# Architectural Principles for Intelligent Queue Management

An ACD Design Team Publication

G J Pearson
M S McKinlay

# Status of this Memo

*This document defines a set of architectural rules for the fully automated management of contact center workloads. It is intended ultimately for publication as an Internet-Draft and, in due course, as an RFC. It is published as a technical contribution to the communications industry and may be distributed without restriction.*

# Abstract

*This paper addresses the problem of real-time resource management in modern contact centers.*

*Throughout the paper the term server is used in the queueing theory sense to describe any resource capable of processing work, whether a human agent or an automated system.*

*Traditional contact center architectures have concentrated on routing: selecting the most appropriate server to handle an individual interaction. That problem is well understood. Modern routing engines can successfully implement longest-idle routing, skills-based routing, AI-assisted pairing, and other established techniques.*

*The more difficult problem is different. It concerns the continuous allocation of finite resources across many interacting queues whose demands, service objectives, and available servers change continuously. As contact centers become increasingly multi-channel and AI-assisted, this resource-allocation problem becomes the dominant operational challenge.*

*This paper defines eight architectural rules for systems capable of automating queue management in such environments.*

*These rules are independent of any particular routing algorithm. Instead, they describe the characteristics required of systems that seek to optimise the operation of an entire contact center rather than individual routing decisions.*

# Contents

# Introduction

For more than forty years, contact center technology has evolved around a relatively stable architectural assumption: queues should be isolated sufficiently to allow them to be measured independently. This approach has served the industry well. It enables workload forecasting, workforce planning and service-level measurement using established queueing models such as Erlang C. It also lends itself naturally to stateless software architectures, making cloud-scale implementation comparatively straightforward.

These advantages explain why siloed queueing has become the dominant model across virtually all commercial ACD and CCaaS platforms.

However, the operating environment has changed.

Today's contact centers routinely manage voice, chat, email, messaging, AI agents and automated workflows simultaneously. Human agents commonly possess overlapping skills and move dynamically between multiple workloads. AI systems introduce new forms of demand while simultaneously creating secondary queues whenever customer interactions require escalation to a human specialist.

Under these conditions the optimization problem is no longer simply one of selecting the best server for an individual interaction. It becomes a continuous resource-allocation problem spanning the entire contact center.

This distinction is fundamental.

- Routing determines **who should handle the next interaction**.
- Queue management determines **how finite resources should be allocated across every interaction currently competing for service.**

Although these problems are closely related, they are not the same. Excellent routing decisions do not necessarily produce optimal queue behaviour when resources are shared across multiple workloads with different service objectives.

Traditional approaches compensate for this limitation through supervisor intervention or static business rules. Neither scales well. Human supervisors cannot continuously evaluate the consequences of every resource decision across dozens or hundreds of interacting queues, while fixed rules cannot respond intelligently to changing operational conditions.

The problem is analogous to predictive outbound pacing. Decades of experience have demonstrated that human intervention and simple rules cannot solve predictive pacing effectively because the optimization problem is computationally intractable at practical scale. The same observation applies equally to large-scale inbound resource management.

The purpose of this paper is therefore not to redefine routing.

Rather, it is to define the architectural principles required for systems capable of automating queue management itself.

The eight rules presented in this document are intended as invariant design principles. They describe the behaviour required of any system seeking to optimise resource allocation continuously across modern contact center workloads.

# Rules for an Intelligent Queueing System

It should be noted that these rules have come about through understanding of the resource management problem for complex inbound loads, mirroring outbound and session blending.

So without further ado, we offer the following as rules for building an intelligent queueing system that deals with modern contact center loads.

## 1. The Rule of Non-Routine Intervention

> *An effective solution for real-time queue management **MUST** automate its decisions in such a way as to consider both change over time and change risk to avoid the need for manual intervention.*

*This rule concerns who is authorised to make allocation decisions. It stems from the simple fact that intelligent machinery can respond to changes in demand both more quickly and in the right measure compared with a human (or dare we say it, AI-driven) supervisor.*

Every decision to move a resource affects the future state of the contact center. Reallocating a single server influences queue lengths, service levels, waiting times, resource availability and, frequently, the performance of several other queues simultaneously. The effect is dynamic rather than instantaneous.

Human supervisors are very good at recognising patterns, managing exceptions and exercising judgement. They are not well suited to solving hundreds of interconnected optimization problems every minute across an entire contact center. Nor can static rule sets anticipate every combination of workload, staffing and demand that may arise during live operation.

Consequently, any architecture that depends upon continual supervisory intervention has already accepted that the automation problem remains unsolved.

Instead of continuously controlling resources, supervisors should supervise the quality of service delivered by the automation itself. They should intervene only when business policy changes or exceptional operational circumstances require human judgement.

An effective queue management system is therefore characterised not by the sophistication of its supervisor controls but by the absence of any routine need to use them.

# 2. The Rule of Estate-Wide Balancing

*Any contact center queueing system with multiple queues **SHALL** have:*

- *servers who may serve load on multiple different queues based on either absolute or relative skills*
- *queues that handle workloads with differing service expectations and timelines*
- *servers which behave differently or have different escalation profiles depending on proficiency*

*As such any system for managing such queues **MUST** seek to balance load considering not just decisions in the instant, but over time.*

*This rule concerns where and at what scope, allocation decisions must be made. Every routing decision changes the availability of resources for every other queue sharing those resources. Optimising one queue independently may therefore degrade overall contact center performance.*

A modern queue management system ***MUST*** continuously balance service across the entire queueing estate rather than attempting to optimise individual queues independently. Fairness is therefore not equality of treatment; it is the continuous allocation of finite resources so that overall service objectives are achieved as effectively as possible.

An important consequence of this rule is that queue management cannot be decentralised.

If individual queues independently determine when to allocate resources, no mechanism exists to optimise the behaviour of the contact center as a whole. Competing queues will inevitably make locally optimal decisions that are globally sub-optimal.

Instead, the queue management system itself must determine which queue should receive attention at any point in time. Routing decisions become subordinate to this higher-level optimization process.

Viewed in this way, routing and queue management perform complementary but distinct roles.

Queue management determines which queue should receive the next available resource.

Routing determines which eligible server should receive the next piece of work within that queue.

Maintaining this separation of responsibility is one of the fundamental architectural principles underlying the remainder of this paper.

# 3. The Rule of Continuous Evaluation

> *Any contact center queueing system that automates the balancing of workloads **MUST** re-evaluate order of queue processing for each routing/pairing decision.*

*This rule concerns when allocation decisions must be recomputed. No routing decision is independent. Each allocation changes the availability of resources throughout the contact center. Consequently, every completed assignment alters the optimal ordering of future work. Queue management therefore becomes a continuous optimization process rather than a sequence of isolated routing decisions.*

This distinction has important architectural implications.

Traditional ACDs process queues independently, allowing each queue to make routing decisions whenever work becomes available. Such an approach is computationally efficient but assumes that queues are independent. In modern contact centers this assumption rarely holds. Shared resources create continuous dependencies between queues, meaning that every allocation influences the behaviour of the wider system.

The consequence is that queue processing order itself becomes dynamic.

Rather than each queue progressing independently, an overarching optimization process continuously determines which queue should receive attention next, taking account of current demand, available resources, service objectives and predicted system behaviour.

Critics may argue that continuously re-evaluating queue priorities introduces unacceptable computational overhead. This concern is understandable but not new.

Historically, PBX systems encountered a similar scaling problem through what became known as the busy lamp problem: doubling the size of an organisation did not merely double the information a receptionist had to process; it increased the number of relationships between extensions dramatically. Modern contact centers face an analogous challenge.

The existence of a difficult scaling problem, however, is not evidence that the optimization itself is unnecessary. It simply means that the underlying algorithms must be designed to scale appropriately.

What has changed is not the difficulty of the problem but its tractability. Continuous, estate-wide re-evaluation at contact center volumes and latencies was not practically achievable when siloed queueing architectures were first designed. Advances in real-time compute and AI-based optimization have made it achievable only recently, which is precisely why this rule is stated as an architectural requirement now rather than being assumed all along.

Continuous evaluation is therefore not an implementation option. It is a prerequisite for intelligent queue management.

# 4. The Rule of Determinism

> *In a contact center system, queues **MUST** contain like workloads that enable measurement through counting and reasonable averaging of handling time for each work item.*
>
> *Queues **MUST** also have clear visibility of the servers who **MAY** handle work items **and the likelihood of availability of servers from the pool of eligible servers.***

*To make meaningful decisions about load on any given queue, a queue should contain 'like' workloads that can be measured. Similarly, a queue must know what servers are available to it. This is a fundamental rule of traditional queueing theory but needs to be restated in such a way as to facilitate determinism in a dynamic environment.*

Without the words in ***bold*** in the rule this could be read as an axiom of queueing systems generally. The data on likely availability is necessary for a queue to determine server resources actually required to keep the queue within service level constraints.

Also, this rule appears to negate the idea of a 'universal queue' where a queue may contain work items of differing cost. The universal queue has no place in the design of a system that uses measurement to make load-balancing calculations. This isn't to say that the idea of a universal queue is a bad thing but rather that it needs to be implemented as an aggregate of queues of 'like' workloads.

The words in ***bold*** also have implications for automated blend between outbound and inbound campaigns. A well-designed blending solution will seek to keep 'just enough' agents working inbound loads and move servers with outreach skills to outbound projects. If outbound agents are doing predictive dial work, they cannot be withdrawn from the outbound pool arbitrarily without causing outbound abandons, with the attendant cost of compliance breach and poor CX. A system implementing automated blend ***MUST*** provide the inbound queue management system with effective metrics for likely time to release either any agent or a specific agent to support deterministic behaviour in inbound queueing, and the metrics ***MUST*** be cumulative. It may only take a handful of seconds to move a single agent from outbound to inbound, but in the case of one or multiple inbound spikes in demand, this 'time to grab agent' increases significantly.

The ideal solution for both blending and automated inbound load management is integrated logic that manages both activities homogeneously.

# 5. The Rule of Resource Capability Modelling

> *A contact center system **MUST** faithfully model the maximum number of concurrent tasks each server can handle, by task type, and **MUST** provide interfaces to change those limits in real-time.*
>
> *Such a system **MUST** also model the impact of a server's endpoint being in use on the real-time availability of other endpoints managed by the same server.*

*An ACD turret is no longer a simple agent with an audio endpoint and possibly some call parking or informal queueing capability. The modern equivalent (for which there is no industry-standard name) may have multiple streamed media endpoints, of which only one may be active, and may facilitate parallel operation of non-streamed media (e.g. chat) transactions. Turrets are no longer fixed. They may be dynamic; the maximum number of concurrent tasks may be changed in real-time for human resources, and bot resources may be created on demand within limits.*

Modern contact centers require considerably richer representations. Every allocation alters not only the immediate availability of a resource but also its future capacity to undertake other forms of work.

The optimization engine must therefore model resource capability rather than simple resource availability.

As AI becomes increasingly integrated into contact center operation, this distinction becomes even more important. Unlike human resources, AI capacity may often be expanded dynamically. However, such expansion carries measurable economic cost. Intelligent queue management therefore seeks to optimise not merely service performance but the overall utilisation of both human and automated resources.

The purpose of the resource model is not to describe the contact center. Its purpose is to provide the optimization engine with sufficient information to make consistently better decisions than would otherwise be possible.

# 6. The Rule of Channel-Agnostic Prioritization

> *A contact center queueing mechanism **MUST** be able to make equivalent measurement of queues that operate on widely differing timescales to enable load-balance to SLA and **MUST** be able to measure the projected impact of adding or removing a server from the pool of resources specific to a given queue.*

*In any contact center queueing system, queues will be operating under load and, sometimes, more load than there are servers available to handle. In such circumstances it is necessary to degrade handling as evenly as possible across the whole queueing estate rather than an individual queue. As an individual queue with finite server resource comes under increasing load, response times exponentiate. It is essential to degrade gracefully and evenly to avoid cliff-edge scenarios where customer wait times for voice service go from seconds to tens of minutes.*

Different channels operate on different timescales and volumes;

- a queue handling a voice channel may have a *'80% handled within 30 seconds'* SLA rule, and in a busy hour may handle 300 work items, with an AHT of 8 minutes.
- A queue handling email response may have a *'99% handled within 1 hour'* SLA, and in a busy hour may handle 10,000 work items with an AHT of 14 seconds

Even though these queues operate on differing timelines the mechanism for allocating server resource must be able to make an equivalence and must understand the relative impact of a single server on each queue. Allocating an additional server to our notional voice queue above means an additional 7 or 8 work items will be able to be processed in a busy hour, contrasted with around 260 work items by adding a server to our notional email queue.

# 7. The Rule of In-Situ Injection

*A contact center queueing system **MUST** present only valid choices to routing machinery so that the routing machinery **MAY** render a decision on pairing a server with a work item. The routing machinery **MAY NOT** make decisions on server selection for servers that may be temporarily beyond its purview.*

*When routing machinery takes a finite and nonzero amount of time to make a pairing determination, the queuing request **SHALL** take 1 of the following 2 forms:*

- *One server, many work items*
- *Many servers, one work item*

*In addition:*

- *the queueing system **SHALL** determine which form of request is appropriate*
- *the routing machinery **SHALL** determine the handling preference order of servers or work items*

*A key principle for load-balanced operation is that it will skew, in real-time, the resources available to make a routing/pairing decision. Since AI or BI-based pairing is not a real-time activity, but resolves in the time range of a few milliseconds to a handful of seconds, it is necessary for the dialog to indicate a preference order.*

In the time that the routing engine takes to render a decision, available resources may have changed and so the queueing system needs to render the routing decision as best it can, with the resources available, at the point in time that the routing engine renders a decision.

Consider the following examples:

- Queuing Request: Server A may handle work items [1, 2, 3, 4, 5].
  Routing Response: Server A should handle work items in the preference order [5,2,1,4,3]
- Queuing Request: Servers [A, B, C, D, E] may handle work item 273.
  Routing Response: Work item 273 should be handled in preference order by server [C, A, B, E, D]

The queueing system ***MUST***, in all cases, deal with the scenario where the routing machinery does not render a decision within an acceptable timeframe, and fall back to a real-time route decision which may be, for example longest-idle or skill-based.

# 8. The Rule of WFM-Agnostic Coexistence

> *A contact center queueing/ routing system **MUST** make real-time decisions autonomously and **CANNOT** be constrained by external systems that make near-real-time decisions on resource availability.*

*Workforce scheduling systems have value for scheduling the amount of resource that may be needed on a given day and at a given hour. Some workforce management solutions also have facilities that enable control of resource allocation during the operating day. Such systems do not have real-time purview and make decisions that are sub-optimum in real-time. Therefore they must be either operated in an advisory/recommend-only mode with respect to the ACD, or disabled.*

Quite apart from this fact, two independent decision-makers issuing conflicting instructions to the same resource creates oscillation, thrash, or race conditions and should be avoided at all costs.

This is not an argument against integration. Workforce data such as schedule adherence and predicted availability is valuable input to real-time queue management, and the relationship should run both ways: the queueing system is equally well placed to feed WFM a running commentary on emerging skills shortages to improve future scheduling.

The distinction is between information and authority. Each system may inform the other. Neither may make real-time decisions on the other's behalf.

In short, use WFM for workforce scheduling and as a source of context, not as a source of real-time control.

# Summary

For decades the contact center industry has concentrated on optimising routing decisions, with the primary aim of keeping SLA under control. It hasn't delivered and as this paper sets out, the right approach is the continuous optimization of finite resources across the entire contact center. Routing remains essential, but when conditions become complex, live, and operationally demanding, it is only one component of a broader optimization architecture.

The eight principles described here provide a framework for designing and evaluating systems capable of meeting that challenge in the AI era.

In practice, when these principles are applied, the benefits are substantial. They include a major reduction in the need for supervisor intervention and a significant improvement in customer experience through both pro-active resource management and supervisors being able to focus on qualitative improvements instead of crisis management.

As readers of this paper may know, Sytel has spent many years developing Sytel Real-Time Optimization (SRO) to meet the challenges described in this paper and in working in accord with the principles we have set out.

**Contact Sytel**

**Web**: sytel.com
**Email**: info@sytel.com
**Phone**: +44 (0)1296 381 200