

# Recycled Designs



Neil Smith & Suzy Smith  
Aspen Software

## Summary

*Consider....*

- ◆ *How do I make an R:BASE report available on demand from my web site?*
- ◆ *How do I email a sales report for the last week to 100 sales managers every Monday morning?*
- ◆ *How do I make sure my system collects and processes its mission critical feed files every morning at 3:00 am?*
- ◆ *How do I make sure my R:BASE databases are correctly reloaded and backed up every night?*
- ◆ *How do I manage the replication of information across an international network of R:BASE databases?*

*These are the typical challenges that can form component parts of many development projects. Since the success of the majority of today's businesses is inextricably linked to the capabilities of their IT systems, clients often demand such facilities.*

***“Recycled Designs” will show how we have used an existing design concept to meet these challenges. We will share a design theme (based on ideas from an early Developer Conference) that has formed and continues to form the basis of many successful Aspen Software solutions. We will explain how we have used R:BASE, Oterro and web access technology with this design and implementation approach to achieve these capabilities and more.***

## Challenges

As developers, we spend our time architecting solutions to our clients' business problems and opportunities. Indeed the success of the majority of today's businesses is inextricably linked to the capabilities of their IT systems.

As a result of the pervasiveness of the Internet, (and the marketing departments of many organisations) today's IT users have high expectations of the applications they will be delivered. Yet, in the current business climate, these applications are supposed to be developed and deployed with breathtaking speed! Furthermore, their build quality has to cope with the demands of increased operation as more and more business move closer to 24-hour operation!

Finally (and not unreasonably) the people who are paying for these applications expect them to work and to deliver real returns to their businesses!

So, we have to produce ever more sophisticated solutions, that satisfy problems of ever increasing complexity, quickly and cost effectively. Let's consider elements of some real life examples.

## Challenges

- User expectation
- Solution sophistication
- Speed of business change
- 24 x 7 operation
- Provide real return on investment

### Web Enabled Reporting



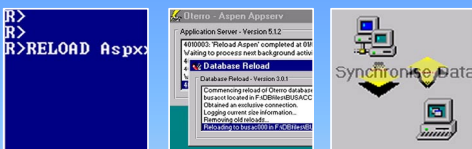
- Specialised web enabled application
- Diverse user reporting requirements
- Users select & request their own reports
- Ad-hoc & regularly scheduled reports
- Fully automated report distribution by email

A web enabled database application, that lets web users select and run reports that are normally available on the LAN based version of the system.

As well as seeing their reports immediately via their browser, they can have the results automatically emailed to them - every week if they want.

How do we do that?

### Unattended Database Maintenance



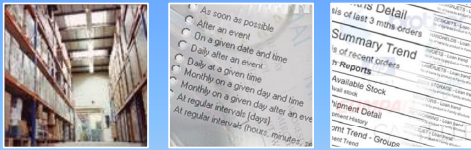
- High usage distributed sales office system
- No local IT support
- Fully automated reload & backup
- Fully automated data replication

A Visual Basic and Oterro CRM system that is available at offices distributed nationally and internationally, each with heavily used databases and minimal local IT support.

The system carries out daily local database backups & reloads and, more importantly, performs the necessary processing to replicate data around all offices before start of business each day.

How have we been doing that for the past nine years?

### Automated Report Production



- High value add – outsource service
- Extensive client reporting requirements
- Fully automated report generation & distribution

An organisation that provides a high value added, outsourced service to major IT vendors. They have a range of client contacts in different companies who regularly need to be kept up to date on the performance of this service. Someone, who could be doing better things, used to spend several days a month preparing and emailing reports.

Now this is done automatically and the clients can choose, when and how this information is delivered.

How does it work?

### Data Loading & Processing



- Business generation data feeds
- Time sensitive data
- Immediate automated processing, matching and sales lead generation
- Ready at the start of each business day

Users of a specialised business development system purchase weekly data files of research & live opportunity data from a variety of research houses.

This information is published 'out of hours' by suppliers, either for download or email delivery.

The information needs to be captured, processed against existing opportunity data and be ready for use in the client's system at the start of each business day.

How can that be achieved?

### Message Consolidation



- Multiple inbound voice messaging
- Unattended operation
- Timely and discrete notification
- Internationally available

Key partners in a professional consulting firm require regular updates and notification of their inbound voice messages (from a variety of sources).

This information is sent to them reliably, in a timely and discrete manner via GSM SMS text messaging 'where ever' they are (nationally or internationally).

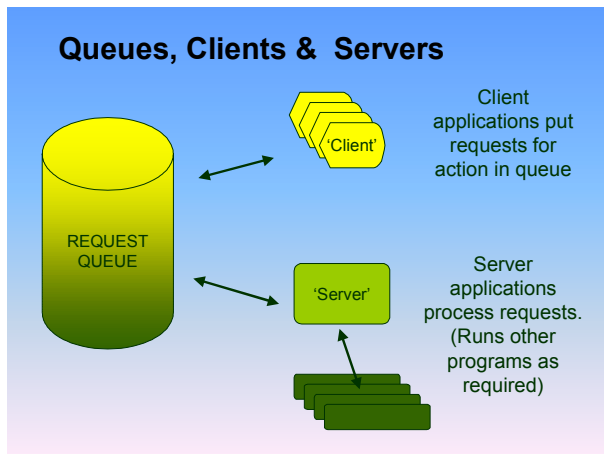
How?

## A Recycled Design

All of these snippets of systems share a common design theme. They are based on an idea that was presented at the 1992 R:BASE Explosion, (*The Network that Never Sleeps*<sup>1</sup>).

Key elements of the theme are that a “queue” is used to allow instructions to be passed between cooperating programs. “Client programs” request actions that are then carried out by “server programs”. This has the advantage of allowing the cooperating systems to carry out actions in parallel. It also provides shared access to a common processing resource.

In spite of its heritage, the design theme has stood the test of time and is probably more useful and effective today than it ever was.



An important point to note about this approach is that the clients (the programs or parts of programs that want an action carried out) would typically be located on different machines to the server or servers that carry out the actions. Depending how the queue is implemented the machines could be on same LAN or on opposite sides of the globe.

Another facet is that the “server” application itself may call on other programs (usually on its own machine) to assist in fulfilling the action requested by the client.

This approach can also be a great aid to rapid deployment. It is quite possible to build a general-purpose queue that uses component software in the client to make the requests. The requests are then carried out by a standard “server”. This then uses additional purpose built programs to carry out any application specific actions.

We will examine this in more detail as we look at the core elements of this design.

## A Recycled Design

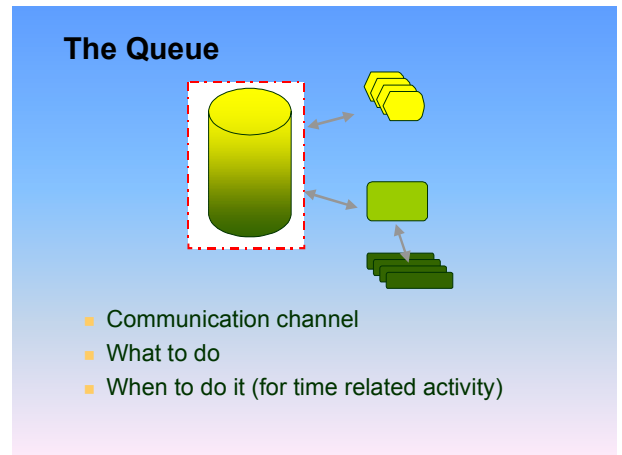
- Shared design approach
- Presented at the 1992 R:BASE Explosion
- Key Features
  - Use of Queues
  - Data as an instruction
  - Clients request actions
  - Servers carry actions out
  - Workload distribution
  - Resource sharing

## The Queue

The “queue” is the main channel of communication between the clients and servers. In essence each entry in the queue is an instruction to a server to carry out some form of action.

Depending on the implementation, the “queue” can take a number of formats:

- **Standard Queue** – this is where the first item to enter the queue is processed first (first-in, first-out).
- **Stack** – this is where the last item to enter the queue is processed first (last-in first, out).
- **Sorted List** – this is where items are given a key on entry to the queue, they are processed according to value of their key, or a sort order based on that value.



In the vast majority of examples we are considering, the queue is in reality a “sorted list”. If we use a date and time entry as the key to the list it enables the client to request when a particular instruction should be carried out. This is the approach we take for our general-purpose queue.

### Queue Structure - Basics

RequestID (Integer)	ReqStat (Text 9)	StartTime (DateTime)	TaskType (Text 16)	Schedule (Text 16)	SchInfo (Text 42)	SchTime (Time)
20000009	Ready	19/04/2002 09:00	REPORT	DAILY	Fri	09:00
20000208	Ready	08/04/2002 03:00	RELOAD	DAILY	Mon, Tue, Wed, Thu, Fri, Sat, Sun	03:00
20000209	Ready	01/05/2002 06:00	REPORT	MONTHLY	1	06:00

- What to do (TaskType + supporting table)
- When to do it (StartTime)
- How to reschedule (Schedule, SchInfo, SchTime)
- Further information (Descriptions, report names, email addresses, run statistics etc...)

In this case the queue is simply a table. The core queue columns are:

- **RequestID** – Our unique identifier for the request.
- **ReqStat** – A simple status for the request, for example this tells us whether it is ready to be actioned, currently active or should be ignored.
- **StartTime** – This confirms the earliest time (UTC/GMT) that the request should be actioned.
- **TaskType** – The task type confirms what action should be carried out. This is used in

conjunction with the “TaskInfo” table to confirm what a server should do to carry out the action.

- **Schedule, SchInfo & SchTime** – These columns are used to confirm what, if anything should happen to the request after it has been processed. They are used by the server to reschedule the request. This allows requests to be carried out at a variety of repeated intervals.

This core queue information is typically supplemented by additional items such as user entered descriptions, time of last processing and ‘trigger’ information to allow ‘events’ to be raised that themselves can be used to schedule further requests.

Two further tables are also used to support the basic queue structure:

- **TaskInfo** – As mentioned above, this table details what a “server” should do to fulfil a request. Typically, along with a description of the “task”, this confirms how the task is started and what it may produce. For example this may be the name and location of an R:BASE command file or the command line to invoke a custom program that produces a file that should be returned to the “client”.
- **RequestInfo** – In order to run a particular “task” and process any output it may produce, additional information is frequently required:
  - **Environment Details** – For an R:BASE command file, for example, there may need to be a number of variables declared and assigned that are expected by the command file. Information is needed (the variable name, type and value) to allow these variables to be set up before the command file is run. For a Crystal Report, details will be needed of the actual report file to run, the database connection that should be used, the SQL required for the selection and the required format of the resulting report file.
  - **Return Details** – Many requests involve the production and return of some information (a report for example). As well as just saving this information as a file somewhere on a server, email addresses can also be held that are then used to return the results to the user who made the original request.

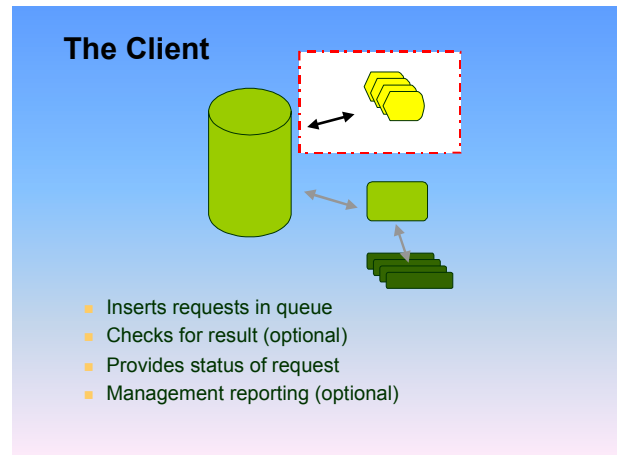
In addition to the above, further support tables can be implemented to hold statistical and management information about requests that have been carried out.

## The Client

The “client” is the part of the design that most users see. In its simplest form, all it needs to do is insert a correctly formatted entry into the queue.

Depending on requirements, the “client” can be used to provide additional features to enhance the benefit users receive from the facility. It can be used to check for the completion of a user’s request and provide an estimated completion time for a request that is currently being carried out.

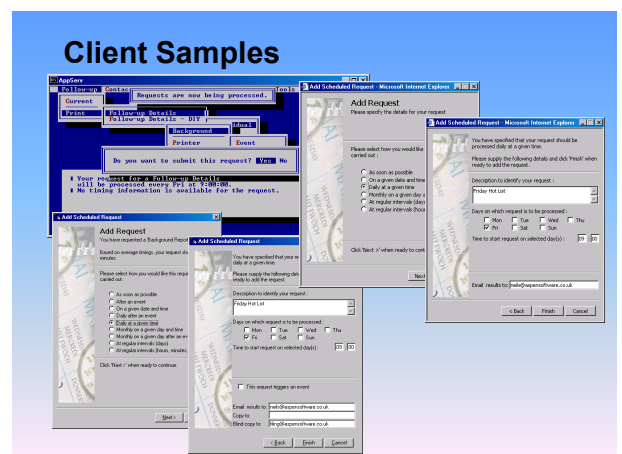
“Management users” can also be given access to reporting capabilities to show the type and frequency of the requests being made.



From a development point of view, perhaps the greatest power of this particular design theme comes from the “decoupling” of the programs used for the “client” and “server” elements.

When new technologies become available, client programs can be deployed that use these technologies and insert requests into the queue. Yet the existing servers will still perform the requests!

As illustrated, clients produced in character mode, Windows and web browser interfaces can all be used to insert requests into a queue.

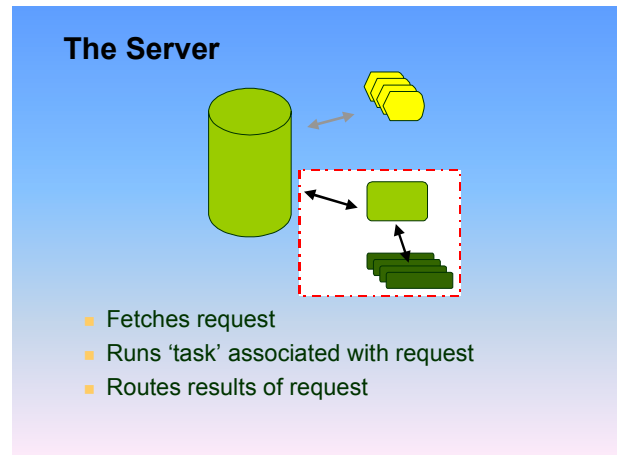


In the examples shown, Microsoft Visual Basic and Oterro have been used to produce ActiveX controls that can be used in a Windows environment to initiate requests. These ActiveX controls also use custom DLLs that can be incorporated into Microsoft IIS ASP web server applications to enable requests to be made from a web site. (These techniques are discussed in more detail in the Millennial R:BASE Developers’ Conference presentation – *Recycled Oterro*<sup>2</sup>.)

## The Server

The “server” is probably the most technically demanding element of the design and the part that fewest people see.

At its simplest, the “server” is a stand-alone program (an executable, or even R:BASE running a specially constructed command file). The “server” fetches the request from the queue, does what is required to ‘action’ the request, ensures the results get back to the original user and, if necessary, reschedules the request to run again.



For a successful implementation of this design theme, the “server” must be robust and available to carry out requests that are passed to it. If one “server” is being used to service several hundred clients then failure at this point can have dramatic consequences.

If required, the server may be implemented as an R:BASE application. For the period 1992 – 1996 we successfully used an R:BASE for DOS version of a “server” – which had as its clients programs produced with Visual Basic and the SQL Engine for R:BASE (the precursor to Oterro).

Today, our preferred choice of environments for a general purpose “server” is Visual Basic and Oterro. As will be covered below, this is predominantly due to the level of access that Visual Basic gives to the Windows operating system in order to control other programs that the “server” may use.

### Server Processing

Whatever is used to develop the “server”, when working with a time based queue, the technique for finding out what to do next is the same.

#### Check

On a regular basis (every few seconds) the “server” checks to see whether a request needs to be carried out, eg:

```
SELECT RequestId, TaskType FROM RegularQ
WHERE
StartTime < DATETIME(.#DATE, .#TIME)
AND ReqStat = 'READY'
ORDER BY StartTime
```

#### Fetch

Information needed to carry out the request is then retrieved based on the value of ‘TaskType’ placed in the request. Some requests, such as general housekeeping and result purging for example, would be internal to the “server” and would not require the use of any external support programs. Typically though, most requests in a general-purpose environment would need the services of additional programs (as defined via ‘Tasktype’) in order to carry out the request.



**Action** Assuming that an external program is going to be used to carry out the request the “command line” information fetched for the ‘TaskType’ would be used to start the program. Typically, to ensure that any files produced by the external program are available for return to the original user, the command line would also be altered to pass relevant file names.

The program is then started using Windows API call “CreateProcess”.

**Wait** The “server” now waits for the external program to complete. It loops using the “WaitForSingleObject” API call to check whether the external program is still running and the “Sleep” API call to suspend itself before it loops again.

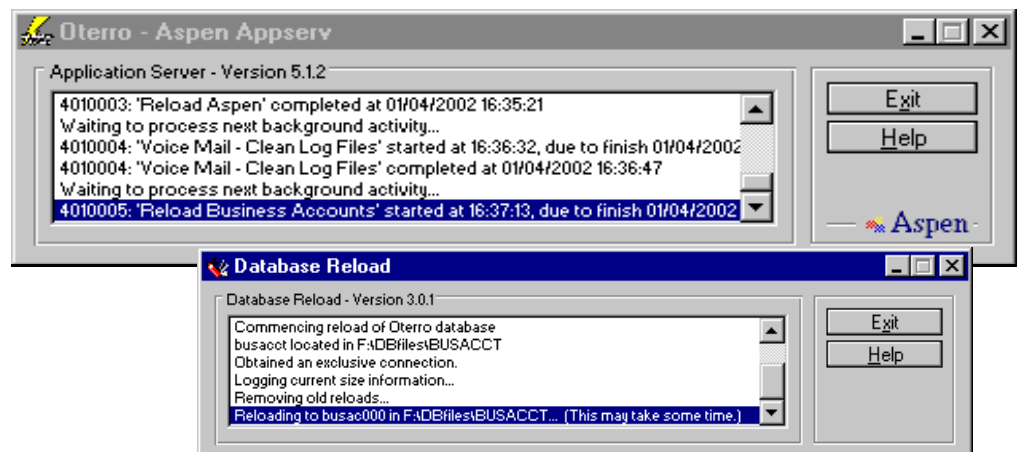
**Clean Up** Once the external program has completed, any resulting files may be emailed to the required recipients. Other details of the results of the request (eg actual processing time, log files etc) may be stored for management reporting.

**Reschedule** If the request is to be run again, the scheduling information that was placed in the queue can now be used to derive the next date and time at which it should run. The ‘StartTime’ for the queue entry is then updated to reflect this new date and time. This rescheduling can be as flexible as required, for example allowing items to run on particular days of the week or particular days of the month. It can also support items that are run on a periodic basis, eg every ‘N’ days or every ‘N’ hours minutes and seconds.

Introducing a notion of ‘event based’ scheduling can also produce even more flexibility. This means that when one request is finished, an event can be “raised” which then allows requests that have been waiting for that event to also be run.

**Repeat** On completion of the rescheduling, the “server” may optionally wait for a number of seconds (to avoid network overloading) before repeating the cycle again.

This illustration shows a general-purpose “server” waiting for completion of an external program that is being used to reload an R:BASE database.



## Key Benefits

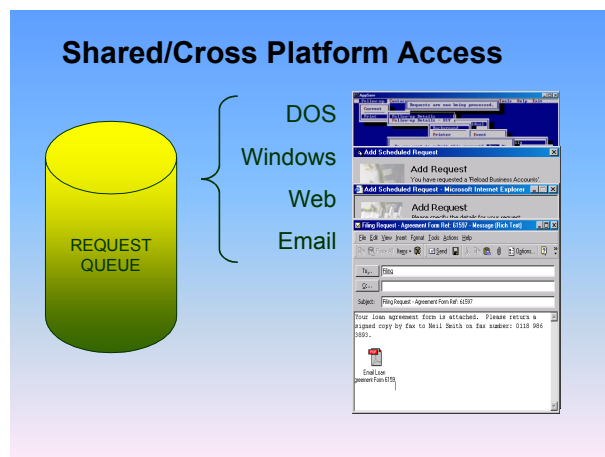
From a user's point of view, the major benefit that a design theme such as this provides is the empowerment to automatically initiate key business processes. When combined with reporting requests, users can automatically distribute key business information to interested parties without the need for further IT development.

For a developer, the simplicity of the design and modular use of "client" and "server" programs leads to a flexible and robust solution. If a general-purpose approach is used, business specific functions can be added simply by the development of purpose built programs for use by "servers". By their nature, and thanks to the general-purpose "infrastructure", these additional purpose built programs tend to be more straightforward and quicker to develop.

By offloading the processing that would normally be carried out on a users PC to the "server", time is no longer lost while a user's PC is slowed waiting for the production of some complex task. Conversely, since any machine that can run the "server" program can also process requests; user's PCs may also be used, out of hours, as "servers" to carry out requests.

## Key Benefits

- User empowerment
- Simple & flexible
- Robustness
- Quick deployment
- Management & load control
- Shared access to common 'back end'
- Cross platform support



One of the greatest benefits of the approach comes from its use in a cross platform environment and the access it gives to a common "back end" set of "server" programs.

As we have seen, character mode, Windows and browser based clients can all provide access to a request queue, and hence the "server". As we shall see when we discuss extensions to the theme, we can also implement "email based" request entry into the queue – for even greater flexibility.

## Building Your Own

As we have mentioned, “clients” and “server” programs in this recycled design can be produced with a number of programming environments.

Whatever environment you may want to use to implement this theme. A key area of attention should be how your “server” programs are implemented. If they are completely self-contained, or if they use a general-purpose approach and run additional supporting programs then you need to check for any “bad behaviour” that might take place when the “server” is running. Typically these programs run on PCs rarely visited by human beings, so a “server” program with an open dialog asking for someone to press a key to continue may wait for a very long time before the request is finished. Your “server” and supporting programs should be able to run entirely without any user intervention.

Other things to consider, particularly on the machines used to run the “server” programs, are how to cope with the results of very long periods of unattended operation. Can your main “server” run for many months? Program memory leaks, in particular, can bring a PC to a halt after periods of sustained operation. What happens when a “server” machine restarts after a power failure? How do you make sure your “server” programs are restarted?<sup>3</sup> What happens to requests that should have been processed while the power was down?

User empowerment is a wonderful thing but if someone can make a request to produce a report for each of the two million transactions in your database then they probably will! If they are allowed to give their request a free format description then they will probably (intentionally or unintentionally) use every “illegal” character imaginable and somewhere along the line try to put a 1000 character text string into a field meant to hold 10 characters! Both your “client” and “server” programs need to have strategies to deal with these issues.

Another challenge is recreating the environment for the “server” program to carry out a request. Frequently, a request will need to be supplemented by additional information (eg for a report; what report should run and what selection should be used along with other user supplied information required in the production process). This information will typically need to be gathered by the “client” at the time the user makes a request, stored in a supplementary queue table, and rebuilt when the “server” program actions the request.

A similar problem also needs to be addressed if the request runs an R:BASE command file, any variables set up when the user issued the request from the “client” program may also need to be regenerated when the “server” program actions the request. (One technique to achieve this to carry out a “SHOW VAR” redirected into a file and then use a program to convert this to a sequence of “SET VAR” commands that is then run to initialise the variables.)

If you are giving your users the ability to request and schedule reports that include date based selections, you will also need to give them a facility to select date based information on a relative basis (for example ‘Today’, ‘Yesterday’, ‘This Week’, ‘Next Week’, ‘Last N days’, ‘Next N days’ etc). This will ensure that if their reports are meant to cover a moving time window, then the correct time frame will be selected whenever the report is produced.

### Building Your Own - Issues

- Bad behaviour!
- Expecting the unexpected
- Exception notification
- Recreating an environment
- Recreating an R:BASE environment
- Date based selections

## Other Choices

If you just need to make a task run at predefined times, then two task schedulers are available within Windows. If you are using Windows NT/2000/XP, a command line utility “AT” and its “friendlier” graphical interface “WinAT” are available to schedule tasks to run on the PC or if you have appropriate privilege, a remote PC.

Microsoft also has another scheduler that provides a common scheduling interface across Windows 9x/ME/NT/2000/XP. This also includes a COM object interface that may be manipulated programmatically.

One of the challenges that you may encounter when using these facilities under Windows NT/2000/XP is that when a task runs underneath the scheduler, it may not run with the user privileges you would expect if you were to run the task interactively!

For programming in a Visual Basic environment, Microsoft provide a Message Queuing product (MSMQ) which provides a full, robust and sophisticated queuing environment that can be accessed by the cooperating programs you develop.

As you will see in “How Do I Find Out More?” there are also a number of third party schedulers available, indeed some anti-virus suites also include a scheduler that you can use to set tasks to run on your PC.

## Other Choices

- Operating system support
  - AT (& WinAT)
  - Windows Task Scheduler
  - Microsoft Message Queue
- Third-party schedulers

## Extending the Design

The design theme we have been reviewing is however, much more than just making a program run on a PC at a given time. As we have hinted, we can extend the design to given an even more powerful solution.

Particular areas that can be extended are the location of the queue and how it is structured. As we will see below, we can extend a queue to include the mailbox associated with an email address. In this case, a “client” can become anyone capable of sending an email to that address! For specialised applications, rather than adapting a general-purpose structure, the information held within the queue can also be modified to reflect the needs of the cooperating “clients” and “servers”.

The following real life examples show what can be achieved by extending the design:

### Automated “Filing”



- Complex service provider
- Multiple fax and email interchange to confirm
- Automated Exchange filing and linking to mainstream database application
- Regularly scheduled “server” scan of email queues

A service organisation uses email & fax to confirm a complex service provision. There may be an interchange of several emails (and faxes) between several interested parties before the service may be confirmed.

All of this correspondence needs to be filed automatically, be readily accessible from their mainstream database application and, in certain cases, automatic updates also need to be made in the database application to confirm that a response has received.

To achieve this, a general-purpose “server” program runs an external custom Visual Basic “Filer” program at 15-minute intervals during the “business day”. Driven by a list of Microsoft Exchange mailbox names held within the database application, the “Filer” program scans the mailboxes, identifies the service contract to which any email applies, carries out any applicable updates to the service contract information held within the database, moves the email to a structured public folder area within the Exchange Server and then creates a “link” from the database to the filed item within the Exchange Server.

When in the main database application and viewing details of a particular service contract, users may access and “open” the filed email with just one click.

### Extending the Design

- Queue location (eg mailboxes)
- Queue structures (specialised data)
- Examples
  - Automated “Filing”
  - Bulk Email Management
  - Telephone Switch Access

### Bulk Email Management



- Highly personalised email & attachments
- Manual approval process
- Automated production & despatch
- Notification of completion
- Queue based email list with immediate production and despatch by "server"

The administration team of an IT service provider has to review lists of particular contract contacts in order to approve the transmission of sets of personalised emails.

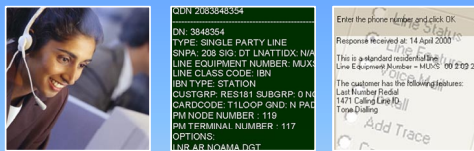
The emails themselves are highly tailored to the recipient and may also include complex documents (again extensively personalised) as attachments.

However once the administrator has approved a particular contact, they do not need to play any

further part in the email and attachment production - they want to get on with other jobs without their PC being slowed down. And yet.... they do want to know when all of the personalised email has been despatched.

To achieve this, once they have finished reviewing their list, the program that has been used to carry out the review process, uses an ActiveX DLL to insert a "one off" request for an email transmission into a general-purpose queue. Elsewhere on the network, a general-purpose "server" program carries out the request by using a custom Visual Basic "Bulk Emailer" program to process the list, generate the attachments, send the emails and update the main database application. On completion, the general-purpose "server" emails the "Bulk Emailer" log to the user to confirm the operation.

### Telephone Switch Access



- Telco call centre
- Interrogation of telephone switches by call handlers
- Removal of specialised switch terminals
- Transparent queue based access via "client", "server" and specialised database queue.

A telephone company needs to give their Customer Services Representatives (CSRs) access to highly expensive & complex 'switch terminals' to interrogate telephone switches in order to answer routine customer enquiries about their telephone connections.

These 'switch terminals' are expensive, use an unfriendly command line interface and are potentially open to abuse (eg setting up free calls for a subscriber).

The need for the switch terminals has been removed and CSRs now have an easy to use application that provides a safe and efficient way to get the required information from the telephone switch (with none of the drawbacks).

This is achieved through the use of a specialised queue approach. A customised Visual Basic client program is used by the CSRs to insert the request and telephone number into the queue. Several hundred miles away, a specialised "server" processes these requests and then selects a further specialised program to interrogate the telephone switch and return the appropriate information. The "server" translates the information into a form understandable to the CSR and updates the queue entry with these details. Meanwhile, the "client" program waits for the response to be entered into the queue, displaying it to the CSR when it is available.

## Conclusions

What can we conclude from our recycled designs?

Generally I believe we should all be open minded and willing to consider and use the ideas of others. Blended with our own experiences we can all adapt these to our own ways of working to give even better results more quickly. Also, as time and developments progress, we shouldn't be afraid to revisit past solutions and see what we can adapt and reuse from those experiences.

Specifically, I hope some of the things we have discussed will encourage you to consider using your database to implement queues between cooperating programs. It may just be the solution you need to solve a particularly difficult problem.

Finally, whatever you are going to do, have at least one way of easily scheduling programs available in your toolkit – as we have seen, you will be surprised how useful it can be.

### Conclusions

- Adapt the ideas' of others
- Adapt your own ideas
- Consider queue based approaches
- Have a "scheduler" in your toolkit

## How do I find out more?

To find out more about the themes we have covered you may want try the following resources: (Please remember that this is not an exhaustive list and links may change over time. Also the inclusion or exclusion of any third party scheduler in the list below does not indicate any form or recommendation.)

### Oterro & Visual Basic

- ✓ You can find out more from [www.oterro2000.com/Products/Oterro.htm](http://www.oterro2000.com/Products/Oterro.htm)
- ✓ Additional references can also be found from the paper available at [www.aspensoftware.co.uk/RBASE/otrecycle.htm](http://www.aspensoftware.co.uk/RBASE/otrecycle.htm)

### Windows API

- ✓ Information concerning the ‘CreateProcess’, ‘WaitForSingleObject’ and ‘Sleep’ API calls can be found from the excellent publication *Visual Basic Programmers Guide to the Win 32 API*, Daniel Appleman SAMS Publishing ISBN: 0-672-31590-4
- ✓ Other online sources include [www.allapi.net/apilist/apifunction.php?apifunction=CreateProcess](http://www.allapi.net/apilist/apifunction.php?apifunction=CreateProcess) (CreateProcess), [www.vbapi.com/ref/w/waitforsingleobject.html](http://www.vbapi.com/ref/w/waitforsingleobject.html) (WaitForSingleObject) and [www.allapi.net/apilist/apifunction.php?apifunction=Sleep](http://www.allapi.net/apilist/apifunction.php?apifunction=Sleep) (Sleep).

### Operating System Support

- ✓ AT & WinAT: [www.ntfaq.com/Articles/Index.cfm?ArticleID=15153](http://www.ntfaq.com/Articles/Index.cfm?ArticleID=15153) along with Windows Online Help
- ✓ Windows Task Scheduler: [www.iopus.com/guides/winscheduler.htm](http://www.iopus.com/guides/winscheduler.htm) and [www.microsoft.com/windows2000/techinfo/howitworks/management/task\\_scheduler.asp](http://www.microsoft.com/windows2000/techinfo/howitworks/management/task_scheduler.asp)
- ✓ MSMQ: [www.microsoft.com/ntserver/techresources/appserv/MSMQ/whatsnew.asp](http://www.microsoft.com/ntserver/techresources/appserv/MSMQ/whatsnew.asp)

### Other Schedulers

- ✓ Arcana Development - Arcana Scheduler: [www.arcanadev.com/scheduler](http://www.arcanadev.com/scheduler)
- ✓ Cypress Technologies – Autotask: [www.cypressnet.com/Products/autotask/autotask.htm](http://www.cypressnet.com/Products/autotask/autotask.htm)
- ✓ DWG Software Ltd – WinCron: [www.dwgsoftware.com/help/task\\_scheduler.html](http://www.dwgsoftware.com/help/task_scheduler.html)
- ✓ Java Scheduler: [www.jscheduler.com](http://www.jscheduler.com)
- ✓ RJ Software – Clockwise: [www.rjsoftware.com/ClockWise/scheduler.html](http://www.rjsoftware.com/ClockWise/scheduler.html)
- ✓ Splinterware - Windows Scheduler: [www.splinterware.com/products/wincron.htm](http://www.splinterware.com/products/wincron.htm)
- ✓ Unisyn Software – Automate: [www.unisyn.com/automate](http://www.unisyn.com/automate)

### Other Resources

- ✓ [www.asb2.com](http://www.asb2.com) for information about the Aspen Software building blocks that have been used in the production of the examples featured in this paper.
- ✓ [info@aspensoftware.co.uk](mailto:info@aspensoftware.co.uk) if you have any questions about this subject.

### References & Acknowledgements

- <sup>1</sup> Bob Hellriegel, Zytech Inc - “The Network that Never Sleeps”, excerpt from “MultiUser R:BASE” presented at the 1992 R:BASE Explosion.
  - <sup>2</sup> Neil Smith & Suzy Smith, Aspen Software - “Recycling Oterro”, presented at the 2000 R:BASE Developers’ Conference.  
(See [www.aspensoftware.co.uk/RBASE/otrecycle.htm](http://www.aspensoftware.co.uk/RBASE/otrecycle.htm))
  - <sup>3</sup> Microsoft Knowledge Base article Q253370 – How to Enable Automatic Logon in Windows.
- ✓ Automated report production “images” produced courtesy of Added Dimension Ltd [www.added-dimension.co.uk](http://www.added-dimension.co.uk).