Small World, but how should it be organized?

Chris Spottiswoode
Synergy Computing

A. Naive Real-Time

You are a healthy human being, aware of what is going on round about you. Your nervous system relays its sense-perceptions to your brain fast enough for you to be able to respond to threats or opportunities in your immediate surroundings: you have a 100% real-time information system.

Your computerized inventory control system records every inventory movement as it takes place, so you know exactly when to reorder or move or ship your goods: you have a 100% real-time information system.

Both statements are wrong. They represent what I call "The Naive Real-Time Fallacy", propagated by the optimists of this world. And I am not referring only to computer salesmen: for are we not all optimists!

Where is the fallacy? And how can it be corrected?

Let us start with an extreme counterexample. Our eyesight and our reactions are not fast enough to spot and dodge falling meteorites. "Of course not," you respond, "but there aren't enough falling meteorites to bother me. 99.999...% is good enough for me." And mankind's millions of years in the survival race prove you right.

No country is fast enough and powerful enough to spot and neutralize falling ballistic missiles. No company has a perfect inventory movement recording system or inventory reorder algorithm. "Just a matter of improving technology," you may well retort. And you are almost right. But not quite, for the solution may prove more costly than the problem, and finally our competitors can improve too. We should not even expect anywhere near 100% real-timeness or accuracy. "Just enough!" should be our watchword, as indeed it has always been throughout the saga of evolution.

So let us accept that we have 90% real-time information systems, which are quite good enough for us. Our knowledge of what matters in the world around us marches along with that world, even though we are sometimes a little behind in our perceptions or reactions.

Many first-time purchasers of inventory control systems are still at this stage of understanding of the system they are expecting. Regrettably many designers of real-time inventory control systems are still there too! For this is still Naive Real-Time. The above qualifications are still not the point: the message from Distributed Processing to Naive Real-Time is not that slow or irregular or infrequent communications make real-time difficult. Let us summarize the naive conception of real-time and see what the real problem is.

Imagine a data-structure diagram of our naive inventory control system: we probably have at least an item master, an inventory master, an inventory movement transaction detail file and various order files, all adding into summary fields for that instant access to the current status. There are data access paths where necessary, and we (the end-user or the programmer) can wander around these paths and establish the exact inventory position, order-status, etc.

As Charlie Bachman ("The Father of Data-Base") puts it (in his ACM Turing Award Lecture of 1973), we are now "navigators in a data-base". The Data-Base Revolution in programming is like the Copernican Revolution in astronomy. The CPU, like the sun, is no longer the center of our universe: we must situate ourselves in a universe of data.

In this Information Age we must take this view, but the mistake one then naively makes is to look at a business enterprise as if it were a human being. A rather similar parallel was drawn in the seventeenth century by Thomas Hobbes, in his book Leviathan, where he compares the state or "commonwealth" with a giant body, with its lines of authority and central sovereign (or as we would say now: lines of communication and central co-ordinating organ). Nowadays we look back on Hobbes as one of the intellectual fathers of political absolutism. This leads us on to look at how the real world has forced us to adopt more decentralized and democratic views.

B. The Real World

Naive Real-Time had its heyday in the late sixties, but the centralized on-line mainframe was soon recognized as the monster or Leviathan that it was. The price and limitations of all those telephone lines for so-called instant access, and the complexity of the programming on those big beasts simply made the costs too high. These issues totally obscured the more fundamental design complexities to which we return later, but they were sufficient to lead to rebellion.

Distributed Processing was rising in the seventies even as Charlie Bachman was giving his Turing address (though in 1973 we were doing it before the name was invented!). By the end of the seventies "Small is Beautiful" had become the motto. By 1984 the independent PC, that symbol of the democratic revolt, had been blessed by IBM (and how satisfying that we should be celebrating this in George Orwell's dreaded year!).

But what has happened to that elusive ideal of real-time information? With slow or infrequent communications between distributed files, it has some definite limitations. How much has it mattered? Briefly: not much.

Clearly there are major application areas where a high degree of real-timeness is essential and used: airline reservations and automatic bank tellers are familiar examples, though even here off-line or degraded modes, with their various consequences, are part of the total design. But the majority of enterprises with integrated computerized applications simply do not require total 100% (or 90%) real-time systems. Why not?

BECAUSE THERE IS A
FUNDAMENTAL DIFFERENCE

We shall see that the fundamental error in Naive Real-Time is to assume just one observer or central actor, who sees the entire data-universe as if at the same time, that is, with no time component, no time dimension within which to distinguish the points in the data-space. We shall see the same problem that Einstein saw in the Newtonian universe: there is no simple definition of simultaneity. Naive Real-Time is equivalent to absolute Newtonian space, the whole of it being observed as if at one time. We need to introduce the concept of relativity to Real-Time.

BETWEEN AN INDIVIDUAL
AND AN ORGANIZATION.

Organizations exist because the individual cannot do everything himself. In our complex society we must specialize. We each live and work in our own little worlds, communicating and co-operating with others. Informal communications between the individuals in any one department enable that department to act as one person, with one real-time. An organization consists of multiple departments, each with its own perfectly valid naive real-time.

In a typical manufacturing and distribution company, we do not involve the sales order processing department with the details of the factory order planning, at most we let them know the results. In some applications warehouse staff might not use the theoretical inventory levels at all: it is their job to move goods and record those movements. Frequently the theoretical levels will only coincide with the physical levels at stock-take, since they may at other times refer to unsold inventory, which is not the same as inventory actually in the warehouse.

The sales order processing department is not concerned with the processing of customer billings and payments. That is the job of the Credit Control department, who in their turn update the on-line customer file with credit status, not "real-time" balance, so that orders for risky customers are referred to them.

So we have two examples of "real-time" values, customer credit balance and inventory balance, which do not necessarily refer to any "real" balance in the world represented. Why not? There are two main reasons. The first is the obvious one that the balance may be incorrect

or misleading. Not all relevant movements may have been entered, and others may have been entered incorrectly (This is part of what made us reduce the 100% to 90% real-timeness). The main reason is however that the real-time balance has a different meaning, such as unsold inventory, which refers not to physical inventory but for example to on-hand plus expected from factory plus resaleable returns less expected shippings for some known period. And it is usually quite a task to relate theoretical to physical inventory from time to time.

The meaning of any value on the real-time database depends on the point of view of the user for whom it is intended. Now this is so obvious that it is well to reflect on why it has become necessary to say it. Well: the culprit is the dogma of non-redundancy of data, created by the DBMS missionaries! Before DBMS, each department had its own files, so of course they were looked at from the point of view of that department. Now we are tempted to have one real-time inventory balance field, and it becomes a non-trivial task to relate it to its various potential users.

But relate we must! Without further ado, let me describe the Relativistic Real-Time Data-Universe.

Modern organizations are not run by some superhuman Leviathan or Big Brother who knows everything in (naive) real-time and directs activities with infallibility. An organization, by definition, consists of separate departments organized to achieve some more or less precisely understood objective. Each department runs in its own real-time, isolated to a greater or lesser degree from the others. Our limited minds require that we work in simplified worlds, and historically organizations have evolved taking the sizes of our minds into account. (As an aside, one wonders how the much-trumpeted Fifth Generation will cope with the essentially human problem of assessing what people can take.)

As in astronomy, all points of view can start by defining themselves relative to the fixed stars, which may in our case be an Item Master and a Customer Master, though at this point without such attributes as balances.

Then a Credit Control department, for example, will add balances, credit status, transaction entry files with associated controls, and sundry other local data. They will work independently of the order-processing department, say, except in two respects. The first involves the Accounts Receivable statement close, when Credit Control determines that its transaction data is complete and accurate enough, and initiates the close. The balances are updated and the customer credit statuses are deduced as a function of the balances and other factors. Order-processing might refuse orders or refer them to Credit Control as determined by these credit statuses, which are derived from the same transactions that are communicated to the customers on their statements. These referrals of orders back to Credit Control represent the second type of interaction between our two departments. Let us characterize these two types of interaction in more detail.

The second (we return to the first later) was a stream of referrals of incoming orders back to Credit Control for possible credit authorization or other follow-up. This is a queue (or queues) of objects or entities representing a flow of units of work between the two departments. There will be more such queues, such as the queue of authorized orders passing on to the shipping department.

The first type of inter-departmental interaction was the close, in the above case an Accounts Receivable statement close. Now we associate this with a batch job on the computer and hence tend to think of it as a computer-created evil. Not so. Such computer batch jobs are always associated with a real application event: a cut-off. In this case Credit Control triggers the event by "cutting off" the A/R transaction flow from outside the computer and thereby synchronizes various parties, for example data capture, order-processing, and the customer base. Refusals by an order clerk may be by referring the customer to the state of affairs as it was at this "sync-point". A later "real-time balance" may confuse or mislead any or all of the parties concerned: because of delays and/or errors in payments, mailing, encoding, data-capture, or data-control (to name a few possible sources) a balance at any time other than at a close should perhaps most diplomatically be used by the Credit Control department itself rather than by an order-clerk. The close- or cut-off- related sync-point provides a much more clear and unambiguous communication.

A more interesting application-defined cut-off is to be seen in the physical inventory count or stock-take. Here warehouse, shipping, Credit Control (re returns), and accounting have a familiar sync-point around which they orient themselves relative to each other. Inventory movements are frozen, or cut off, during the stock-take. A batch run will "transform" the real-time inventory balance used by order-processing - by

making adjustments such as adding back allocated orders - to make it coincide with the real physical balance. If this balance was being maintained then only the next step might be necessary: an exception report of variances to assist all relevant departments with the error correction or reconciliation process. At other times the separate departments will operate much more independently.

So we build up the picture of each department with its private data, its communications by means of queues, and the synchronization of its own real-time with the real-times of other departments by means of cut-offs and possibly associated batch runs on the computer.

At this point I might just add that these concepts lie behind a package that has now reached an advanced stage of beta testing: SYNQ (for SYNchronizer and Queue-manager) is a system control package based on a system design language called IDIOM, in which the system designer describes the organization in terms appropriate to its own peculiar interrelations.

By means of the SYNQ package, or by appropriate use of, for example, Data-Flow Diagrams, or simply by otherwise emulating SYNQ's queue-management and synchronization functions, the system designer thus defines the interactions between the various departments, whether they are manual or computer-assisted.

We end up with the relativistic picture of an organization's departments each working in their own real-time or time-frame, with the time-frames being related to each other, or synchronized, by transformations effected by means of batch jobs.

This parallelism with Relativity in physics is not merely contrived. Each recognizes that we must start with the viewpoint of the observer in his department or frame of reference; that absolute simultaneity between such viewpoints is not possible, since simultaneity is based on finite-speed limited communication between viewpoints; that different viewpoints, except where they share common fixed entities or resources, can only be made to correspond by means of transformations of data between viewpoints.

C. Consequences for computer system design

C.1 In Application Design

It is safe to design an application for use within any one department. Naive real-time is good enough. If the entire application is managed in detail by one person, then you can obviously ignore Relativity. This is one reason for the success of micros. And PCs will continue to play a major role in the more isolated activities, whether partly on-line or not.

But if your intended computerized application spans departments look for the shared static resources and for the interfaces and transformations - the communicating queues and synchronizing cut-offs. In this way you reduce a complex integrated organization to multiple, simple, end-user understood viewpoints in which Naive Real-Time holds true. If you are cautious you will not change the departmentalization, because it defines the simplified worlds of the actors in your drama.

Incidentally you will find that your database design is much simplified, because you too will be able to work in largely separate simplified worlds or databases.

But if you want to remove or change inter-departmental walls, beware! The move may be justified, but it will only be accepted and work if the computerization removes the raison d'etre for the barriers, and if the effect can be seen by the end-users to be a simplification. My warning is not that change is always resisted: my experience is rather that change is welcomed if it can be perceived to be a simplification. It is confusion and perplexity that is resisted.

Now if the computer can simplify the activities of the organization, how is this to be brought home to the first and all later users? The first must be sold on the benefits. Prototyping and system modelling or simulation would help. Then all users must be able to find out how the new organization hangs together. Remember that the new technological organization probably relies heavily on the computer for its communications. On-line help is probably the answer.

The structure of this on-line help must also be Relativistic: a Leviathan of a text, even one beautifully organized in top-down fashion, will probably not be read. Your on-line help must be structured to mirror the organization:

simple and relative to the context, but it must also be possible to easily establish the role of the current activity in any continuing interdepartmental communication or approaching cut-off.

Finally, the operations of the computerized organization should be under the control of the end-users. Those batch runs are not yours, they are part of the end-users' regular functions and should not be hidden from them. If a particular close normally takes two hours, the end-user wants to know and draw up his schedule accordingly. He must be able to find past run time history and (within limits) program different schedules on the computer.

An end-user oriented application operating system is indicated. You can program your own (Who has not written an applications monitor or control program?) and build in those modelling and on-line help functions, or you could use a package like SYNQ.

C.2 In technical design

One often hears statements such as this: In on-line systems, 75% of the difficulty is in ensuring data integrity. This is surely true if one broadly interprets integrity to encompass access security, data consistency, accuracy, and fault-tolerance.

Clearly a Relativistic approach will result in a rational and simple solution to defining appropriate access rules.

Data consistency is less of a problem the less you are a slave to Naive Real-Time, which dictates that you maintain multiple on-line summaries and access paths, "in case someone wants to use them".

Inaccuracy of data can never be avoided, it can only be reduced. Errors must be controllable, that is, detectable and correctable before too much damage is done. This is one of the major functions of the separation of departments: get the work done quickly, but don't pass it on to anyone else until it is complete and accurate enough. That is what cut-offs are all about.

By fault-tolerance I mean tolerance to technical computer faults such as power or system or program failures. There is no simple solution to these problems, only compromises. But there is one general approach to this set of problems: the system must be restartable or reconfigurable from known checkpoints or sync-points. On-line programs can use logging and recovery mechanisms, batch programs must be broken down into restartable steps. Clearly an application designed with rational and clearly understood interrelationships between functions, that is, a Relativistic design, will more easily be made restartable and have more easily designed and implemented degraded modes.

D. Conclusion

This paper has taken a largely informal and sometimes high-faluting approach to what is essentially a very simple, though much overlooked, issue: some common traps into which the application designer can fall when designing integrated applications.

The human and philosophical aspects were emphasized because they are part of the problem, and the analogy with Relativity concentrates the mind on the essence of the logical problem.

A more formal and systematic formulation can be made (though this cannot be done here) of the simple concepts this analysis reveals. Such an approach - with which we are experimenting at the moment - will create opportunities ranging from Critical Path analyses for integrating inter-department scheduling with the D.P. implementation, to network resolution exercises for simplifying the multi-department organization.

Chris Spottiswoode has been in D.P. for over 14 years, the first 6 with a manufacturing and distribution company, for which he implemented the first mini-based nation-wide commercial distributed processing system in South Africa. Robert Gibson and he formed Synergy Computing in 1978 and in 1979 were responsible for the first purely commercial HP3000 in South Africa. Synergy is the Distributor in S.A. for Robelle Consulting, Quasar Systems, VESOFT, and Account-A-Call. Chris is married, with 3 children, and enjoys mountaineering, skiing, sailing, tennis, philosophy, and politics.