The Minimum You Need to Know

About Service Oriented Architecture

By Roland Hughes

Logikal Solutions

ISBN 0-9770866-6-6
ISBN-13 978-0-9770866-6-5

This book was published by Logikal Solutions for the author. Neither Logikal Solutions nor the author shall be held responsible for any damage, claim or expense incurred by a user of this book as a result of its use or reliance upon.

Acknowledgments

Books like this, the planned closing of a series segment, come about because dedicated readers like yourself purchased, read and wrote about the prior books in the series. When that doesn't happen, a series segment simply gets left undone. To paraphrase the Bartles & James commercials: I thank you for your support.

Source Code License

Any person owning a copy of this book may use the source code from this book and accompanying CD-ROM freely when developing software for their personal use, their company's use or their client's use. Such persons may include the source code either modified or unmodified provided that the source delivered makes reference to the original author and is part of a fully functional application. It is expressly forbidden for anyone to post this source code on any bulletin board system, Internet Web site, or other electronic distribution medium without the express written permission of the author. It is also expressly forbidden to sell this source as part of a library or shareware distribution of source.

Users of the source code contained within this book and on the accompanying CD-ROM agree to hold harmless both he author and the publisher for any errors, omissions, losses, or other financial consequences resulting from the use of said source. The source code is supplied “as is” with no warranty of any kind expressed or implied.

Table of Contents

Chapter 1

Introduction

1.1 SOA – Where We Came From

For quite some time now, marketing shills have been preaching and selling SOA (Service Oriented Architecture). Vast quantities of money have been poured into "The Next Big Thing." Well, it is time to ask a few questions and rain on their parade.

What is the Service Oriented Architecture? This is a re-hash of an old business model that played itself out from the 1970s into the 1980s. Back then computers were multi-million-dollar investments. Small to mid-sized companies simply couldn't afford them. Programmers were scarce and qualified programmers even scarcer. Small to mid-sized companies could afford a few thousand dollars for terminals and special modems. Then they spent several more thousand dollars per month leasing time on computer systems. There were actually businesses that did little more than lease time. They would also sell consulting services and lease access to some of their applications.

Those of you who grew up with $20/month Internet access probably have a hard time imagining someone paying thousands of dollars per month for access to a computer system, but it happened. Commercial-class operating systems provided system and application run-time accounting. Among other things they would measure the amount of computer time, connection time, and IO that each user ID performed on the system. Every month the subscribing companies would be billed by the CPU second, connection minute and block of IO. It shouldn't take a rocket scientist to do the math here. One thousand customers paying $10K+/month to use your computer system paid off a $2 million price tag pretty quickly. Even 100 customers could make it viable if you were using the computer system to run your own business. You bought the machine on time and they made the payments for you.

Of course, the companies looking to lease time didn't have programmers (initially) and didn't have applications, so those time-share businesses got into the custom software business. They had proprietary data layouts that locked their clients into ever escalating support costs. It was like a drug addiction you couldn't kick. (Sound familiar to those of you using stored procedures in a commercial database?)

Mid-range computers came onto the scene. Many of the customers of mainframe time-share systems opted to buy a mid-range computer for about 6-12 months worth of their time share cost. Eventually, they too got into the custom application and time-share business, but not on as wide a scale.

Then came the PC. Promising to put computing power in the hands of the people without any regard for where that would lead. Vendors of hardware and software for these little boxes (for the time anyway) made money hand over fist. Competition became cut-throat, and soon carcasses of once proud vendor corporations littered the landscape. Companies had a rash of "desk drawer" applications that were now required to keep their businesses functioning. Very little could communicate effectively or exchange data reliably.

To stem the bloodshed, companies began focusing on networking technology. Multiple competing proprietary networking protocols emerged. File servers and network application servers sprang up all over the place. If you could afford the proprietary network interface equipment and the licensing, you could all use the same applications and share data (in theory).

Greed set in even worse than before. It was nothing to end up paying well over $1,000 for a network card to get a computer online. Sadly, you usually only paid about $3,000 for the PC to begin with. Somewhere, in my souvenirs from the past, I have an ISA-based BiSync modem that operated at a whopping 1200 baud. I paid just over $1,300 for it and it was a buy! The name brand version was around $2,500 as I recall. The great part about these proprietary networking systems was that they didn't talk to each other. Communicating across them required even more expensive bridging software and dedicated hardware. Even when you had the networking people configuring your machine in the office, you didn't always get network access. In their own weird way, the same barriers people complained about provided a nearly unparalleled level of security. Networks were so finicky to get working from the inside, hacking in from the outside was almost a myth.

Next came the TCP/IP standard, and the Internet. People wanted everything to connect to the Web. Hackers now had a standardized medium to exploit and they took to it with zeal. PC prices dropped dramatically and PCs started popping up all over the place. PC operating systems were (and still are) so unstable and riddled with security holes that every application and project had to have its own server. Servers started appearing around corporations like rabbits taking both Viagra and fertility drugs. Suddenly companies found they were spending as much on these "cheap" systems as they had been on the mainframes that took up less total space and consumed less power.

Enter the great server consolidation move. Completely disregard the fact your PC server OS is unstable, requiring frequent rebooting and patching, taking the applications off-line. Load virtualization software on one, and consolidate 8 or more into one machine. Uptime be damned! Now when you have to reboot you can take out 8 or more servers.

One problem still remained though. Each application running on a server was pretty much its own data island. Even when it stored its data in a database, that database was either on the same server itself, or the database wasn't the same one used by other applications. Management had cost cut its way into a very bitter pickle indeed.

Fear not though, the marketing shills had even more to sell them under the heading of "cutting costs." Management always listens intently to the "buy more to spend less" sales pitch. Repackage these applications with this magic middleware (for an additional fee) and you can use pieces of them, tying them together as you wish. They will all have a protocol to exchange data. You can use order entry from this company, inventory management from that company, and if you buy today, company three will sell you a piece of their picking ticket & truck loading module.

In fact, if you want to use it all from one vendor, it will work the best. No need to install their applications, simply lease time on their machines. Get rid of all these pesky servers. Let someone handle the updates and capacity issues for you. Vendor A has built a massive data center in a single location with redundancy upon redundancy all in the same 40 acres. After all, the world is one network now isn't it? Why should you keep buying all of this equipment, simply pay for the time you use...

Sounds kind of familiar doesn't it? Pay by the application module, computer use and storage. If it only ended there, it would be laughable. But it gets worse.

1.2 The Disaster We Are Headed Toward

What are they using to host all of this wonderful stuff? Racks and racks of PC-on-a-card servers. Heat is a bigger problem for them than it was for the old mainframes. Some companies are now looking into liquid cooling systems for those “personal computer” chips. Liquid cooling, just like some of the big mainframes used.

What operating systems are they running? The same crummy operating systems that were running at the client company. Security holes and patches are a seemingly daily if not hourly occurrence, but the client companies are used to outages. Outages have become a business standard.

What happens when one of those PC-on-a-card devices fries? Why, they load you up another one at some point. Can't really say how long the outage will be, but you're used to it already. Sorry about the transactions that were being processed, but you are used to losing those as well.

These faults can all be classified under the heading “Ignorance of a Cheap Mind.”

What cannot be classified under that heading, or even forgiven, is the packaging. These supposed forward thinkers have all bought into the idea that one massive data center located on X acres some place cheap is a good idea. What is even more unthinkable is that Congress is letting them do it.

I'm not talking about Congress being in control of business. I'm talking about Congress letting the National Security Agency take these “industry leaders” into a dark alley with ball bats and cattle prods to educate them on the finer points of economic and industry stability.

Let us assume for a moment that Congress is too busy getting re-elected to do anything useful. (Admittedly, not a big assumption.) These data centers on steroids are allowed to prosper. They have huge dollars behind them and drive out competitors so we have only a handful. Nah, let's shrink the number. We end up with only three, just like we did with auto

manufacturers years ago.

With greed coursing through their veins stronger than any narcotic on the market, they enter a cut-throat competition just like the PC manufacturers did. Pretty soon, they've driven the price down to just a couple thousand per month no matter how much you use. Talking heads at companies around the country start chanting “cut costs” and looking in the direction of these “service providers.” Yes, it's a financial Utopia all right.

In a need to differentiate themselves in this market, the “Big 3” buy up applications focusing on specific market segments. One provides the best of the best in banking software. Soup to nuts online checking, accounting, wire transfer, you-name-it-they-can-do-it banking product. Another ends up focusing on the complete solution for manufacturing. We are talking order entry, inventory control, sales force management, assembly line systems, shop floor control, the works. The last of the three ends up providing best in class shipping logistics. From truck loading to routing, and everything in between, they have modules to do it.

Ah, Utopia. They even develop a working relationship, so the manufacturing systems can use the shipping logistics modules and they both use parts of the financial modules. It truly is bliss. This is what marketing has promised and they finally delivered. Sort of...

No matter how you slice it, these providers spent heaps of money building up one location. Usually where there was really cheap power. One of them ends up getting built well within the blast radius of Mt. Rainier when it finally pops its top. (According to scientists within the next 20 years, but they guessed a little long on Mt. St. Helens, so it is probably at the shallow end of the 20.) Another gets built on a major fault line because power was half the price of anywhere else. The last gets built in the low cost real estate district found near the end of major airport runways because they got a sweet deal from the state and city governments.

It shouldn't be any surprise where this is heading. It doesn't matter how much power generation and redundancy each location has built into it, they will be taken out, either for months or permanently. A jet has engine trouble on take off and crashes, Mt. Rainier decides to prove the nay sayers wrong, and that 8.2 quake finally does originate from the dormant fault line just like geologists said it would eventually. In this day and age, we also have to deal with the possibility of a terrorist group either crashing a jet or driving a truck load of homemade explosives into the site. Let us not forget things like Katrina and the tsunami for the centers within a few miles of a coast line. Once we get to a “Big 3” or even “Big 4” situation with these

single location service centers, the destruction of one, by any means, cripples a nation and quite possibly the world economy. PC and UNIX people think only about loss of a single piece of equipment or power. Midrange developers think about the loss of an entire facility.

1.3 Why We Are Headed There

Where we are headed is not good because of the premise which started it all. Actually, there were multiple premises that lead us down this gilded path:

- Running core business systems on feeble operating systems was OK because the hardware was cheap.
- Using operating systems which cannot cluster completely is OK when they are cheap.
- Replacing existing systems with a flat monthly fee service was a great way of cutting costs.
- Building a single massive location to do everything and get economies of scale was a good idea.

Cheap hardware lead to the PCs on Viagra and fertility drugs scenario. Well, that, and unstable operating systems. Even IBM is running ads about servers being so numerous you cannot walk in a building. While they are an exaggeration, they aren't off by much. Oddly enough, IBM marketed the original IBM PC direct to businesses in the first place, so this is kind of like profiting from the sins of your past.

Cheap operating systems. Gotta love that. I remember companies and people complaining about spending $80 for DOS. At least with DOS, once you bought it, it was yours. Now Windows is moving toward the “As soon as you stop paying for the service agreement your *use license* expires” model with an initial rental price of around $300. Commercial flavors of UNIX had their own trappings when it came to license agreements. Now people hold up Linux as the be all and end all. Only one problem, none of these operating systems cluster correctly or well. They network, but do not cluster. Even the Tru64 version of UNIX, released by the company that gave the world its first (and still only) commercially viable clustered operating system didn't manage to get clustering working correctly in UNIX. The problem is, you can't be loyal to the UNIX installed base and cluster. The platform is missing critical pieces, and when those pieces are added, it won't be UNIX anymore, it will be a clone of OpenVMS with some command syntax changes.

Flat monthly cost. Management loves the flat monthly cost model. Indeed, Internet access providers and now cable phone providers are herding consumers towards that mentality. Management loves it because they now only have one thing to look at when they sit around in a talking heads meeting chanting “cut costs.” Leaves more time for the golf course.

A single massive data center. Now here is an idea straight from the 1970s. You have to be old to remember it. The classic IBM mainframe data center. Raised floor to hide cabling, route air conditioning, and provide a water trap. Bullet resistant glass wall so people could walk by and be amazed by the equipment. Two to three layers of fire protection. Multiple lock entry security. In some cases, even data center guards by the doors. Let's not forget the massive diesel engine generators parked right outside the building and in later years the room full of UPS (Uninterruptable Power Supply) equipment.

IBM was the originator of the “big iron” trend. It didn't cluster, but it did have it's own proprietary networking for communications. As the platform progressed it also had machine partitioning. This would let you divide up your “big iron” box so it looked like multiple mainframes to the outside world. Sounds a lot like virtualization doesn't it? Big iron was big bucks. They didn't bother with clustering technology because very few companies could afford to scatter a dozen or so of these around the country. Instead, they put forth what became known as the “moat and fortress” data center design. This lead to the bank teller bullet resistant glass plates and the other things we talked about. I never really did figure out what the moat was, unless it was the raised floor or the fact many companies were told to locate their data center on the second floor, not the first or the basement. They used to get put in the basement because the equipment was so heavy, then someone thought about the flooding issue.

DEC (Digital Equipment Corporation) focused primarily on the midrange market for mid-sized companies. These were companies that could afford to spend one half to two million for their entire data center and computer system. The companies knew they needed a computer system to compete and keep growing, but they couldn't bite the bullet for a multi-million-dollar mainframe setup. As the companies ran more efficiently with the computer system they kept finding more and more uses for it. For a time they bought bigger and bigger boxes, but the engineers at DEC came up with an even better solution. Buy more of the same and hook them all together so files, batch jobs, users and processing power could be utilized throughout a cluster of computers. As communications improved worldwide, these clusters, literally spanned the globe.

1.4 The Roots of SOA

Both DEC and IBM hit on the need for transaction management. IBM created CICS (Customer Information Control System) and DEC created ACMS (Application Control Management System). Both of these were and still are amazing products. ACMS had the most far reaching implications, yet few realized it. Sadly, much of the market missed it.

While the concept will be completely foreign to those coming from the PC and UNIX sides of life, these systems allowed you to both package functions for a Service Oriented Architecture and provided guaranteed execution. A side effect was that each SOA task you created with them had a messaging format that did not allow for data overrun or SQL injection like the current Web fiascoes. Yes, there are many third-party messaging systems these days making claims about guaranteed message delivery, but none will go so far as to guarantee its execution.

Because ACMS was the most far reaching and the one I have the most experience with, we will be discussing ACMS combined with clustering in this book.

1.5 Additional Reading

If you like the style and content of this book, or would simply like to learn more about application development on OpenVMS, please pick up copies of “The Minimum You Need to Know to Be an OpenVMS Application Developer” ISBN 0-97708660-7. You can follow it up with “The Minimum You Need to Know about Java on OpenVMS Volume 1” ISBN 0-9770866-1-5 ISBN-13 978-0-9770866-1-0.

At some point “The Minimum You Need to Know About Logic to Work in IT” will be available as well. This is not an OpenVMS specific book, but meant to be used as a textbook to teach those new to IT about logic. It appears today's graduates haven't the foggiest clue about it. ISBN 0-9770866-2-3 ISBN-13 978-0-9770866-2-7.

Chapter 2

Clustering

2.1 What is Clustering

Clustering, even today, is one of the most forward thinking concepts in the IT industry. It was so far ahead of its time, the market couldn't really comprehend it. Clustering requires that when a node joins the cluster, it honors and respects the consensus of the cluster. To start with, it must use the same UAF (User Authorization File) as the rest of the cluster. Access to cluster resources is controlled both by the standard (System, Owner, Group, World) protection settings provided by many platforms and ACLs (Access Control Lists).

ACLs require quite a bit of knowledge to implement. You cannot simply bolt them onto an OS which doesn't have native support for them. Their use and concept must be firmly entrenched within the OS kernel. Let me show you an example of a set of ACLs on a directory.

```
$ dir/sec dka1200:[000000]acl_test.dir

Directory DKA1200:[000000]

ACL_TEST.DIR;1             1/18       29-NOV-2004 17:37:24.04  [HUGHES]    (RWE,RWE,RE,E)
          (IDENTIFIER=GST_S,OPTIONS=DEFAULT,ACCESS=READ+WRITE+EXECUTE+DELETE)
          (IDENTIFIER=GST_RE,OPTIONS=DEFAULT,ACCESS=READ+EXECUTE)
          (IDENTIFIER=GST_RO,OPTIONS=DEFAULT,ACCESS=READ)
          (IDENTIFIER=[*,*],OPTIONS=DEFAULT,ACCESS=NONE)
```

The series of RWE letters enclosed in parenthesis to the right is the standard (System, Owner, Group, World) protection I mentioned before. Read, Write, Execute and Delete are the access privileges you can use for each class of user. If a user is missing one of those letters, it cannot perform that function on the file or directory. UNIX and its derivative works do not implement four levels, so they fall a little short from the start.

Take a look at the ACL list though. Each identifier gets its own unique set of access privileges. The possible entries are READ, WRITE, EXECUTE, DELETE and CONTROL. Because this is a directory, the OPTIONS=DEFAULT clause is used to cause the ACL to be inherited by all files and directories created underneath this directory. The very last entry in this list tells the operating system to completely ignore the standard (System, Owner, Group, World) settings for anyone other than the file owner. The wild card entry [*,*] matches all identifiers.