



# Autonomic QoS & Collaboration Systems: Helping Systems Help Themselves

Wedge Greene & Barbara Lancaster

LTC International

**Date: 07/09/2006**

**Article for Pipeline Magazine, Vol 3 - Issue 2**  
**<http://pipelinepub.com>**

### **About LTC International Inc.**

LTC International provides leading companies in the telecommunications and IT sectors with a unique level of service based on true subject matter expertise. Our Business Operations Architects® each have at least ten years of hands-on experience in service provider and IT intensive companies. Our consulting team has experience in all areas of business profit optimization, wireless and wireline communications, Internet services, as well as software and hardware planning, implementation and operations.

LTC has incorporated more than 1,000 years of first hand operating company and software application experience into our Business Management Toolkit. This comprehensive set of tools, guidelines, checklists, templates and training programs is designed to remove uncertainty and accelerate success for our clients.

<http://www.ltcinternational.com>



*Expect Results®*



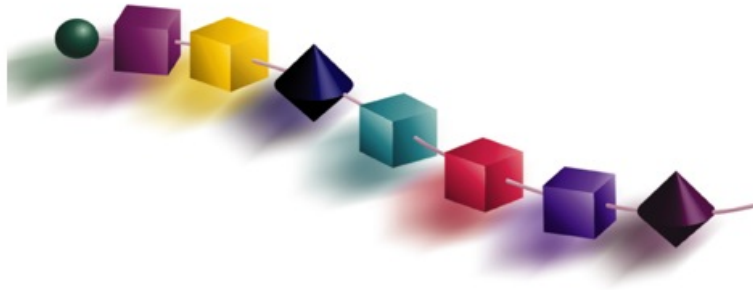
## AUTONOMIC QoS & COLLABORATION SYSTEMS

### Summary:

Providing management systems robust enough to not just measure but to also assure QoS will require adoption of new management paradigm. This paradigm, evolving from the TMF's NGOSS program and the work in SOA governance architectures, will rely foremost upon the collaboration of services for the delivery of management applications. Collaboration can also be extended to replace existing trouble ticket systems with a Groupware solution.



## Business Operations Architects ®



**Getting it right:** Proactive monitoring and response has been a core goal of Network Operations Centers (NOCs) for as long as I've been a part of telecom. The very first problem I was given when I moved into designing OSS systems was to “build a Frame Relay performance reporting system that will show users that data *really is* flowing over the pipes.” The business requirement was to preempt calls from customers who were confused about virtual circuit turn-ups. Soon after the first report was sent to a customer, our competitors jumped to match and better this ad hoc product feature; starting an escalation war that went way beyond original business requirements. So the whole data performance reporting industry was started in order to be proactive in satisfying customers. Rather ironic now, given how it turned out: thousand page performance reports headed by “top problem circuit” lists of dropped packets and low utilization – *that actually were not problems* (link to blog main page). In retrospect, we, as OSS systems designers got this solution attempt very wrong. But the business goal (better customer relations by being proactive) is still quite valid.

**Efficiency:** To this business requirement, we perhaps need to add an efficiency requirement. Certainly, huge masses of data sent to the customer are not good. During the peak of the Frame Relay product competition, a service provider had masses of enterprise server clusters fed from dozens of smaller poll servers, collecting information on a 100,000 circuit endpoints every 15 minutes. This data went into expensive, near-capacity-limit database systems, to generate massive email reports sent daily to customers. One operator had two entire rooms of a data center dedicated to reporting machinery for just one network product. Yet this still was a problem for customers, who had to hire someone to follow, interpret, and take action on these massive reports. Again massed data sent to customers is not good; where, a call to a customer contact that their circuit seems to have issues but “we are on it”, is very good. How do we design systems that make this accurate and feasible?

These issues were investigated in the TeleManagement Forum (TMF) [Service Level Agreement \(SLA\) project](#). Given the maturity of thought and technology at the time, producing common definitions and expectations was the best that could be accomplished. The SLA Handbook was a powerful artifact that enabled vendors, service providers, and customers to get on the same page when addressing problems. It was in the TMF SLA work group that early discussions on QoS and SLAs occurred. It was proposed that automatic notifications were possible and technically should be the standard.

**It is possible:** In 2002, a group of collaborating companies solved this difficult notification approach to QoS violations. This was demonstrated live as part of the **FineGrain NGOSS Catalyst**. That specific design and demonstration exercise was unfortunately packed with so many new solutions and efficiencies that even the people who watched the demo may have missed the implications of this specific advancement. The hands-off scenario of this project demonstrated *automatic recovery and notification of a customer, based on a QoS threshold violation warning delivered automatically from a customer site*. This autonomic process involved (1) a software probe attached to a video codex on an end user's application (2) sending an automated application performance issue message to a service providers system. Upon (3)

receipt by the providers OSS, this event message was (4) categorized as a potential SLA violation and therefore (5) fired off an automated business process, (6) driven by this specific customer's SLA policy, which (7) interacted with multiple element management systems and smart switches to (8) provision a new optical circuit [via TL1], (9) set up a new MPLS path [via XML CLI], and (10) rebuild Ethernet service [Java Jini Jmatoes interface] - then the process (11) automatically switched the customer over to the new data path. Finally (12) an email was automatically sent to the customer representative, detailing the proactive response that was successfully undertaken. The elapsed time for this entire process, acting upon optical/IP/Ethernet multi-layered network equipment, (discovery, to circuit switchover, to email delivery) was under 15 seconds.

**It is real:** It is certainly possible to build systems today which can deliver like results in real networks for real customers. Today, these are called **Autonomic Systems**. Originally, autonomic systems just referred to software that watched itself and fixed software or server problems before these impacted performance. Often this was accomplished by automatically reloading/restarting software or automatically switching applications to healthy server resources. Today, systems which do this are said to be characterized by **Virtualization**. However, the advent of **Pervasive Computing** extended the reach of networked event-reaction systems to devices and sensors 'at the edge.'

In use today, autonomic supply-chain logistics systems currently can rebalance delivery manifests and timing on shipments based on notifications of barcode reads on packages passing along a conveyor belt half way around the world. These refreshed, expected-delivery notifications are automatically sent to their manufacturing customers whose systems adjust assembly schedules for their factories. As this works today for international shipping companies, similar new software could do this for Telecom Service Providers.

**Technical solutions alone cannot cut it:** Even though the business goal is clear (proactive action for customers), it is still difficult for NOC executives and OSS designers to visualize the leap to this type of solution. For so long, the NOC has concentrated on indirect indications of problems. "Getting an alarm for a continuity problem and then launching a trouble response system process for fixing a down circuit" is only attacking a specific cause of a problem and not truly reaching out to the problem symptom experienced by the customer. This worked for black & white problems – service or no service. But even today, it is possible the customer has switched traffic to a competitor's network before a long outage will be fixed. With the advent of QoS, which implies the measure of and reaction on grey-scales of performance, the old approaches will not work.

I do not have space to go into all of the rationale for why existing systems will be overwhelmed by this issue of proactive response to QoS gradations, but listing a few major issues:

- Speed of required response (sub minute to sub second)
- Increases of scale associated with multiple kinds of threshold warnings and violations (being a order-of-magnitude multiplier on types of alarms)

- Integration of SLA definitions and other new OSS artifacts into existing systems and processes
- Embedded manual processes in linear work flow

I have long maintained that this requires a completely new architecture for OSS systems.

**Alternatives:** The [TeleManagement Forum's](#) New Generations Operating Systems and Software (TMF NGOSS) has described how to build OSS systems which are reactive to events, that is, are "event driven." NGOSS systems could adapt to provide autonomic responses to QoS issues arising from probes. [Prospero](#), the new alliance of OSS/J and the TMF NGOSS, will provide commercial, modernized approaches to adapt *existing systems architectures, following traditional OSS functional component descriptions* for "event driven" responses.

But first, service providers and customers must jointly agree to *extend the scope of what is considered a problem* – all the way to the user's application system. This is foremost a business visualization issue and not a trivial one. Technically, this is what is embodied in Pervasive Computing. It is being solved by close cooperation of instrumented hardware & software, wide-area networks and distributed computing. Standards for this are being worked within the [Global Grid Forum](#) (GGF) which is merging with the [Enterprise Grid Alliance](#) (EGA)

The maturity of web service standards now will let an application at an end customer location send a secure message to a service provider OSS system that indicates issues "as the customer will perceive them." [Organization for the Advancement of Structured Information Standards \(OASIS\)](#) is providing standards where any web service application can communicate with any other web service application. These standards can be extended from e-business to e-business (B2B) into Pervasive Computing architectures – thereby delivering autonomic proactive QoS. Good news is that EJB application server companies are putting these facilities inside their products. Yet while existing applications can be retrofitted to intercommunicate with web service messages, the current state-of-the-art architecture is Service Oriented Architecture (SOA). This requires a rethinking of application design requirements from "solve all the business issues in an integrated package" to "provide one business service to any other authenticated service which needs it".

**Coming together:** A convergence of architectural design is occurring among the TMF NGOSS, OASIS, W3C, GGF, and the Jini community around designs that have been catalyzed by the Internet 2 project. Basically, new services which manage both the network and the systems which are critical to the SOA are based on familiar internet middleware: Directories/registries, authentication services, etc. These converged architectures are just now arriving in products like the [Systemet 2](#) SOA governance and lifecycle management platform.

Yet it is not always good, or possible, to build systems which work around and leave out human interactions.

**Work shedding:** In telecommunications, the applications in OSS that traditionally facilitate customer relations and satisfaction are Trouble Ticket systems and to a lesser extent Customer

Network Management applications. It was not long ago that the communications companies believed the best strategy to accomplish all the work that needed to be done for complex future product systems was to hand that work off to the customer. Really! Let customers do their own data entry for orders; they will make less mistakes. Let customers recognize the alarms for their leased circuits; they can decide if the problem is real. And finally, let customers open their own Trouble Ticket systems; they understand their problem best.

Well, customer open trouble ticket systems for telecoms and enterprise help desks are now standard. And customers do enter their own information into orders. Internet web portals accomplished this data capture reengineering marvel. Sigh! The reasons behind this are mostly incorrect. Customers cannot design their own networks or at least strategically it is not always best to ask this of them; for when they can, they do not need service providers. Customers make data entry mistakes also, and even if they have responsibility for the errors, it does not make for an accurately delivered bill. And customers are not best at recognizing significant QoS problems and communicating the nature of their problem. In the future, automation and group intelligence must be applied to this.

**Handoffs are evil:** In truth, getting orders, designs, and trouble reports correct is a partnership activity of service providers and customers. And the best partnership is a team. This is my long standing gripe with Trouble Ticket systems: the work flow model of information capture, assignment, activity and hand-off does not fit the way NOCs actually operate. Spend time watching a NOC floor and you will see *ad hoc* teams congregating around problems. They will dial up conference calls with field operatives and domain experts. Then having jointly decided the best course of action, they must go back to their desks and send the work on to each in succession to access their role-based tools and enter the trouble report information – over and over. Handoffs are evil; they waste time and dollars and interfere with the natural flow of NOC activity. People do better work, finish restorations quicker, and are more satisfied when working these problems in teams.

**Engaging the whole team at once:** QoS trouble shooting is going to be quite complex. Adding custom policy for individual customer SLA requirements, each negotiated separately in cutthroat competition will complicate this more. The future trouble correction systems and customer relationship “killer application” will be **team collaboration**. Upon receipt of an automated QoS threshold event, a network alarm on an access circuit, or a customer visiting a portal (not entering data, but pushing the connect button), the automated collaboration engine will kick into gear. Based on information automatically gleaned from the network or probe, or from directed customer questions via an AJAX (Automated JavaScript and XML) interactive portal, the rapid reaction collaboration system will use policy, availability, experience, skills, job roles, and past positive interaction ratings with the customer to automatically select and assemble the best, available **team** to solve this issue. Then these experts will get a common view of the problem, all the data gathered at once before them, a menu of potential network commands, and communications tools to interact as a team: wiki’s, blogs, instant messengers, and automatically set up IP conference calls. These systems, depending on the SLA policy rules, can network in the customer account team and/or the customer contact. Working together, such teams can solve



problems more accurately with less wasted resources than traditional network managers & trouble ticket systems. Customers get very positive memories from working with the provider to reach common goals. And of course, there are many fewer misunderstandings.

*- End -*