The 25-Year History of Service-Oriented Architecture

By Erik Townsend

With illustrations by Mike Ronayne

July 17, 2008

Introduction

If I were to tell you that I'm the guy who invented *Service-Oriented Architecture* (SOA), you'd probably laugh even harder than you did when Al Gore claimed to be the inventor of the Internet, and rightly so! Major trends in the software industry generally only emerge after numerous visionaries and early adopters independently "invent" the same concept. SOA is no different. It emerged as a trend after several different pioneers had success with their own homebrewed versions of what was essentially the same idea. Although I can't claim to be the sole inventor of SOA, a fair statement would be that my colleagues and I had a lot more to do with inventing SOA than Mr. Gore had to do with inventing the Internet.

The common notion that *Gartner Group* invented SOA is simply absurd. An accurate statement would be that Gartner recognized the trend, and gave it a name (throughout the '90s I was promoting the same ideas under the name Service-Based Distributed Systems). Without a doubt, the publicity Gartner Group brought to the table accelerated the adoption of the trend dramatically, but the underlying approach to distributed computing was invented at least 15 years earlier.

Essentially, SOA is the technique of functionally decomposing the application functions of a large organization into a set of interoperable services that can be accessed through "network APIs". It is commonly believed that this is a "new idea", first conceived in the late '90s. That is simply absurd, and plenty of "prior art" exists in the public domain to disprove that fallacy. The buzzword SOA may be less than a decade old, but the style of distributed application software it describes is what I spent my entire career working on, beginning in the early '80s.

The propensity of engineers to believe that because they thought up an idea on their own they must be its sole and original inventor is just silly. I've already served as an expert witness in one patent lawsuit pertaining to SOA-related "inventions". Someone was naive enough to believe they had invented something new in this space in '97, and managed to obtain a patent on the idea. It was easy to disprove their claims citing work we did at Wells Fargo Bank five years earlier! And even right now, one colleague of mine is serving as an expert witness in yet

another very similar suit at the time of this writing!

What is even more striking is the way that people in our industry seem so unwilling to learn from history. There exists a lot of knowledge and wisdom from the early days about how to make SOA work. It's more about organizational structure than technology, and yes, there are case studies dating back to the '80s and early '90s that remain as pertinent today as they were when published. But few people seem to make the connection.

Origins of SOA in Discrete Manufacturing

I can only tell the story from my own perspective. I wouldn't be surprised if there are other pioneers out there who were working on SOA even earlier than the early '80s, and I'm certainly not foolish enough to claim to be the first inventor of anything. However, my colleagues and I really did innovate some very compelling technologies, and our work with Wells Fargo Bank in the early '90s was the first publicly disclosed example of an SOA case study (and success story) that I am personally aware of.

My own story begins in 1982 on the factory shopfloor of now-defunct *Digital Equipment Corporation*. DEC was a pioneer in the area of factory automation in discrete manufacturing, but faced a problem we used to call *islands of automation.* There were a number of different, incompatible automation solutions for problems ranging from automated material handling (the conveyor belts that move work around the factory), test and repair stations, automated IC insertion machines (later *on*sertion machines), and so forth. But none of these "islands" of automation were designed to talk to one another.

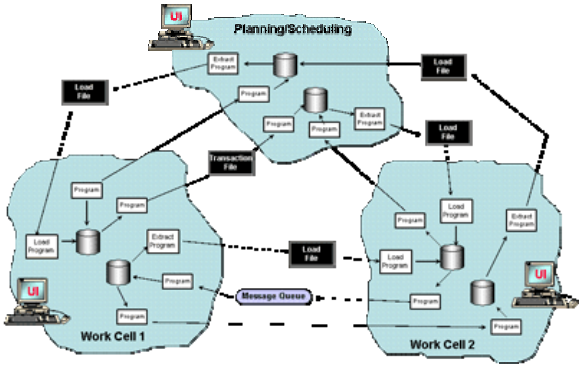

Figure 1 - Islands of Automation

DEC wanted a truly integrated factory. In other words, if a printed circuit board didn't pass a quality inspection test, DEC wanted the test station software to be smart enough to tell the automated material handling system to redirect that board to a rework station, and they wanted the information about the test history of the board to be visible to the technician at the rework station automatically. They also wanted to build in advanced rules and instrumentation so that they could optimize the flow of work-in-process through the factory.

INFINET: SOA circa 1983

In answer to this problem, the *INtegrated Factory Information NETwork*, or INFINET[1] was conceived in 1983. The goal of INFINET was to define an architecture that would allow monolithic "islands of automation" to be functionally decomposed into a number of discrete functions that could be accessed using middleware as a sort of network API. In other words, SOA. Of course robust middleware like *Web Services* hadn't been invented yet – in fact the Internet itself had yet to be commercialized and the Web hadn't been invented yet. So we had to start from scratch.

[1] INFINET was an internal project name at DEC, and was not related to a company of the same name that emerged later in the '80s.

Over 40 people worked in the INFINET group at DEC, and it would be impossible to name them all here. Steve Benson was our group manager, and a team of "three amigos" were the principal authors of the INFINET Architecture: Mike Ronayne (who also illustrated this article), Melissa Dever, and Mike Harden. My own role (as an outside consultant to DEC) was in developing the first home-grown middleware used to interconnect the various shopfloor application systems being developed by others in Steve Benson's INFINET organization.

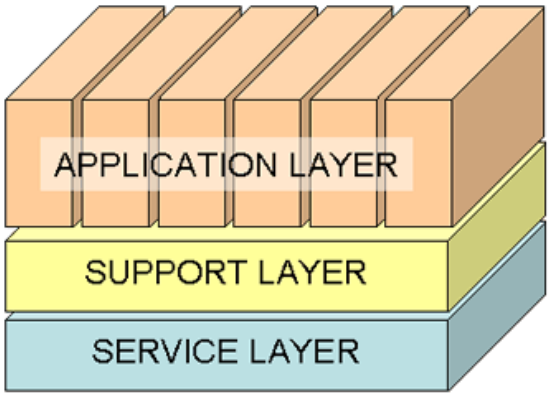

Figure 2 - INFINET Distributed Application System Architecture, circa 1983

At the core of the INFINET initiative was the *Reference Model for Production Systems*, which had as its central feature a "cube drawing", which we jokingly called Rubik's Architecture. (The Rubik's Cube puzzle was quite popular at the time).

The names have changed: In those days we referred to the basic computing infrastructure (hardware and operating system) as the *Service* layer. The *Application* layer represented the discrete application server functions, which would be called *services* in today's nomenclature.

Of course, in order for application services to talk to one another over a network, you need middleware. There was no commercial middleware in those days, so we had to invent our own. We called this the *Support* layer of the INFINET architecture. Keep in mind, this was all circa 1983.

The INFINET "Support Layer" was essentially a custom-made version of what is now called middleware. Mike Ronayne was given responsibility for developing middleware to support the INFINET application developers. As an outside consultant to DEC, I was Mike's primary technical resource at the time.

Homegrown Middleware

In 1984, I personally invented one variant of what is now known as *Message Queuing* middleware. I also "invented" what is now called *Publish and Subscribe Addressing* at the same time. I was oh-so-proud of myself for thinking up these brand new ideas that nobody else had ever thought of. Or so it seemed at first.

Sadly, my proud exuberance only lasted a few weeks, until I discovered to my shock and dismay that another smart guy in a different part of DEC had the same idea, and was already working on his own message queuing middleware tool. Both of our efforts were supporting internal projects, and DEC had no middleware product offerings to its customers at the time.

By 1986 we had discovered no less than 17 separate, completely independent examples of yet another group inside DEC "inventing" message queuing! Each independent "inventor" seemed to go through the same process of denial and emotion when they discovered that they were neither the only person nor the first person to come up with the idea for message queuing. I don't know of any greater example of *Not Invented Here Syndrome* than the message

queuing wars that went on inside DEC in the mid '80s!

We knew that "homegrown" system software was a bad idea, so we started lobbying DEC's Networks and Communications group to develop and market a message queuing product. This was before TIBCO (a message queuing middleware company) was launched, so DEC had every opportunity to be first to market. Sadly, to call DEC's "NAC Marketing" group incompetent would have been unduly complimentary. We were repeatedly told that "customers don't want big networks", and that file transfer and remote login were the only network functions that had any commercial value.

For over a year, a big part of my job as a consultant to DEC was co-chairing a committee chartered to try to understand the requirements of all 17 different redundant message queuing efforts, and propose a consolidation plan. This eventually led to funding to productize one of the 17 internal non-products (called PAMS), and the DECmessageQ product was born.

Shortcomings of Message Queuing as SOA Middleware

Ironically, by the time we had won the long battle to get DEC's management to wake up and realize that there could be a commercial market for middleware, our own experience in the INFINET applications group had led us to realize the shortcomings of message queuing as a middleware paradigm. The primary issue was that our MQ tools had no formal interface definition (message formats were essentially determined and documented by source code of the sending program). By the time DEC finally productized DECmessageQ, we were ready to abandon message queuing, and had begun searching for a new approach that would allow distributed communications interfaces to be defined separately from the programs that communicate using them. We called this idea *formal interface definition.*

Meanwhile, the INFINET application group had made great strides. Using our message queuing middleware (which we called ISL, for INFINET Support Layer), they built truly integrated factory shopfloor automation systems that were deployed in more than 14 different manufacturing plants in 9 countries.

The Birth of ACA: A precursor to Web Services

By 1989, DEC's management was very slowly waking up to the need for commercial middleware products. Mike Ronayne and I became very interested in an initiative called the *Application Integration and Interaction Architecture*, later renamed the *Application Control Architecture*, or ACA. Mike Ronayne and I represented DEC's Manufacturing division in providing technical requirements for ACA, which we saw as a replacement for message queuing in the INFINET Support Layer.

ACA was to be the world's first and only object-oriented middleware product, so far as we knew. Of course that fantasy too would soon be shattered with the discovery that several other leading computer companies were working on object-oriented middleware strategies of their own designs.

Open systems and industry standards had become all the rage by then, so DEC's ACA group collaborated with numerous other companies to develop the *Common Object Request Broker* specification, or CORBA, under

the auspices of a standards body known as the *Object Management Group*, or OMG.

The CORBA specification itself was notoriously vague, and interoperability between vendors was very slow in coming. But the idea of an *Object Request Broker* – a sort of predecessor to Web Services – was long overdue.

Mike Ronayne and I were still at wits' end, however, feeling that DEC's management just wasn't seeing the outstanding opportunity to take our experience with INFINET (essentially, doing what is now called SOA), and bring it to DEC's customers. We were too naive back in those days to see the obvious opportunity to go do it on our own. We tried and tried and tried to get a rapidly declining company's management to "get it", but never succeeded.

Taking SOA to Market

I left DEC in 1991, after almost 10 years as a consultant, and founded The Cushing Group, Inc. The express purpose of The Cushing Group was to bring the style of distributed computing architecture that we had invented[2] out of the high-tech company sandbox, and bring it to market.

My personal slogan at the time was that we wanted to do INFINET-style distributed applications (i.e. SOA), "solving real business problems for real customers for real money". I wanted to launch my own middleware software development company, but couldn't access enough capital. So instead, The Cushing Group was set up as a niche consulting firm that specialized in building distributed application systems using the approach now known as SOA.

[2] Again, I don't claim to be the exclusive inventor of anything in software. All great ideas like SOA emerge from numerous independent co-inventions of the same ideas.

At the time, our preferred middleware technology was CORBA, and our company tagline was "*Business Solutions based on Distributed Object Computing*".

For the first year or so, the only business I could drum up was teaching the ACA Services (i.e. CORBA) programmer training course for DEC, and occasionally teaching a DECmessageQ class for them as well.

By 1993, MIT professor John Donovan had made a name for himself giving seminars promoting what he called *3-Tier Client/Server*, an idea that encompassed some of the concepts of SOA. His seminars were evangelical and extremely compelling to the business communities to which they were directed, despite the fact that his claims about the availability and reliability of middleware to support such an approach were dubious at best.

One of the many customers to see Prof. Donovan's seminar was *Wells Fargo Bank.* The executives from Wells Fargo who heard Mr. Donovan speak were quite impressed with the benefits he promised, so they asked their IT department to look into Donovan's Cambridge Technology Group and its middleware offerings, which were based on the DCE standard.

Wells Fargo concluded that the CTG tools weren't quite "ready for prime time", but they still wanted to pursue the general approach, and started looking for middleware alternatives. They found DEC's ACA Services (CORBA) product, and liked it.

SOA's First Case Study

DEC asked us to help them with an initial 3-month consulting engagement at Wells Fargo, which began in October, 1993. That project was the first commercial example of an end-user

customer (as opposed to a technology company like DEC) deploying SOA that I am aware of.

The real proof of the business case for SOA came in May of 1995, when Wells Fargo became the world's first Internet Bank. That project was completed in an unprecedented 60-day project cycle, because all of the *services* needed to build the banking web site in '95 had already been developed for another internal application in '93.

The fact that Wells Fargo became the world's first Internet bank in only 60 days was remarkable, and it got both Wells Fargo and The Cushing Group front-page attention in the trade press.

The Wells Fargo story stands out in my mind as the first really good case study to prove the competitive advantage aspects of SOA.

Considering the fact that the Wells Fargo project was well publicized at the time, I find it remarkable that it is almost never referenced by authors writing about SOA today. The full story of our work at Wells Fargo can be found in a white paper I wrote in 1996, which you can find here: Wells Fargo White Paper. A condensed version of that paper also appeared as the cover story in the February, 1997 issue of *Distributed Object Computing* magazine.

Fueled in large part by our success at Wells Fargo, The Cushing Group went on to do similar projects for several other Fortune 500 customers. By 1998 we had grown to 35 employees and had broken into 8-figure revenue territory. Our future looked bright, and in many ways we were poised to become the IBM Global Services of SOA.

An unexpected demise

Ironically, the demise of The Cushing Group resulted in a bizarre way from the Y2K craze, which pushed valuations of publicly traded IT services companies through the roof. Those companies all became acquisitive, in something of a miniature repeat of the conglomerate boom of the 1960s.

It was by pure accident that I met an investment banker on an airplane one day, who explained that because of Y2K mania, my company was worth between $30 and $40 million dollars! I kept explaining that we had absolutely nothing to do with Y2K, and he kept explaining that Wall Street isn't nearly smart enough to understand that, so it didn't matter. Suddenly, distributed systems architecture seemed less interesting to me! ☺

We got out because the getting was good. Sadly, the public company that acquired The Cushing Group lacked the managerial intellect to comprehend what they'd bought. Our employees were quickly fed up with the new organization, and most of them left within a year. Many of them went on to innovate the next generation of middleware products at BEA Systems.

I learned a lot from that acquisition, and next time I'll know that it's just as important for the company being acquired to conduct due diligence on the acquirer as it is for the acquirer to check out the company being acquired. Life's most important lessons come hard, and to this day I deeply regret allowing the company I worked so hard to build to be acquired by a company that didn't even comprehend what they had purchased.

Important ideas are lost when nobody has financial incentive to promote them!

Having sold our equity in The Cushing Group, Mike Ronayne and I no longer had any commercial incentive to publicize our work at Wells Fargo, and it appears to have been forgotten by the mainstream for that reason.

Mike and I can't help but roll our eyes as we see blog after blog assert that "SOA actually goes way back to the late '90s!!!" They're 15 years off base! But what's even more amazing is that the myriad lessons learned at DEC and at Wells Fargo seem to have been ignored or forgotten as well.

Encapsulating Mainframe Apps as Services is a 15-yr Old Idea!

The whole basis of our work at Wells Fargo was to encapsulate existing mainframe-hosted application functions and expose them on the network as CORBA Object Servers (a predecessor to Web Services). The re-use of the existing mainframe apps, and the techniques we developed to eliminate the need to re-implement mainframe-hosted legacy systems on modern platforms was one of the greatest innovations of my career.

I spent most of the '90s giving talks at trade shows about what I called *Service-Based Distributed Systems*, and we published several papers disclosing publicly our techniques for giving Mainframe-hosted legacy systems a whole new life by encapsulating their functions as services. I even developed and taught a course on the subject!

Yet despite the fact that we published that information more than 12 years ago, Oracle recently released a white paper you can find here: Unlocking the Mainframe asserting these concepts as if they were a new idea! I wonder if the authors of that paper know about search engines like Google? Their research seems incomplete to me!

SOA is all about the Org Chart!

What's really important is not that SOA is much older news that most people think. What's important are the lessons learned from enterprise-scale projects like the work we began in 1993 at Wells Fargo.

Most notably, our experience has been that success with what is now called SOA has more to do with the structure of the IT organization than the nuances of middleware technology.

Traditional IT organizations are structured around functional deliverables. One manager or director is put in charge of a specific project, and has her performance measured by the success or failure of that project. Another project is a different manager's responsibility. In this model, there is no incentive for collaboration or re-use of services, and each manager may feel tempted to "grow their own" in order to avoid dependency on a peer's output. The solution requires a fundamentally different approach to structuring an IT organization.

Giving credit where it is due, Dave Lohman and Eric Castain (our principal customers at Wells Fargo) were chiefly responsible for recognizing that SOA requires a different organizational structure. They were pioneers in building such an organization at Wells Fargo, starting way back in 1994. It took some experimentation, but they eventually perfected the art of shaping an IT organization to succeed in a service-oriented environment.

I recently held a 15-year reunion for the core group of principals involved in the original 1993 Wells Fargo project at my summer home on the coast of Maine. We all had a good laugh about how the major consulting firms are now publishing white papers with "new ideas" that we actually developed more than a decade earlier!

I encouraged Eric Castain to write a paper about organizational structure and other managerial implications of SOA, but Eric has advanced to the position of Senior Vice President in charge of Enterprise Architecture at Wells Fargo, and keeps a very busy schedule. Sadly, Eric was unable to commit his time to such an effort. That's a real pity, because in my view he is easily the most qualified person in the industry to write on that subject (Dave Lohman has long-since retired).

Conclusions

It's interesting how information flows in the computer industry. Our original 1996 white paper about our work with Wells Fargo was enormously popular between its publication in 1996 and the acquisition of The Cushing Group in 1998. But that was because we had a financial incentive to promote it actively.

The information the Wells Fargo paper and others we wrote since contain is even more topical and important today, now that SOA has seen large-scale adoption. Yet it is completely ignored because nobody has any incentive to promote it[3].

Meanwhile, papers like Oracle's piece on SOA and Legacy applications are promoting the exact same concepts, and the industry perceives them as new ideas! At least the good news is that we can now access all this information on the web, thanks to Al Gore who invented the Internet! ☺

About the Author

Erik Townsend was a prototypical "computer geek" as an adolescent, and spent most of his high-school years at MIT's Artificial Intelligence Laboratory, where he taught himself programming and got his first exposure to the Arpanet (precursor to the Internet). It was this exposure to the 'net as a teenager that inspired him to focus his career on distributed application systems.

The body of this article chronicles Erik's career as a consultant to Digital Equipment and the founder of The Cushing Group, Inc. After selling The Cushing Group in 1998, Erik took a few years off to pursue personal interests, and then became an active private investor.

Erik is now returning to entrepreneurship, because he is convinced that location-enabled mobile computing will be an immensely important paradigm shift, and the opportunity to apply innovation and creativity has never been better. He is currently evaluating a number of ideas for new ventures in the location-enabled mobile computing area.

[3] In case you're wondering, this current paper was motivated purely by personal disbelief and frustration about how history is being misrepresented. Mike Ronayne and I have moved on to the next frontier (location-enabled mobile computing), and no longer have a business incentive to promote our work with SOA. We just couldn't stay quiet any longer and had to set the record straight!