the table to be altered, background processing can be used to update it. In any event, even a loss of data integrity, known to be

[28] Some attention was paid, as an afterthought, to this problem with the publication of MIL-STD-1388-2B, which included an expanded set of "modelling data" in the A tables of that standard. Unfortunately, the authors overlooked large areas of significant input data requirements despite having their attention called to the problem. For example, in the H tables, supposedly addressing everything necessary to the proper treatment of provisioning (ranging and scaling), the sole entries treating deployment and support structure are repair cycle times at three echelons of maintenance – all portrayed at program maturity. This despite the fact that many European agencies and even a few US defense establishments had already discovered the inadequacy of support structures represented as if all bases or all intermediate sites could be portrayed as having the same attributes. More recently, many modelers have turned their attention to time-phased optimization processes whose inputs are completely absent from all the standards. In a larger sense the oversight indicated by this aspect of faulty database design appears to occur everywhere. The literature of database design is silent on the issue of modelling and the reduced form of complex variables.

The problem of independence is what makes the conversion factor a difficult data element, as illustrated at Figure 2, above. It is nothing more than a chain product of all the duty cycles along the chain of parents of the part in question to the top of the system. Note the same problem in another type of variable. Reliability prediction frequently uses a metric called the junction temperature to help determine the failure characteristics of an electronic component. If stored as junction temperature, exactly the same problem arises as shown for conversion factor. If, instead, however, the system value is redefined as the configuration value, rise in temperature (between parent and child), the same set of problems goes away. Here, it is even more significant, however, because the system itself may well travel through a variety of ambient temperatures, each one affecting all the junction temperatures below in the structure[29]. isolated in the query table, poses little problem since the results of the analytical run are rarely mission-critical. If they are (say, the final run of a spares optimization model) the sensible analyst will take the time to extract a fresh query in any case, thus removing the possibility of an integrity loss.

Other examples of reduced form manipulation of data elements are also crucial to truly "common source" databases[30]. These include the need to decompose a composite concept such as the mean time between maintenance actions (MTBMA) into its components, the failure rate, MTTR and the scheduled maintenance rate and average duration of scheduled maintenance events. This is crucial to the consistency of result when, for example, one model uses failure rates and another the more comprehensive term, arisings. If the two values have been stored separately, there is no reason why one would expect consistency of the failure rate core of arisings with the actual failure rate stored elsewhere. If instead, the concept arisings is re-constituted from its several components, among which is the failure rate, whenever needed, then it and the models that use it will remain consistent with other models that use the more fundamental concept of the failure rate.

Data Sharing and the Import-Export Process

Virtually all logistic applications have defined some form of database in which to store data for their own use. In addition to serving the needs of the model itself, the database must be capable of communicating with other sources and uses of data (the "outside world," if you will). The ability to do so effectively depends on three issues. The first issue is consistency of definition of internal data elements with those of the outside world. The second issue is the choice between interface and integration. The third issue is the strategy adopted to forge links to the outside world.

Data Definition Issues

The clever model builder will attempt to define the software implementation of his models in terms that are familiar to his target user and that are readily available in recognized and standard sources. Not to do so places a difficult burden on the user and is unlikely to be a successful commercial strategy[31].

[29] Both of these examples illustrate one of the more difficult problems preventing advancement of logistic practice. Neither temperature rise nor duty cycle are defined in the LSAR world. In fact, that world does not use the 2-indenture device of a parent-child relationship. As a consequence, when database experts, even including experienced logisticians, set out to design the database, problems of the sort raised here were not on anyone's radar. Even when the problem was clearly understood by the designers of one MIL-STD-1388-2B system, they elected not to use the superior data devices like duty cycle. The reason, probably quite appropriate at the time, was their fear that the government agency responsible for validating the compliance of their product with the standard would misunderstand the device and withhold validation.

[30] The term common source database achieved great currency, especially in the UK a few years ago. Unfortunately, those who spoke the loudest had little idea what was actually required to make one work.

[31] I speak from hard experience. An early version of one of my own models, a spares optimization model, had its inputs defined in the manner in which they were discussed technically. Thus, a primary input was demand rate. After receiving the new, improved version of the model, one user called with an irate complaint: she understood none of the names of the variables and could not figure out how to use the model. We quickly withdrew the improved version and went to some considerable trouble to redefine concepts so that they were familiar to the rather large provisioning community rather than the small-to-disappearing model builder community.

Presentation of the data to the user does not necessarily control presentation of the same data to the model, however. That is, the user may well see variable names with which he is familiar, but which are inappropriate for use by the algorithms of the model. We distinguish, in fact, three different forms in which the data may be treated. These forms are the external, user interface and internal forms. The external form of data is what occurs in the world and is completely outside the control of the model builder[32]. The user interface form of data is what the user of the software sees, which may have been converted during import from some other, less useful or convenient form. The internal data form can be a further transformation of the data concepts defined for the user's benefit to those required to drive the model itself.

In developing the algorithms of the model, the first thing to try is to define them in terms of the external form of data. There are several reasons for this, not least of which is that, by doing so, one at least insures that data can be made available to the model[33]. Failing this – and such failure occurs more often than not – the next best thing to do is to define the model's variables in terms that can be derived from external variables. That is, the model builder must first develop a set of algorithms that translate externally available data into the internal variables required by his model. Failing this, a state that occurs with some frequency when we build models that break new ground, the only expedient is to use the variable required by the theory, explain it as well as can be done in the user interface and provide the user the means for guessing about the range of values it might actually have.

The third expedient is important, because it makes it possible for modelers to explore new territory; territory known to be theoretically significant, but as yet unaccepted in the practitioner's community being served. An example of this arises in spares optimization. The logistics community has taken a very long time to migrate, at least partially, from the use of fill rates to the use of operational availability as a favoured measure of effectiveness. The author suggested some years ago that a further migration was necessary before spares solutions would be truly efficient. The suggestion was to use maximization of profit or net public benefit as the direct measure of effectiveness. The argument was that profit or public benefit was the clear objective of anyone charged with working out the spares solution, and that its maximization would never be coincident with a given availability target, except by wild chance.

The problem with using a concept like net benefit is to define the value of the services delivered by a system in order to be able to compare it with the cost of increasing the flow of those services through the stock-piling of more spare parts. This is easy if the system is of a commercial form, such as a power plant. It is at least feasible for such equipment as a commercial airliner[34]. It poses real problems when one is considering a tank or a fighter plane.

The concept, missing from discussions of the cost of public goods, is that of Value of Productive Capacity or VPC. This term was coined to describe the theoretical value of the services derived from

[32] Would that this were not so! It is a persistently recurring daydream of any model builder that the world would capture and document the data elements he prefers to use in his models, rather than those he finds in readily available sources. It is, nevertheless, no more than a daydream.

[33] In an early critique of extant models prepared for the U. S. Navy, the author found that a large percentage of the data elements prescribed for the group of models then being prescribed for use in acquisition studies were not among those we are calling external. That is, they would have to be measured independently by the user of the model. A smaller, but still significant percentage of the variables defined for these models fell into a category labelled "un-knowable." That is, not only did they not appear in any known source, it defeated the author's imagination how one might go about measuring variables so defined. This was sufficiently ludicrous that it called to mind the cartoon showing a professor writing voluminous equations on a blackboard, stopping at a certain point when he couldn't complete the derivation he was seeking and finally, giving up, wrote "and here a miracle occurs." See Butler, Robert A., "Guidelines for Hardware/Manpower Cost Analysis" (with T. Neches), AG-PR- A100-2, The Assessment Group, Santa Monica, 1978.

[34] United Airlines was the only airline among a dozen or so with which we became familiar, that actually used a related concept. They defined total costs as the some of the cost of downtime and the cost of inventory necessary to reduce it. This very useful approach provides a 1/y curve for cost of downtime and an exponentially increasing cost of inventory curve, the vertical sum of which is a U-shaped curve, featuring a unique minimum total cost.

a system over its entire useful life if it were available for operation every minute of that time. The net benefit approach to spares optimization simply looks at the increasing marginal cost of inventory and compares it to the decreasing marginal benefit (VPC times the operational availability rate achieved at each inventory level). Despite the virtual "un-knowability" of VPC in certain applications, this concept was included in the spares optimization tool, VMetric, because it could be used – backwards, as it were – to determine what VPC is implied by a given availability target. The value of such a computational procedure derives from the ability it imparts to measure relative levels of implied value between systems meant to accomplish similar goals or provide similar utility.

The problems inherent in exchanging data with the outside world will be a direct function of the degree to which the model builder can make use of commonly available data sources. As the demands of modelling correctness lead him away from those readily available concepts they will increase the amount effort required to convert external data to the user interface form and thence to the internal form. Nevertheless, the model builder will almost invariably have to perform some level of transformation, simply because the development of modelling techniques and concepts is more dynamic than the development of data sources. An illustration of that unfortunate truth is the defined content of the MIL-STD-1388 LSAR 036 Report. This report, modelled on the older MIL-STD-1552A, describes a range of variables thought to be sufficient to accomplish all the tasks required in provisioning. However, these standards were developed in a single item modelling world and very little of their content is useful for modern spares optimization techniques. In fact, a great deal more data are required to apply these techniques in a way that captures their full value.

Interface versus Integration

Two terms are often used indiscriminately when discussing the movement of data between a software application and the outside world. These are *interface* and *integrate*. While the terms are subject to alternative meanings in various contexts, most computer professionals have adopted meanings that make interface a distant or arms length link between source and target and integrate to mean a tighter link, more opaque to the user. For example, interface implies extracting data from the source and placing it in an intermediate medium with a neutral format such as a flat ASCII file (the method used by MIL-STD-1388), a spreadsheet or a database. The target either reads the intermediate medium directly or converts it to an appropriate format for import. Integration implies that the target can make use of the source's data *in situ* without moving it from one database to another – that is, the target annexes the source database.

To integrate source and target requires full knowledge of both. For this reason integration is very difficult (i.e., costly) to achieve, unless a single agent has control over both. It also implies a certain interdependence between source and target, which may be quite inappropriate for logistic analytical applications. Consider that logistic modelling for decision support consists of asking hypothetical questions and advising the decision-maker of the answers. This is done so that he will understand how to react to a wide variety of potential situations, *none of which actually pertain*. That is, the modelling process often deliberately introduces untrue data into its database in order to study situations that might arise, rather than the current situation. As a result, integration between a management information system doing the business of an enterprise and a modelling system meant to improve the business by exploring different ways of doing things (without actually having to do them) must be very carefully done.

The virtue of integration is that the user of the system is generally unaware of its workings. Actions required to utilize an interface are unnecessary since the communication of data back and forth is done at the database level. This is a virtue, of course, only so long as those hidden actions are consistent with the user's desires. If not, the user has no recourse to prevent or modify the communication.[35] No matter how much integration is practiced, however, there will always be a need

[35] An interesting example of such a problem arose in the author's experience when implementing the first instance of the TFD Database. In that application, there were 30 action officers sharing a central database. Each

for obtaining data from the outside world through interface – even for enterprise resource planning (ERP) systems.

Interface Strategy

The requirements and difficulties of data sharing will be quickly recognized if you construct an interface diagram such as the one shown in Figure 3. It illustrates what is needed to move data between seven different sources and uses of data.

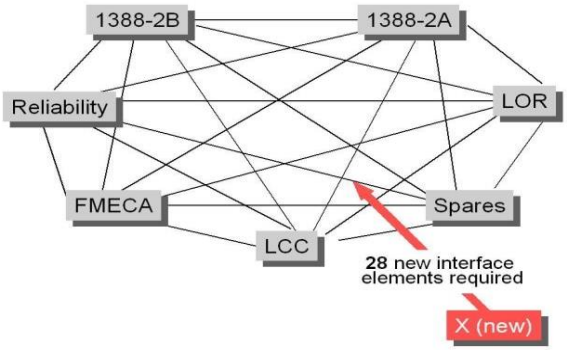

Figure 3: Conventional Interface Strategy

Each line represents four software modules, one for import and another for export at each end of the line. There are 42 lines representing a total of 168 modules, some of which, like the line between the two 1388 standards, represent very significant code modules. The even more unhappy news comes when one considers what happens when, for example, a new LSAR standard (say, DEF STAN 00-60) is defined. To link the new source or use of data into this system will require 28 new code modules. The one after that will require 32 more and so on. In other words, this interface strategy is inherently non-scalable.

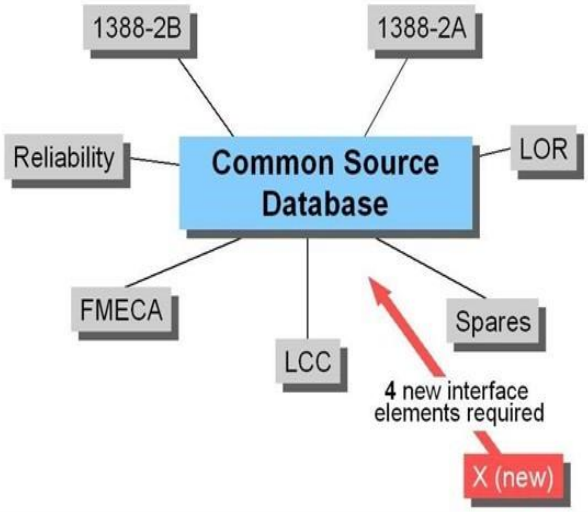

Figure 4: Common Source Database Interface Strategy

More than non-scalable, it is not maintainable, either. Consider what happens when you wish to release a new version of one of the applications. Every link to the other sources and uses must be re-programmed if the data definitions or layout of the new version has been changed.
The alternative is actually quite straightforward and exhibits qualities of both maintainability and scalability. As illustrated in Figure 4, it relies on the use of a correctly defined common source database shared by all the sources and uses of data. The snarl of interface lines is gone.

The number of interface lines now equals the number of source and use components and the number of modules is four per component. Adding or changing existing components requires much less work. Moreover, as the group officer was assigned a different aircraft type. Nevertheless, in many instances, different users were maintaining the data for the same parts, since those parts were common between different aircraft. Since, prior to the introduction of TFDdB, each user had been using an independent database, it came as a rude shock that someone other than himself might be altering the data describing his aircraft. While in the past, data could be shared – or blocked – through an interface process, the new system using an integrated and normalized database meant that there was only one description of any given part. Once the process was understood this aspect was gradually accepted as a virtue. At first, however, it appeared to be a significant problem.

of sources and models is expanded, the ratio of four interface modules per component never changes. The system is, therefore, both maintainable and scalable. This is the basic architecture for a logistic decision support system (LDSS).

While the finished product makes import and export of data much more convenient than the alternative, the simplicity suggested by Figure 4 does not come easily. The problem is that the common source database must be designed properly, not just achieving normalization in the database itself, but taking all the steps necessary to achieve attribute independence as well. Since the content of the database includes a large number of numeric values representing inputs, outputs and sometimes both, all the algorithmic links between them must be determined, formalized and used to dismember calculated values into their independent elements.

Database Updating

The diagram in Figure 4, with links to the outside world, illustrates the main elements of an LDSS. It provides the data and the means to manipulate that data to provide the decision-maker with the information he needs to do his job intelligently. What happens, though, when the data first loaded into the database become dated and incorrect? What will the update methods be?

Conventional Updating Methods

Unless all changes to the data will occur as a result of the user's manipulation via either data maintenance forms or the applications' user interfaces, some data will have to be brought into the database from external sources. The separation of data-creating processes from analytical data using processes characterizes in-service or running systems. The problem is not so prevalent for acquisition systems because there is no business to do yet. The business during acquisition is creation of the system itself and the tools very often (at least this has been our objective since the late 1970's) are used in the creation of those systems. That is, during the acquisition process, data first enters the world through the user interface of the tools later confined to the LDSS.

In almost every in-service case, then, data will be imported to the LDSS from the outside world. Moreover, this will also be true in the acquisition process when a new user of a tool has begun his work in another way. The most prevalent use of import mechanisms will be the initial loading of data. Thereafter, however, in order to make the whole system useful, it is crucial to provide for frequent and low cost updates.

Notice in the diagrams above that in the conventional interface methodology, there is no single source to update. Instead, data must work their way around the applications in an uncontrolled way. This can create endless problems with inconsistencies between departments, applications, disciplines and, unfortunately, decisions. In the case of the common source database, however, it is clear that external links would be made to the database only. Having linked to it, a source is now, by extension, linked to every source or use of data tied into it with no additional effort. Consistency is maintained through the simple expedient that only one instance of any data element is stored in the database (data integrity).

The real question, for the on-going utility of the LDSS, is how to tap into the constantly changing primary sources of data.

The conventional answer to the updating problem is to produce interface software to standard outside sources and then trigger their operation periodically. In practice, a source is usually known to produce an updated output report at a given interval. These are detected, either automatically or by the database administrator, and the import process is triggered. It is unlikely that all the data required by the database will be found in such sources, so analysts will also find themselves updating other fields not found in standard, accessible sources.

Bulk Loading versus Updating

The interface strategies discussed above pertain to updating as well as initial loading. Note that there is a difference between one-way and two-way interfaces and a further difference between bulk loading devices and update devices. Although much of the intelligence in a one-way interface can be used to support the reverse flow of data, not all of it can. That is, more software must be written to

support the reverse flow. Similarly, the development of an updating interface tools is far more demanding than that of a bulk loader.

A bulk loader is an interface that can acquire data from an outside source and write it into an unpopulated database – or conversely. An updating interface must have considerably more intelligence built in. Upon import of external data to an already-populated database, the first step is to identify entities in the source that already exist in the target.

If any of the data defining the (missing) are different between source and target, there must be a reasonable method for resolution. The most common, and not very useful, method is a comparison of the vintage of the target and source records, the youngest one being accepted as correct. It is easy to devise scenarios in which this is untrue. For example, two people, *A* and *B*, work on the design of the same part at different points in time. *A* does his work, but leaves the file open for a week. During that week *B* retrieves data from the archive, makes a change and saves it right away. *A* then saves his work, overwriting the changes introduced by *B*.

Other expedients are database checkout systems, table locking and record locking. Each of these merely enforces time discipline without the need to compare the time stamps. A checkout system effectively locks all users out of the database or part of it while anyone is using (changing) it. This brute force method is not suitable for databases with fast-changing data. Table and record locking methods are more specific, but are really only suitable for on-line multi-user systems.

While these approaches prevent the errors cited in the previous paragraph, they don't address the more significant problems arising from updates coming from outside the system.

Data from outside the database have been created by largely unknown and un-controlled data gathering processes. Perhaps one of the most difficult problems faced by the developers of database management systems is how to reconcile the vast de-normalization problem arising from replication of a database, allowing different individuals or groups of people to work on it without the need to use the locking mechanisms described above. For the most part, the only method allowed is a one-way process. That is, data from the replica is never returned to the master database.

Updating Using Data Quality Attributes

Both the problem of resolving conflicts between simultaneous on-line users and of updating the master from replicas can be addressed with an alternative method based on something called a data quality attribute or DQA. A DQA is a one-byte number (0-255) associated with every cell in a database, which describes the relative quality of the process that created the value now residing in that cell. It was invented to resolve the difficulties discovered in a data synthesis system called SDU (System Design Utility).

SDU was a software product used to visualize hardware hierarchies and fill them with

"synthetic" data generated by a variety of top-down allocation processes. For example, a target failure rate could be ascribed to a system and distributed among the elements of its proposed breakdown structure. The problem arose that, if the analyst had a better failure rate for a component than the allocated value, he had to be able to protect the value he entered from being changed by another allocation. In other words, the system had to understand that data entered through the keyboard should not be overwritten by data generated in the allocation process. A generalization of this concept became the (single) rule of hierarchical access or edit control: *no data-creating process may overwrite any data element whose DQA is greater than that of the process.* Every method by which data could be created was assigned a DQA to characterize the relative accuracy of that data-creating method. Every datum stored in the database was accompanied by the DQA that had been assigned to its method of creation.

Some examples of DQAs and their meaning are given in Table 2.

Table 2: Data Quality Attribute Definitions

DQA	Meaning
0	Null (no data)
1	Default value
2	Allocated value
3	Reserved
4	Keyboard entry
5	Special Authority
6	Computed from keyboard entries

In an administered network, the database administrator has the job of assigning DQA authority to users. Note that the use of the DQA allows setting controls more finely than usual – tables, columns, rows and all the way down to individual cells, if desired.

In practice, the administrator assigns a higher-than-keyboard DQA authority to a specialist whose area of expertise is reflected by specific columns in the database. When he alters entries in those columns, the new value will have the DQA assigned to him (usually 5), while all other values in his copy of the database will have been set to the reserved value, 3, which is intentionally lower than the keyboard default value, 4. This device makes it possible to update the master with his authorized changes, which will overwrite any other data found there, but none of the changes he makes in other columns will overwrite anything but default or allocated values.

It is noteworthy that DQAs provide the only method ever suggested capable of allowing remerging of replicas with the master without the risk of overwriting good data with bad. Moreover, the method does not rely on the review and intercession of a data administrator.

Data Transformation in Decision Support Systems

In most sophisticated organizations that use management information systems, these systems are normally running in tandem with some form of data warehouse. The warehouse, in turn, is intended to form the core of a decision support system. Transactions data are archived in some form, usually in the form of time series, then subjected to a variety of analytical processes. These almost always include financial analysis and very often they also address demand forecasting for use in ordering. These “data concentrating” functions are intended to make the data more useful to the end user. Similarly straight-forward analytical interpretations are available to such databases when used in conjunction with retail sales and manufacturing processes.

The archiving function may be doing a good job or a poor job of constructing time series data pertinent to some of the forecasting processes the logistician wishes to use. In fact, the data warehouse, constructed for different purposes entirely, may be doing very damaging things. For example, the demand forecasting methods used in most inventory management systems leave much to be desired[36]. The forecasts, moreover, may well obscure the raw data required to perform an alternative analysis. Even the raw time series data may be suspect because of reporting and recording failures and inconsistencies in compilation of the time series from the underlying transactions.

[36] An example is the USAF D041 system (replaced by an equally suspect D200 system, to be followed by another system supposedly representing “best commercial practice.”

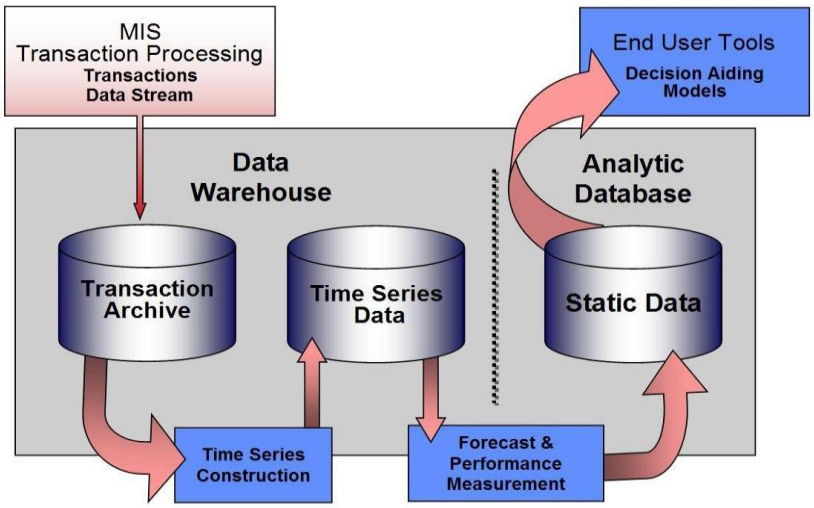

Figure 5: From Transactions to Static Data

The flow of data from the recording of a transaction through various transformations to its corresponding static metric in the analytical database might look like that in the Figure 5. The utility of the data that finally reside in the analytical database is a function of the intelligence and accuracy of the processes used to convert it. There are three broad steps required to transform the "transactions stream" of data.

First, starting at the upper left of the diagram, the transactions are archived as they occur (the stream of transactions is recorded directly in the transactions archive). The archive is important in case better methods are developed later for extracting information from the stream. In some cases, however, the archive may simply be impossible because of the volume of data. In that event, data is lost immediately by committing the transactions to time series. Second, periodically (as infrequently as once a quarter or as often as daily) the transactions are compiled into time series entries. Third, the time series are analyzed to forecast the values of static values such as demand rates and delay times that will be used in modelling.

Each time series records what has happened in each time period. There will be several in a logistics environment, measuring demands (by location and time period) as well as the elapsed time associated with a number of supply chain intervals. Their compilation at first appears to be a trivial recording task, which is largely true for demands. Recording delays, it turns out, is a demanding analytical process, however. All delay time measurements require two time-stamped transactions, which must be properly matched to each other. The length of the time separation between matching transactions is randomly distributed, so pending transactions may stay open for very long times before they are completed. Errors in matching transactions can cause great errors in forecast delays. The treatment of pending transactions is an important analytical task itself, because it provides insight into risks associated with late deliveries. The mathematics of statistical inference methods based on such "censored" data are complex, even arcane[37].

Supply Chain versus Support Chain

The forecasting of demand rates absorbs significant energy in the *supply* chain management business. In many areas, particularly manufacturing and retail sales, demand forecasting is thought to be the final analytical step in the spare stock prediction process. That is, once you have a reliable prediction of demand, you know how much stock to lay in. The process is far different in *support* chain management, which is the relevant field for logisticians.

[37] Consider the problem of determining the average number of years required to complete a PhD at a particular school when half the people who ever entered the program never completed it. Should the period since their enrolment to the present, in the absence of a clear indication they've given up pursuing the degree, be counted? If so, how account for it, since it must be shorter than the total time they'll represent when they *do* finish. For a look at the mathematics involved as, for example, applied to the problem of survival times associated with fatal diseases see, Leung, K M, Elashoff, R M and Afifi, A A, "Censoring Issues in Survival Analysis," *Annual Review of Public Health* Vol. 18, pp 83-104, 1997.

The distinction between supply chain and support chain is crucial. The supply chain has generally come to refer to the processes by which inputs are fed to a production or retail sales process. By contrast, support chain refers to supply issues as they refer to the through-life or in-service period of running systems. One can readily appreciate that the demand-generating processes in these different fields operate in radically different ways.

The supply chain is concerned with the provision of raw materials to a basic manufacturing operation or parts and finished goods to final assembly and retail operations. In manufacturing, the operator exercises significant control over demand rates by regulating the rate of production. In both manufacturing and retail sales, of course, the supply problem is complicated by variations in demand. Nevertheless, demand for the inputs to manufacturing and retail operations generally exhibit a great deal of predictability, if not outright control.

In support systems – the focus of our attention – the measurement and forecasting of demand is quite apparently an exercise in which stochastic models are required to confront highly variable circumstances. This is because demand rates are rooted in the propensity for things to fail, rather than the speed at which a production line is moving. While the latter certainly has stochastic elements derived largely from the pattern of demand for the finished goods, there is a significant difference in the inherent predictability of those processes and the patterns of random failure of complex hardware systems[38]. At best, predictions of the highly volatile demand for repair parts provide the inputs to a difficult risk management exercise, rather than the definitive answers about how much stock is required. A stock transaction managed by the inventory management system is compiled into time series form in the data warehouse. This compilation may require, especially in the case of measured delay times such as repair cycle time and procurement lead time, some considerable analytical effort. Once in the warehouse in the form of a time series, the data can now be further processed, using a variety of analytical tools, to extract their essence in the form of static metrics like a demand rate.

If the owner of the LDSS is forced to take whatever comes out of the bottom of this process, he may be subject to a number of error-inducing problems. In general, the processing done above the level of the static database will have been done for reasons that have little or nothing to do with the needs of analytical models. The methods may be naïve or inappropriate, the time series too long or too short and the compilation methods for the time series reflective of other, inapplicable, management concerns. Therefore, the more of the process illustrated in Figure 5 that can be brought under the control of the owner of the LDSS, the more the result will reflect appropriate practice.

The Frequency of Analytical Exercise

As noted at the outset of this essay, the use of logistic analysis is sparse and expensive. The most significant contributing factor is the time and treasure required to assemble data for each analytical problem. This is a two-part problem. The first part is that, normally, the data will not be found in a form that is compatible with the analytical tools in use. They must, therefore, be gathered, reformatted, vetted etc. The second part is that, even if data are found in a trusted, analytically useful form, it may require a significant effort to bring them up to date.

The first problem is solved by appropriate design of the database in which reside the “static data” required to drive static equilibrium models. The second part of the problem is addressed by the idea

[38] The author discovered how different the supply and support worlds are during a conversation with a supply chain consultant who was explaining his demand prediction process. When he came to a part whose demand pattern exhibited a variance to mean ratio of about .2 he explained that such a part would have to be excluded from the analysis because there’s just no predicting demand with so much variability. In the support chain business, of course, it was a big step forward when Sherbrooke suggested that rather than a variance to mean ratio of 1, which characterizes a Poisson distribution, one might expect to see ratios of 3, 4, perhaps even 10! In fairness, one should also note that in manufacturing demand forecasting, the problem is both stochastic and complex. Retail demand forecasting for finished goods interacts with the forecasting of supplier response times to create a complex, although perhaps not equally complex, problem to those encountered in the support chain, where both sides of this equation (demands and delays) are quite a bit more variable.

of autonomic analysis – automatically maintaining the freshness of the static data. The combination of analysis-ready and up-to-date data should lead not only to more timely and efficient analytical effort, but, by virtue of far lower cost, to more frequent use.

All these attributes, timeliness, efficiency, ubiquity – in short, lower cost – will inevitably lead to more frequent use of analysis in the decision process, and therefore we assume, as a matter of faith, to better decisions.

We contrast three states of analytical effort below. The first, intermittent analysis, is the overwhelmingly dominant form of analysis used in the world of logistics. By periodic analysis, we mean the routine re-visiting of analytical questions. In some organizations, this is done with real regularity, but in most, it is nothing more than a voiced desire. Autonomic analytical systems do occur, though not for the types of planning functions so crucial to orderly management of the support of systems. Among autonomic systems, we distinguish between partially and fully autonomic systems. The boundary between autonomic operation and the introduction of cognition has a lot to do with the utility and adaptability of these systems. We look at examples of successfully operating autonomic systems and lay out our proposed system to conclude.

Intermittent Analysis

The general model for analytical effort has followed the same steps for as long as logistic and cost analysis have been used. First, a question arises from the need to make a decision about a hardware system – whether to buy it, how to design it, how to support it, which type of upgrade to perform, what change in operational capability will result from a budget cut and so on. Next, the analyst either chooses an existing model that addresses the problem adequately or builds his own. Once the model is identified or built, the analyst spends what always turns out to be the bulk of his time gathering data, recording them, checking their correctness, adjusting them for different vintage and application properties[39] and formatting them properly for use in the model. Finally, with whatever time remains to him, he runs the model.

The first runs are to verify that the combination of model and data provide answers in which he can invest a certain amount of faith. Thereafter, when he has convinced himself of both the correctness of the model and the adequacy of the data, he gets to the stage of actually running the model to study the problem at hand. All too often this stage is reached at the eleventh hour, leaving little time to do what was wanted all along – run the model repeatedly, exploring all the alternative outcomes of interest.

Oh yes, there's one more stage: when the job is finished, the analysis budget spent and the decision made, everything is consigned to either a literal or a figurative waste bin, there being no way to make any further use of the data or even, sometimes, the model.

Intermittent analysis in current practice, as just described, is wasteful in the extreme. The least part of the cost is the labor that must be repeated every time a new decision requires analytical input. The more significant cost is the frequency with which the wrong decision might have been avoided. If the analysis takes place at all, it is often too little and too late. In most cases, questions that could be usefully addressed by analysis (that is, analysis could have increased the probability of taking the best decision) have to be answered without that benefit because they had to be made too soon for the normal method to be applied.

[39] Especially in the field of front-end analysis, almost any analytical excursion requires that analysis be performed by comparing the forecast attributes of the system under study to similar attributes of an existing system – for which some data may exist. The process of comparative analysis consists mainly of assembling appropriately adjusted data to describe the system under study, rather than (as popularized by the MIL-STD-1388-1A task descriptions) comparing it with other systems assumed to be similar. The assumption of similarity led to the companion assumption that the comparison would provide detailed insights about the efficacy of the new system design. These expectations were invariably disappointed and extraordinarily costly.

Periodically Repeated Analysis

There is a special case in current practice where things are not quite so grim. In certain organizations and regarding certain types of problems, the same analytical problem is revisited, or intended to be revisited, periodically. For example, an enlightened inventory management organization will periodically revisit stock levels (ranges and scales). As noted above, such models suffer from the handicap of a static equilibrium approach. That is, they believe that the state of the world described to them in the input data will persist, which, of course, it will not. Accordingly, whenever a budget becomes available to allow adjustment of stock holdings, such organizations re-run their models with data that reflect new knowledge gleaned since the last time they did this.

When actually practiced with consistency, this periodic revisiting of the same analytical problem leads to a harmonious relationship between sources of data and the requirements of the model. No matter in what form the data become available to the analyst, he will undoubtedly have developed a variety of automatic filtering and reformatting devices to simplify the flow of data from the source to a form capable of driving his model. The more enlightened the analyst and his organization, the more sophisticated will be the filtering mechanisms. Periodically repeated analysis offers a glimpse into an analytical world that *could* be, not one that really exists. There are several problems. First, there are few examples of this kind of on-going practice, although many organizations have resolved to undertake it. It is much more the exception than the rule. Moreover, it seems only to happen on running or in-service systems, never during the acquisition process when the greatest opportunities to alter the economics of the system still exist. It is also confined to relatively data-simple analytical problems like spares optimization and not extended to more data-complex problems like total ownership cost and level of repair analysis. Finally, although perhaps practiced with some consistency, the analysis tends to be stylized, rarely deviating from a formula laid down sometime in the past, as the practice became a normal part of the organization's activities.

In the in-service world, the practice of periodic analysis is rare and fragile. Rare because most system owners lack appropriate incentives (or misinterpret perfectly appropriate ones) to actually address the decisions required for efficiency, and fragile because the analyst rarely has a voice in the resource allocation process required to keep the data stream intact. If his organization decides to stop collecting a particular variable, changes the definition of the variable, alters the format of the reports he uses or even stops generating the reports altogether, then he may have to change his importing and filtering devices or even abandon the analytical practice, being unable to accommodate whatever changes have occurred.

Even when periodic analysis is practiced the periods are far longer than they could and should be. A strategic spares analysis, for example, is carried out only infrequently; perhaps once or twice a year. There seems little point in doing so more frequently, since spares budgets, like all budgets, are only made available at these long intervals.[40] Level of repair policies are revisited at longer intervals, if ever. In that case, support infrastructure – tools, skilled labor, parts and facilities – have already been purchased and put in place in accordance with the original support policy results. Changing the composition of these long-run assets appears to be either impossible or excessively costly. These assumptions are not generally true, but serve to justify avoidance of the investment necessary to carry out such an effort.

The fact of the matter is that in both the cases cited – level of repair policy and spares holdings – the determining factors change constantly and so do the solutions. That is, policies are changed daily and so are stock levels. Repairs to be made at the intermediate shop are often not done there (the

[40] A much shorter interval is used in the rerunning of a DRIVE algorithm (distribution and repair in a variable environment). In practice, the only application of which the author is aware is the use of the Express model in the U.S. Air Force. The model is used to task USAF repair shops, providing a prioritized list of items for repair over the coming interval. The intervals are set at two weeks. It was initially thought that the resetting could be done more frequently, but as a practical matter, the shop managers could not be asked to change set-up and tooling arrangements any more frequently.

frequency with which this happens is called the “not reparable this station” or NRTS rate in USAF parlance and BCM or “beyond capability of maintenance” rate in US Navy terms). Stocks are augmented continuously between strategic re-optimizations by the item managers responsible to keep airplanes flying or trains running. The problem is that these changes occur without benefit of the types of analysis we believe to be appropriate to make such adjustments efficient. Consequently, in any running system we see continuous departure from whatever strategic plan there is and from any best solution that might have been attained.

Autonomic Analysis

There are very good examples of working autonomic analytical systems, albeit not from the world of logistic analysis. Most of those that come to mind might be thought of as partially autonomic systems because they don’t control much of the day-to-day activity of their systems. They are used, instead, to detect out-of-bounds conditions leading to the shut down of operation in extreme cases or to the prescription of an action (such as a maintenance action) sooner than it would have been undertaken otherwise. There are also, albeit more rare, instances of fully autonomic systems. The most obvious of these is the autonomic nervous system of the human body.

Partially Autonomic Systems

An example of a partially autonomic system is found in the maintenance strategies adopted by modern builders of main propulsion engines for ships. In this case, a large number of condition monitoring sensors are built into the engine and their readings are brought to a central interface[41]. If the buyer of the engine wants to make use of the service, he purchases an interface and satellite transmitter that transmits the state of the sensors at frequent intervals. These are monitored by a maintenance engineer skilled in diagnostics at a central location. When out of tolerance events are detected, the diagnostician determines the severity of the problem and initiates a fix. In the most dramatic form, this may entail a skilled technician getting on an airplane with the appropriate repair parts to meet the ship at its next port of call.

Similar systems are in use or planned for newer aircraft such as the F-22 and the Joint Strike Fighter (F-35). While in the air, on-board diagnostic systems transmit information about failures or imminent failures to the home base soon enough for repair technicians to take steps to prepare for quick turnaround of the aircraft as soon as it lands.

Another example is found in medicine. Patients’ vital signs are monitored automatically in intensive care facilities and the monitoring equipment signals the nursing staff if any of the vital signs depart from an acceptable envelope.

Notice that in the examples, there is a mechanism, either built-in as in the case of the main propulsion engine, or added as in the case of the intensive care patient, whose purpose is to monitor the conditions crucial to the subject’s health. These transducers are almost invariably measuring extremely basic and simple functions. The measurements are fed into fixed algorithms capable only of determining out-of-bounds conditions. The detection of an out-of-bounds condition triggers an alarm, which is answered by the nurse or the diagnostic maintenance technician in our examples. In both cases, these systems are set up so as to economize greatly on the skills required to respond accurately and effectively to the alarm. Two or three engineers are enough to service hundreds of engines, for example, in contrast to a normal military regime in which each ship would have at least one and possibly several people with similar training.

[41] The example with which the author is familiar is the Wartsilla main propulsion engine that was proposed for use on the LPD 17 ship. The engine contained 147 different sensors, which were monitored with varying frequency from every few milliseconds to daily. Results are stored in flash memory and transmitted via satellite to a small group of engineers working in Zurich, Switzerland who handle diagnosis for all of Wartsilla’s engines operating throughout the world.

Whatever the example of a partially autonomic system, it always requires a skilled analyst to read, understand and react to the conditions detected by the system. In fully autonomic systems, the system reacts without explicit intervention by a human intelligence[42].

Fully Autonomic Systems

The best example of a fully autonomic system is the autonomic nervous system of the human body. This system reacts instantly to thousands of stimuli, making slight corrections and the like, without cognitive intervention, for the most part. A finger approaching a flame reacts to the stimulus of intense heat without sending a signal all the way to the brain – the muscles in the vicinity of the finger move it away without conscious intervention. Balance is preserved by tiny signals that trigger muscle reactions, the digestive system passes food through the body through peristaltic muscle action.

Of course, human physiology accommodates for more than these autonomic functions. Many of the same muscles and components of the nervous systems are commanded by cognitive functions as well as autonomic ones. When things go wrong (or right) the brain intervenes, using the same data that caused muscles to twitch automatically, to correct for danger or pursue pleasure.

A fully autonomic system may only be possible in pre-cognitive organisms. It is enough, however, to note that in a system like the human body, most of its day-to-day function is, in fact, handled by the autonomic system. In this, the human body represents a reasonable metaphor for the autonomic logistic analysis system we propose.

Recently, the opportunity was presented to integrate operations of the LDSS with the transactions and data warehousing functions on a specific program[43]. In particular, the system design called for the integration of strategic spares optimization functions with the day-to-day or tactical management of the aircraft's inventory. The resulting, partially autonomous, system is called support chain optimization (SCO). SCO presents an opportunity to gain access to data right at the start of the transaction process, perform fully automatic updating and provide a continuously updated analytical database.

Support Chain Optimization and the Use of Transactions Data

A common problem in even the most enlightened material management organizations is to maintain consistency between the strategic planning functions and the tactical responses to day to-day crises made by item or asset managers (IMs). The central problem is that strategic planning involves system optimization, whereas item management denies the IMs a system view – they must concentrate instead on the range of parts for which they've been assigned responsibility, invariably not all the parts that make up a given system.

In fact, the parts range assigned to an item manager is usually grouped according to commodity types (input, if you will) rather than function (output). There is good reason for this, since part of the IM's job is to be aware of technical issues surrounding the kinds of parts he is responsible for, familiarity with the vendors who supply them and so on. Nevertheless, this practice contributes to putting distance between the system perspective and the daily concerns of the item manager. Inevitably, an IM's purchasing or repair decisions tend to alter stock holdings in a manner inconsistent with the strategic plan, at the same time that they drain funds from the budget that might have been used to improve the performance of the stock holdings. As a consequence, any strategic optimization solution set is overcome rather quickly by the choices made by the IMs.

---

[42] Recall the example of periodic analysis, above. One of the drawbacks cited was the tendency of organizations that actually use periodic analysis to stop exercising analytical judgment, continuing the periodic "studies" without really doing any analytical thinking about the results. Thus, their system has become autonomic in way that one would not imagine was useful or effective.

[43] The pilot project, begun in 2002, concerns management of system-unique consumables for a classified aircraft program as part of a Total System Support Partnership program between the USAF and the aircraft's manufacturer.

Periodic Strategic Optimization

In the best of current practice, a strategic re-optimization is performed at some interval. The length of the interval would normally be a function of the frequency with which spares budgets become available. In the best practice, a static equilibrium spares optimization tool is run and a set of stock buy, move and sell instructions is generated[44]. These are carried out and the system generally responds with increased operational availability or whatever other measure of support effectiveness has been used in the optimization process.

Unfortunately, the optimization process depends on the fidelity of forecasts of crucial values such as demand rates, anticipated changes in operational tempo, basing changes, vendor behaviour and the efficiency of shops portrayed as repair cycle times and repair fractions. In the normal course of events, the rule is that, to paraphrase a popular expression, *things happen*. Little, if anything, turns out as anticipated. Failure rates are higher or lower than forecast, shops close, move, have labor shortages or react well or poorly to a new manager. The equipment operating plan is not fulfilled, or over-fulfilled. Nuances of time distributions of planned maintenance events may not have been taken account of correctly and maintenance load differs during the forecast interval than was implicit in the demand forecasts. This kind of list is virtually endless.

The consequence of the rule that *things happen* is illustrated in Figure 6, below. The solid red curve charts the course of decaying (supply) support effectiveness over time as the things that happen cause the computed stock levels to be less and less efficacious. Whatever the efficiency of the initial solution, it will decay over time as the inputs to the spares model drift farther and farther from what actually happens[45]. The task of the SCO system is to retard the rate of decay by deploying other analytical and managerial devices meant to accommodate the changes in surrounding circumstances.

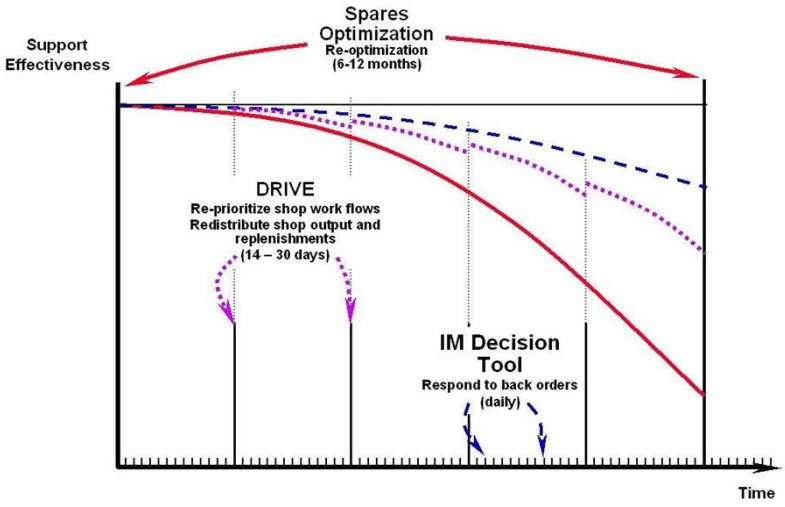

Figure 6: Time Decay of Support Effectiveness

[44] Perhaps an oversimplification. A "sell" instruction, for example, might be infeasible for a variety of reasons, but could be translated into a "delay repair" instruction instead.

[45] While as yet, no empirical evidence exists, one would guess that the shape of the support effectiveness curve will react negatively to the quality of optimization – in a way that advocates of modelling would rather not see.

That is, as the stock solution achieves greater A_o for a fixed amount of money, the rate of decay will be faster. Understand this by noting that the change in circumstances denoted by *things happen* would be irrelevant to a stock solution that ignored those circumstances in the first place. Some evidence is provided for this view by organizations that achieve very high A_o rates without doing much in the way of optimization modelling. Their success is due to the fact that they have squandered excessive funds on spares – they have an inefficient solution, rather than the efficient, but fragile one that arises from optimization modelling.

The first useful device is the DRIVE algorithm[46]. The purpose of DRIVE is to determine the optimal sequence in which ready-for-issue parts should emerge from the shop system with the objective of maximizing the A_o rate of the fleet. In this sense, it replaces the more traditional FIFO (first in, first out) rule for induction of broken assemblies into repair lines.

The notion, akin to the difference between the long run and the short run in economic theory, is that the capital assets of the system are fixed in the short run and the analytical technique attempts to maximize their utility by tracking the changes that have occurred since the strategic optimization run. As a practical matter, the output of DRIVE is a shop directive indicating which parts should be inducted into the repair stream and in what order. The frequency with which DRIVE runs can be made and promulgated as new shop instructions is predicated on the ability of the shop to react efficiently to new instructions. Done too frequently, redirection will pose setup costs, for example, that neutralize the beneficial effect of repairing the right parts.

Implicit in the idea of DRIVE is that stock levels first determined in the strategic optimization run will be ignored once they have been established. In other words, stock no longer belongs to any operating or support entity, but to the system as a whole. Once it is repaired, the part may be sent to any base or lower echelon shop, not just the one from which it was received. This is also true of replenishment parts reaching the supply system. The effect of these adjustments is shown in Figure 6 by the dotted purple curve shown in segments between the intermediate length intervals on the time axis.

Day-to-Day Inventory Management

While the DRIVE algorithm is capable of overcoming some of the negative effects of the *things happen* phenomenon, it can, at best, only improve the efficiency of use of existing capital stock. As noted above, however, the stock is dynamic. It is continually revised by the actions of the item managers in reacting to local shortages or threats of shortages. If the decision process of the item manager could be informed by the *system* implications of his actions, we might expect that the changes in capital caused by his actions would counter the effects of the *things happen* phenomenon, rather than exacerbate it.

Accordingly, the second device one might use to ameliorate the decay of A_o is an economic decision tool intended to sit on top of an item manager's asset management information system. The effect of this device is represented in Figure 6 by the higher dark blue dashed line.

Loosely described, the device consists of two elements. The first element is a comparison of two tables of identical dimension called the "what is where" table (WIW) and the "what should be where" table (WSW). The dimensions of both tables are the range of parts for which the IM is responsible by all the locations they are or might be. The first table is a conventional part of the asset or item manager's tool kit, usually called asset tracking. The second table is determined by running the strategic optimization model, providing a set of stock recommendations identical to those that would be used for periodic re-optimization, re-prioritization or re-distribution. The optimal stock solutions (quantity and location) for the item manager's range of parts populates the WSW table.

The second element of the device is a group of algorithms that measure the true economic cost of alternative actions that might be taken by the item manager in response to an increased probability of a back order.

[46] First invented by Dr. Sherbrooke, DRIVE (distribution and repair in a variable environment) is currently in use by the USAF as the core processing idea behind EXPRESS. The USAF implementation, however, does not perform both the shop planning and re-distributive computations spoken of in the text. These have since been added by Sherbrooke and have been given the name OverDrive™ to distinguish the integrated analytical tool from the simpler DRIVE algorithm.

What is Where versus What Should be Where

Comparison of the WIW and WSW tables signals the existence of problems whenever there is a difference in the corresponding cell. Two situations are illustrated in Figure 7. An excess of six of part # 123 is detected at location 1 (corresponding green cells) at the same time that a shortfall of one is noted at location 4 (corresponding red cells). The remedy appears obvious – move one of the excess parts. A more common situation is illustrated with part # 129, namely, a shortfall at location 4. In this case, there is no apparent excess in the system. The output of the spares optimization model provides both the stock values for WSW and the quantification of the change in the probability of a backorder associated with the shortfall, illustrated for part # 129 as the striped region of the probability density function. This probability, in turn, provides part of what is needed to compute the expected cost of a backorder.[47]

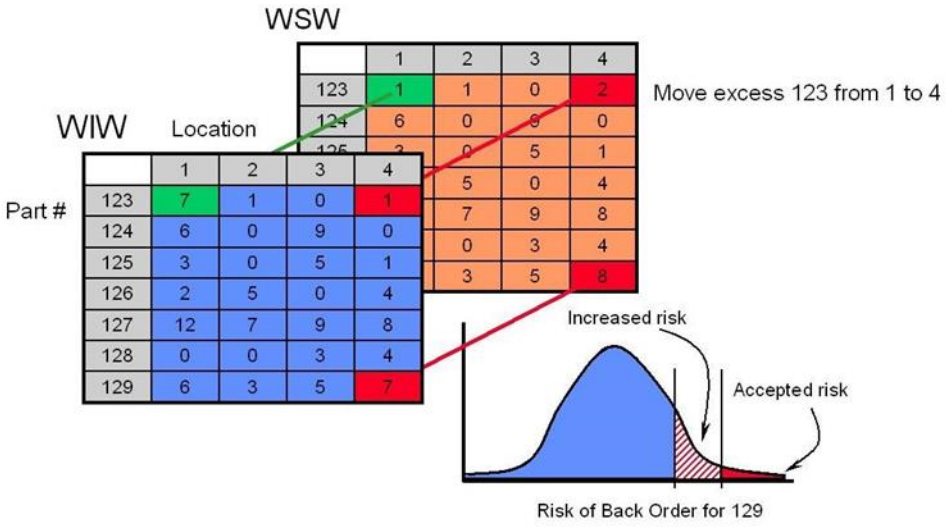

Figure 7: Real-Time Comparison of WIW and WSW Tables

True Economic Cost of Alternative Responses

Assuming the cost of downtime is known, or a figure has been decided upon to represent it, the cost of the shortfall of part # 129, if one takes no special action to correct it, can be modelled in simple terms as:

$$Cost_{DoNothing} = Cost_{DT}(t) \Delta \Pr(BO) * ED(t),$$

where $Cost_{DT}(t)$ is the cost of downtime as a function of the duration of the downtime event, $\Delta \Pr(BO)$ is the increase in the probability of a backorder[48] and $ED(t)$ is the expected delay before the shortfall is corrected by normal operation of the logistic system. The cost of downtime is an estimate or policy variable. The increase in the probability of a backorder is estimated by the spares optimization model and is available from the same stored data that provided the WSW stock estimates. The expected

[47] Much – perhaps too much – is made of the difficulty of determining the cost of down time. In particular, in a public welfare forum, it is conventional to claim that, since we are looking at public goods, there is no market to produce a price. This argument is, for the most part, specious. Any decision process that uses a different metric or measure of effectiveness is implicitly dealing with the cost of the metric (i.e., the price). In fact, I argue elsewhere that the problem of setting an A_o target in a public setting is the obverse of the problem of determining the value of downtime. The decision maker who sets the presumably technical target is constrained to do so according to the discipline of maximizing net public benefit – which implies a feel for the value of avoiding downtime. A moment's reflection will show that budget allocations made by public bodies are less pointed, but nevertheless relevant measures of the relative market value of the goods and service provided by the budget recipients.

[48] The probability of a backorder at a moment in time is usually referred to as expected backorders or EBO.

duration of the shortfall is a function of a complex of tracking information that must be made available to the asset management software – the disposition of all instances of part # 129 not held in ready-for-issue stocks. These are dues-in from both organic and contracted repair processes, replenishment stock or stock currently in transit between locations.

In fact, beyond doing nothing, there are several other options available to the asset manager to respond to an increase in the probability of a back order. These include:

- Purchase a new part
- Borrow a part from another stock location (lateral re-supply)
- Cannibalize (remove the required part from another disabled system)
- Move stock from one location to another
- To reduce risk
- In support of anticipated changes in operations
- Expedite various actions to reduce waiting time
- Delivery priority (means of transportation)
- Purchase process
- Organic repair
- Vendor repair

The essence of the Supply Chain Optimization (SCO) idea is to provide decision guidance to the asset manager in the form of true cost[49] estimates of each alternative action intended to avoid the shortage. All costs are the discounted present value of cost streams since each action implies a different length of downtime. For the owner of the supported equipment, who provides his own support, these relationships are relatively straightforward, depending on good knowledge of how his logistic system actually works. For an enterprise providing support under contract to the owner of systems, the situation is more complex. In general, the cost of a shortage relevant to the contract supplier will be related to the incentives contracted with each customer. In the case of single stocks serving multiple customers with whom different incentives have been negotiated, the algorithm can become quite complex.

The direct and indirect costs of each alternative action must be re-defined dynamically. For example, the cost of buying an additional part might be $10,000 compared to the cost of expected downtime of only $5,000. However, if the asset manager had foreknowledge that, say, two more of that part would be purchased in any case sometime in the near future, then the $10,000 cost is reduced to only the interest on $10,000 for the interval between the current date and the date on which it would have to be purchased anyway. How is such foreknowledge made possible?

Continuous Database Update

Consider how real time data about stock transactions are provided to the item manager. The stock, store and issue authority continuously catalogues orders for parts, receipts of shipments from repair and supply vendors, drawing down of parts to satisfy requests and so on. This “transaction stream” is monitored by the item manager in the form of changes to the What Is

[49] By true cost, we mean the true economic cost of alternative actions as opposed to their budget cost. This idea is amplified in the text. In general, making a decision about *what to do* can only be done correctly by comparing the true economic cost of alternative choices, paying attention, in particular to both marginal cost and opportunity cost. Alternatively, financial cost estimates are needed to determine *how much money* is required to carry out the actions upon which the decision-maker has decided. These two forms of analysis must be rigorously separated, at the same time that they faithfully describe the same circumstance.

Where (WIW) table. Those changes, in turn, are evaluated in the light of What Should be Where (WSW) and alarms are selectively sounded to gain attention where necessary. At the same time that the transactions are passed to the WIW table, they are also recorded for analysis in the data warehouse associated with the SCO system. There, they are catalogued and compiled into time series data measuring both delays and demands.

> *Every transaction carries with it the possibility of changing the forecast demand rate or delay time associated with the part in question*[50].

The cumulative effect of thousands of these transactions in any time period will be to cause many, albeit usually subtle, changes in both demands and delays. If these metrics are re-estimated periodically, their values are, therefore, subject to change. In addition to this mining of the transaction stream, every day brings new information regarding future operational plans, basing changes, configuration modifications to the hardware, new price lists from vendors, plans by parts suppliers to terminate production of crucial items and a plenitude of other things. All this information can be translated into new values of the variables that drive estimates of optimal stock levels[5151]. That is, even if the spares optimization model is run every day, there are likely to be changes, sometimes small, but other times very significant, that will work out their effect through the comparison of the WIW and newly-refreshed WSW tables[52].

Autonomic Logistic Analysis

In the same way that the human autonomic nervous system takes care of the mass of reactions required for day-to-day function, autonomic logistic analysis (ALA) could regulate much of the day-to-day function of support for a running system. To extend the metaphor, the data used to guide the day-to-day automatic functioning of an inventory management system is also the basis for cognitive intervention or logistic planning.

The ALA analogy to a condition monitoring system is the data stream recovered from stock, store and issue transactions produced by the inventory management system. These transactions provide the low-level data from which everything else proceeds[53]. The transaction stream is converted into time series data, which is, in turn, subject to analysis aimed at providing updated data inputs for the models. The strategic analytical functions represented by the optimal spares model and the DRIVE

---

[50] Other data elements may be updated from these transactions as well. An example is the unit price of the part, which may have changed since the last order. Administratively useful data elements such as the vendor's contact details might also be updated.

[51] The discussion of this section is limited to support chain optimization. Opening the discussion back up to the entire field of logistic analysis, however, the same could be said of such other crucial information as level of repair policies, life cycle cost estimates, budget requirement forecasts and so on.

[52] A discontinuity should be noted between the purposes and use of static equilibrium strategic optimization models and the need to manage inventory day-by-day. This is illustrated by the problem that, say, in two months, the fleet will undergo a sudden change – say its size will be doubled by the delivery of new systems or it will deploy to a new station. Planners use the static equilibrium model to detect what changes will be necessary to accommodate the change. How are these results passed on to the WSW table to elicit the correct responses from item managers, however? The issue is that a detected shortfall will only cause downtime two months into the future. Is this discounted to present value and seen as a gradually heightening risk or is the whole issue of planned change taken off-line and considered separately? We believe that the methods outlined here may actually help bridge the gap between the two normally separate processes. In any case, we anticipate learning more about such questions only as the system described here comes into day-to-day use and such situations actually arise.

[53] The narrow focus on inventory management and on running systems is intentional. I believe that there are many other useful sources that would supply updates to a wider range of variables in an analytical database. For example, the transactions processed through a maintenance management system should yield improved data on repair times and matching data on consumables. These systems, however, are notorious for their inaccuracy and some considerable thought needs to go into the proper interpretation of the data. There are also continuously updated or augmented data available during the acquisition process that offer a potential source analogous to the transaction stream. The problem in this area stems from the lack of consistent approaches taken by different organizations to the generation and management of data during the various stages of the acquisition cycle.

algorithm (see above, page 37), which produce the analytical guidance needed to continuously update the WSW table. The transaction stream updates the WIW table. The out-of-bounds alarm trigger is derived from the comparison of the two.

The out-of-bounds condition has two aspects. First is to discover the existence of the condition and next is to determine the correct response. Recall that the SCO system described above performs calculations of the true economic cost of each alternative action the item manager might take in response to the condition. These cost equations combine information about the cost of downtime, the change in the probability of a backorder causing downtime, the expected waiting time associated with each action, the budgetary cost of the action and the opportunity cost issues that might ameliorate or exaggerate the budgetary costs. [51]

It is likely that the item manager will be able to account for additional information not anticipated or recognized by the cost model or other algorithmic processes. For this reason, the relative true costs are presented to him as a decision aid, rather than the decision itself. The item manager then either follows the advice of the cost equations (chooses the apparently least costly alternative) or makes a different decision based on other knowledge. Conceptually, we could imagine that the equations had been tuned to such a degree that, after some lengthy experience, this intervention would either become unnecessary or have a lower success rate than simply following the advice of the equation system. There is a certain amount of social and organizational change, as well as a good bit more research required to foresee that kind of system, however. For now, we expect the cognitive function would intervene at this point.

The other cognitive function, analogous to human cognition using the same data as the autonomic system, is the planner's use of the database automatically updated by the conversion of the transaction stream into static data updates. That is, when the time comes to perform the periodic analytical excursions that will set the strategic plan for the coming time period, the data need no further updating, except in the area of the operational plan. This arises from the continuous update process used by the specific tactical inventory management function, which also serves to update the broader database used by the strategic tools.

The complete autonomic system is illustrated in Figure 8. The sources of inputs are the continuous transaction stream from the stock, store and issue authority and inputs defining operational plans, basing, deployment, configuration changes and other similar data from a variety of sources, shown as dark grey boxes with white lettering at the top of the diagram. The latter may be partially or fully automated, but will not be in the form of transaction streams. Instead, those data will appear in the form of periodic updates.

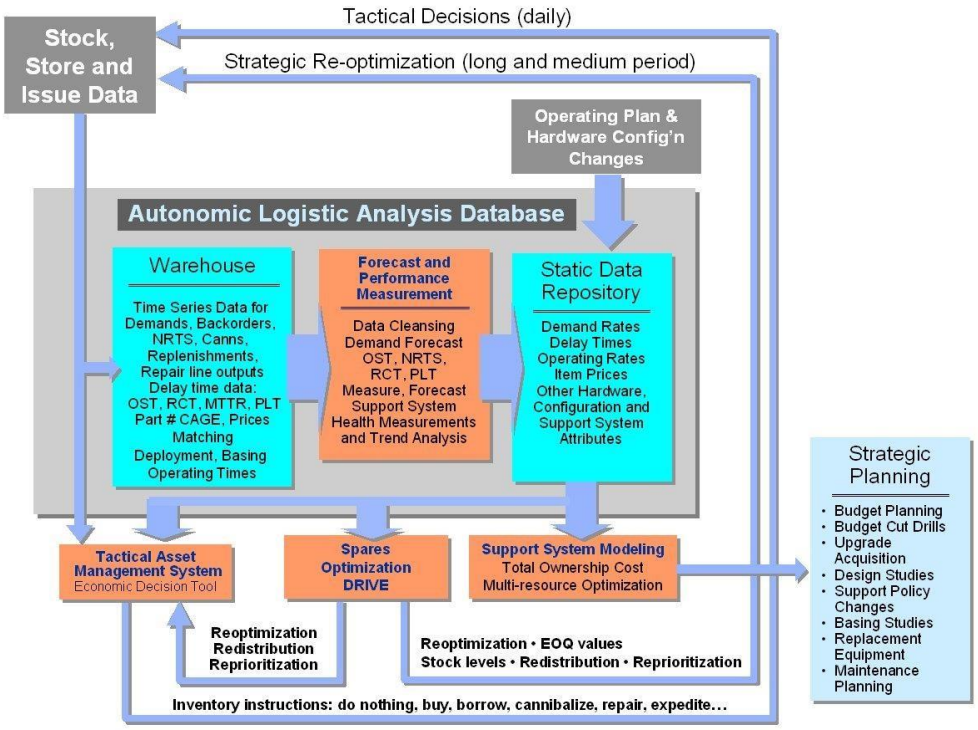

Figure 8: An Autonomic Logistic Analysis Architecture

The data holding areas of the data repository (the commercial name for this repository is the TFD

Data Vault) are the warehouse on the left and the static data used for analytical modelling on the right. They are connected by the transformation function of the forecast and performance measurement software. This group of algorithms will have to be modified to suit different industrial sectors, although the theories reflected by the processing are similar. It should be noted that the biggest difference in demand forecasting practice occurs between the methods used in low variance situations normally encountered in areas like retail sales and manufacturing, and the high variance processes observed in the demands for randomly failing equipment. The latter are the central focus of this thesis and are the most suited to (and helped by) the proposed approach. Nevertheless, we see no reason why the same methods would not be of substantial benefit to other sectors.

The autonomic aspect of the logistic system envision in Figure 8 is best expressed by the box on the lower right corner of the diagram. These strategic planning functions, special studies and the like are the focal point of the ALA idea. The motivation for the data system is to be able to manage inventory – and that is sufficient to warrant its implementation. But the additional capability to plug a complete and up-to-date dataset into any one of a wide variety of analytical methods *without any specific effort to gather and update the data* is of great significance and comes at virtually no additional cost. The apparatus needed to performing the inventory management function automatically provides this crucial ancillary benefit.

Future Implications of the Autonomic System

Decisions made by item managers can be significantly improved by implementation of the type of system envisaged here. In particular, the tendency of the system to maintain consistency between tactical and strategic functions will save money by causing the item manager to make purchases that anticipate, rather than conflict with, the coming strategic plan. A by-product of this new consistency will be lower top-up spares budget requirements (effectiveness levels remaining constant) and a corresponding reduction in importance of the periodic optimization exercise. Because of underlying structural changes, the need for such exercises will remain. However, because the system will not

have been allowed to become as noticeably broken between times, the top-up events will no longer be the much-anticipated moments when the system gets "fixed." [54]

There are other implications that may well emerge as we gain experience with autonomic systems. Efficiency of inventory management is certainly a worthwhile goal, albeit a narrow one. The proposed system includes analytical devices that reach out to every decision-making area of the management of both acquisition and operation and support. By cutting data preparation time effectively to zero, it drastically lowers the cost of analysis and reduces the waiting time for its results. This will enhance the usability of analytical inputs to decision processes. It will also undoubtedly enhance the sophistication of analytical effort by increasing the time available to the analyst by an order of magnitude.

These effects will be felt, not just in the inventory management sphere, but also in budget forecasting, mid-life upgrades, support policy changes and other more subtle, and potentially more economically powerful, decision processes.

Further exploitation of the autonomic properties might also be anticipated. For example, item management is generally labor intensive. The same computer system that provides the item manager with the relative true costs of alternative actions can record everything about the decision transaction that ensues. We can record both the stimulus and the response, as well as the relative costs displayed – and even the outcome of the event. By doing so, item managers can be "calibrated" regarding the extent to which they accept the guidance of the cost equations. One imagines that some will follow the equation's advice slavishly, others will ignore it as a point of honour and still others will exercise a more considered judgment in its use. For the first group, the computer system has become the *de facto* decision-maker. The performance of those in the second and third groups might be compared to that of the computer. Those whose decisions perform better are clearly a valuable human resource. Those whose decisions do not might be replaced or retrained.

If the cost algorithms themselves were made adaptive – their parameters a function of continually changing data about how events actually turned out – their rate of success could also be expected to increase over time. Ultimately, much of what we currently believe to be a labour-intensive process might be converted to a partially automated process in which far fewer and more capable practitioners could perform the decision-making duties of item management.

Robert A. Butler
Chairman and CEO
TFD Group
70 Garden Court, Suite 300
Monterey CA 93940
Robert.Butler@tfdg.com
+1 831 649 3800
+1 831 649 3866 (fax)

[54] In fact, since the first writing of this paper, our views have come to the expectation that, over time, periodic spares optimization exercises will disappear altogether. The reason is that the tactical decision process, being guided now by strategic considerations becomes a perfect substitute.