Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Agentic Cloud Infrastructure Management: The New Age of Development and Operations

Mohamed Rizwan Syed Sulaiman
Independent Researcher, USA.

ARTICLE INFO

Received:05 Sept 2025

Revised:09 Oct 2025

Accepted:19 Oct 2025

ABSTRACT

Cloud infrastructure management is at a revolutionary crossroads where autonomous software agents are essentially redefining operational paradigms. Conventional infrastructure automation is based on procedural scripts necessitating explicit specification of all sequences of actions, involving heavy maintenance overhead and restraining flexibility to unexpected conditions. Agentic management systems provide goal-directed execution frameworks in which human engineers specify high-level goals while autonomous agents autonomously decide optimal implementation schemes. These smart systems utilize continuous learning features to monitor telemetry streams, identify patterns in performance, and create advanced behavior models for proactive intervention prior to insignificant deviations becoming service interruptions. Moving from reactive threshold-based monitoring to proactive self-healing mechanisms greatly decreases the number of incidents while improving recovery processes. Quality-of-service-based component choice, constraint-based resource placement, and adaptive autoscaling are key technological underpinnings for autonomous optimization across various operational axes. Human engineering jobs change proportionally, moving from tactical deployment towards strategic architecture definition, policy making, and governance framework design. Engineers have to develop skills in declarative specification languages, multi-objective optimization formulations, and mechanisms for validating agent behavior. Service-level agreement languages and service-oriented programming models offer crucial abstractions for describing operational intent to autonomous systems. The intersection of machine learning, multi-agent coordination protocols, and cloud-native architectures creates the technical foundation for infrastructure environments that are always evolving, self-optimizing, and ensuring resilience through smart autonomous decision-making in lieu of manual action.

Keywords: Agentic Management, Autonomous Infrastructure, Goal-Oriented Execution, Self-Healing Systems, Adaptive Optimization, Cloud Orchestration

Introduction

Cloud infrastructure management follows an ongoing path to increasing abstraction and automation. From manual server configuration in the early days to the scripted automation era, the market now stands at the cusp of a revolutionary paradigm: agentic management. This new direction significantly rethinks the way that cloud infrastructure is managed, serviced, and optimized. Instead of depending on human engineers to foresee all situations and program responses into procedural codes, agentic systems utilize autonomous software agents with the system's ability to make independent decisions, learn over time, and adapt. The difficulty in administering current cloud infrastructure is rooted in the intrinsic complexity of distributed systems, where optimizations at the protocol level and design choices have a direct effect on operational efficacy. Protocol optimization research illustrates how performance at runtime can be achieved through precise examination of method dispatch mechanisms and memory allocation patterns, as shown to introduce quantifiable overhead through existential containers and witness tables in protocol-oriented environments [1]. These results emphasize the value of knowing low-level operational behaviors when creating autonomous systems for making real-time optimization choices throughout distributed infrastructure.

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The move towards agentic management is also brought about by the added architectural complexity resulting from microservices adoption, which has drastically altered how cloud applications are built and run. Industry reports indicate that organizations adopting microservices architectures will deploy between eight and fifteen different services per application, with some big systems having more than fifty independently deployable units [2]. This distributed nature of architecture generates significant operational overhead since every microservice needs separate monitoring, deployment pipelines, and resource management. Research into microservices practice reveals that seventy-three percent of practitioners surveyed mention more complexity in debugging and troubleshooting distributed systems, and sixty-eight percent mention monitoring and observability as key operation challenges [2]. Traditional manual methods and scripted automation find it difficult to handle this kind of complexity efficiently since human operators have to integrate data across many service boundaries, dependency chains, and failure modes that interact in a non-linear fashion.

These agents embody the transition from human-led automation to machine-led autonomy, in which smart systems take on the operational complexity of contemporary cloud infrastructure while human engineers are relegated to strategic guidance and architectural design. The shift to agentic management fills the gap that microservices environments create in producing enormous amounts of telemetry data in distributed tracing systems, metrics aggregation platforms, and log consolidation frameworks, which call for advanced correlation and analysis capabilities beyond human intellect [2]. In addition, the dynamic nature of containerized workloads, where service instances horizontally scale in reaction to demand and communication patterns continuously change, requires real-time decision-making that occurs at machine speeds instead of human reaction times. Autonomous agents that possess machine learning can deal with this multi-dimensional operational data, discern faint patterns of performance degradation before cascading into system failures, and implement remedial strategies that dynamically adjust to the context of each occurrence, thus realizing degrees of operational resilience and efficiency that are not possible with traditional automation techniques.

From Scripted Automation to Goal-Oriented Execution

Procedural thinking is the foundation of traditional infrastructure management practices. Engineers write meticulous scripts that outline unique sequences of steps for provisioning infrastructure, configuring offerings, and reacting to nicely-defined conditions. Even though this infrastructure-as-code approach has immensely superior consistency and reproducibility, it still fundamentally stays reactive and brittle. All edge cases have to be foreseen, all failure modes explicitly addressed, and all operational workflows hand-coded. The nature of contemporary cloud provisioning requires advanced modeling techniques that are capable of abstracting infrastructure elements and their dependencies. Studies in infrastructure modeling illustrate the fact that cloud infrastructures need to be represented in terms of multiple architectural layers, such as virtualization infrastructure, networking topologies, storage environments, and application deployment models [3]. The problem becomes especially pressing when dealing with multi-cloud heterogeneous environments in which infrastructure descriptors need to consider different types of resources that have varying provisioning parameters and configuration needs. Research on cloud provisioning tools discovers that human error-prone infrastructure specification processes occur, with misconfiguration levels averaging twenty-three percent in environments where automated validation processes are not in place, and provisioning entire application stacks manually takes from forty-five minutes to multiple hours based on complexity [3]. The cost of upkeep for these increasingly sophisticated automation frameworks takes enormous amounts of engineering time, since infrastructure models need to be iteratively updated to support new service offerings, revised API standards, and shifting organizational needs.

Agentic management actually reverses this paradigm by accepting goal-driven execution. Rather than outlining exact steps in implementation, engineers define top-level objectives that specify outcomes in terms of what, instead of the exact mechanism by which. The autonomous agent translates such objectives, regularly reviews the corresponding systems, and autonomously decides on suitable

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

actions to attain and sustain the defined goals. Multi-agent systems enable the architectural basis for such transformation, supporting distributed intelligence where professional agents work together to realize complex operational objectives beyond the capabilities of monolithic automation systems [4]. Studies of multi-agent architectures show that agent-based systems are superior in dynamic situations with uncertainty, incomplete knowledge, and real-time adaptation requirements—exactly the circumstances that prevail in contemporary cloud computing. These systems utilize highly advanced coordination mechanisms such as contract net protocols, in which agents bargain task allocation on the basis of capability and availability, and blackboard architecture in which observational data and intermediate results are shared by agents to construct collective situational awareness [4]. The effectiveness of multi-agent approaches stems from their ability to decompose complex problems into manageable subproblems handled by specialized agents, with empirical studies showing that agent-based systems can reduce problem-solving time by forty to sixty percent compared to centralized approaches when addressing distributed optimization challenges [4]. This declarative approach liberates engineers from the cognitive overhead of anticipating every operational scenario, allowing them to focus on defining what should be accomplished rather than micromanaging how it should be done. The agent takes on the role of mapping intent into action, dynamically modifying its strategies as conditions change and new information is introduced, based on reasoning abilities that permit agents to choose actions as a function of present environmental state, anticipated outcomes, and specified utility functions as opposed to simply following a pre-defined sequence of instructions [4].

Management Approach	Configuration Complexity	Adaptation Success Rate	Policy Specification Requirements	Problem Resolution Time
Procedural Scripting	Extensive code requirements	Low for unexpected conditions	Detailed module specifications	Extended resolution periods
Infrastructure as Code	High maintenance overhead	Limited flexibility	Explicit edge case handling	Significant debugging time
Goal-Oriented Agents	Substantial complexity reduction	High for unexpected conditions	Minimal policy definitions	Real-time adaptive response
Multi-Agent Systems	Decomposed problem spaces	Faster problem-solving	High-level intent declarations	Dynamic strategy adaptation

Table 1. Performance Comparison of Infrastructure Management Paradigms [3, 4].

Continuous Learning and Adaptive Optimization

The full potential of agentic systems is realized through their ability to engage in continuous learning and self-enhancement. In contrast to fixed automation scripts that apply fixed rules, smart agents examine patterns in huge amounts of telemetry data, performance statistics, and operational occurrences. As they continually conduct this examination, they improve their models of system behavior to such an advanced degree that they can identify subtle anomalies, forecast impending problems, and maximize resource consumption with sophistication that goes beyond the simple logic of rules [5]. Quality-of-service driven component ranking research in the cloud environment proves that intelligent systems need to analyze service components along various axes, such as response time, throughput, reliability, and availability, in order to build end-to-end performance models. Web service composition research identifies that cloud applications incorporate between fifteen to forty-five different service components, each having varying performance attributes that change depending on

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

network conditions, resource contention, and workload patterns [5]. Optimal component configuration identification poses computational challenges as the solution space increases exponentially with the increase in the number of service alternatives. Empirical studies of component ranking models demonstrate that quality-aware selection mechanisms increase end-to-end application response times by thirty-two to fifty-one percent versus random selection of components, while, at the same time, improving reliability measures by selecting components with historical failure rates under three percent [5]. These learning systems utilize collaborative filtering methods and matrix factorization strategies to forecast service quality dimensions for component pairs that have not been directly experienced, with prediction accuracy levels of over eighty-five percent when trained on adequate amounts of historical invocation data [5].

This adaptive nature is realized in several aspects. Agents learn to associate seemingly unrelated signals to detect underlying causes of performance degradation, understanding that infrastructure optimization needs advanced placement strategies that consider heterogeneity of resources and workload patterns [6]. Studies of virtual machine placement algorithms show that intelligent decisions at placement have a significant impact on overall cloud infrastructure efficiency, with experiments comparing different strategies such as First Fit, Best Fit, Worst Fit, and optimization-based strategies under varied workload profiles. Experimental analysis in realistic cloud simulation environments shows that simple placement schemes like First Fit can produce resource fragmentation levels of more than forty-five percent, which translates into huge capacity wastage and higher operation costs [6]. They observe seasonal workload behavior patterns and act in anticipation to modify capacity provisioning ahead of time, with sophisticated placement algorithms lowering the number of active physical hosts needed by eighteen to thirty-two percent over baseline methods, thus saving the amount of energy and infrastructure costs commensurately [6]. They compare the efficiency of their own interventions and adjust their decision-making strategies accordingly, as confirmed by comparative studies illustrating that optimization-based placement methods cut service level agreement breaches by fifty-three to sixty-seven percent while increasing resource utilization rates from baseline rates of forty-eight percent to optimized rates of seventy-two to seventy-nine percent [6]. This has a feedback effect whereby operational intelligence builds up over time, evolving the infrastructure from being a passive pool of resources to an actively self-optimizing system that improves with experience, with simulation results showing that adaptive placement approaches can handle thousands of virtual machine allocation requests with placement decision latencies below two hundred milliseconds, facilitating real-time optimization even in extremely dynamic cloud environments [6].

Optimization Dimension	Performance Metric	Improvement Range	Implementation Approach
Component Selection	Response time improvement	Significant enhancement	QoS-driven ranking frameworks
Service Reliability	Component failure rate	Below threshold levels	Collaborative filtering techniques
Prediction Accuracy	Quality attribute forecasting	High accuracy levels	Matrix factorization methods
Resource Placement	Fragmentation reduction	Fewer active hosts required	Optimization-based algorithms
Utilization Efficiency	Resource allocation rates	Optimized allocation levels	Adaptive placement strategies
SLA Violation Reduction	Service compliance	Substantial violation reduction	Intelligent placement techniques
Decision Latency	Placement processing time	Sub-second processing	Real-time optimization engines

Table 2. Continuous Learning and Optimization Metrics in Cloud Systems [5, 6].

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Proactive Self-Healing and Resilience

Arguably, the strongest strength of agentic infrastructure management is its ability for proactive self-healing. Conventional monitoring and alerting systems follow threshold-based reasoning, only alerting human administrators when certain conditions are exceeded. Reactive systems have inherent delays between trouble beginning and repair, during which time service quality can deteriorate and user experience is impaired [7]. Virtual machine provisioning with placement constraints research indicates that cloud infrastructure needs to support sophisticated application needs, such as affinity rules that require co-location of communicating services, anti-affinity constraints that require segregation for fault tolerance, and capacity constraints that provide sufficient resource availability between failure domains. Experiments on placement optimization problems show that infrastructure as a service environments normally handle provisioning requests on collections of three to twenty-five virtual machines with interdependent placement constraints, in which constraint satisfaction becomes computationally hard when growing larger [7]. Analysis of heuristic placement algorithms indicates that greedy strategies may accept between sixty-two percent and seventy-eight percent of requests under moderate load, but are severely impacted in acceptance levels to thirty-five to forty-nine percent once resource fragmentation is built up due to poor previous placement choices [7]. The computational complexity of best placement increases exponentially with problem size, with exact optimization methods requiring solution times of more than several minutes to solve for requests with more than fifteen virtual machines, which renders them ineffective for real-time provisioning scenarios where response latency must be less than ten to fifteen seconds to continue to provide an acceptable user experience [7].

Autonomous agents overcome this limitation by constantly monitoring system health and acting before small problems snowball into large-scale mishaps. By catching early warning signs like gradual resource depletion, forming bottlenecks, or marginal behavioral anomalies, agents can take control measures when issues are still contained and within control [8]. Experiments on self-adaptive autoscaling algorithms for cloud services show that self-determined decision-making should trade off several conflicting goals, such as maintenance of service quality, minimizing operational cost, and configuration stability to prevent thrashing between scaling states. Research probing trade-off decision models indicates that cloud applications have very volatile scaling demands, with patterns of workloads showing coefficient of variation values between zero point four and two point six based on application nature and patterns of users' behavior [8]. The autoscaling challenge becomes even more intricate when taking into account that horizontal scaling via instance replication presents varying cost-performance behavior compared to vertical scaling via resource supplementation, with test results indicating that horizontal methods usually provide superior fault tolerance and load distribution but are more expensive in terms of licensing and network overhead [8]. Adaptive scaling policy analysis using reinforcement learning methods illustrates that smart agents can decrease service level agreement breaches by forty-seven to sixty-three percent relative to reactive threshold-based autoscaling, while at the same time lowering infrastructure expenditures by twenty-three to thirty-nine percent through better resource utilization [8]. This could mean undoing recent alterations, reassigning workloads, or reconfiguring system parameters dynamically, with simulation experiments being shown to suggest that self-adaptive systems can converge toward optimal scaling configurations in eight to fifteen adaptation cycles, which corresponds to twenty to forty-five minutes of operation observation based on workload dynamics and latency of feedback [8]. The outcome is infrastructure that is resilient not by over-provisioning and redundancy but through smart, adaptive measures to changing conditions, with field tests demonstrating that multi-objective optimization methods balancing quality, cost, and stability can realize Pareto efficiency gains of thirty-two to fifty-one percent over single-objective optimization methods that ignore key trade-off dimensions [8].

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Self-Healing Capability	Constraint Type	Performance Impact	Operational Benefit
VM Set Provisioning	Affinity and anti-affinity rules	Moderate to high acceptance	Complex placement satisfaction
Heuristic Placement	Resource fragmentation management	Variable under load conditions	Dynamic constraint handling
Constraint Optimization	Multiple VM configurations	Fast response latency	Real-time provisioning
SLA Violation Prevention	Multi-objective balancing	Substantial reduction	Proactive quality maintenance
Cost Optimization	Horizontal and vertical scaling	Significant cost decrease	Efficient resource utilization
Adaptive Convergence	Learning-based optimization	Multiple adaptation cycles	Near-optimal configuration
Pareto Efficiency	Quality-cost-stability trade-offs	Multi-dimensional improvement	Balanced optimization outcomes

Table 3. Proactive Self-Healing Capabilities and Resilience Outcomes [7, 8].

The Evolving Role of Human Engineers

The rise of agentic management does not devalue the role of human expertise but instead pushes engineering work up the stack to concentrate on higher-order issues. Rather than spending time crafting maintenance scripts or answering recurring operational alerts, engineers focus on architectural design, policy development, and strategic planning. They determine the goals that inform autonomous agents, set guardrails that limit agent action into tolerable bounds, and create feedback loops that allow autonomous improvement [9]. Service level agreement language studies for cloud-based collaborative systems prove that human engineers have to use formal specifications to implement intricate quality-of-service conditions, operational requirements, and performance guarantees that determine the behavior of autonomous systems. Research into collaborative service provisioning finds that contemporary cloud applications usually consist of three to seven organizational entities that run under separate service level agreements, each having specific response time thresholds, availability commitments, and throughput guarantees [9]. The challenge of handling these multi-party contracts becomes acutely pronounced when one takes into account the fact that service level agreement breaches in collaborative settings tend to cascade across organizational silos, with research indicating that one provider's inability to honor commitments will set off consequential violations impacting downstream consumers in forty-three to sixty-one percent of intricate service composition contexts [9]. Examination of agreement specification languages shows that engineers need to specify not just performance metrics and threshold values but also complex monitoring mechanisms, violation detection algorithms, penalty computation formulas, and remediation processes, with typical enterprise contracts having between twenty-five and seventy-eight separate clauses controlling different aspects of service delivery and quality assurance [9]. The mental complexity of doing this manually at scale becomes untenable, calling for autonomous agents that can continuously monitor agreement, verify compliance, and adaptively respond to maintain contractual terms within dynamic operating environments [9].

Engineers need to acquire new skills to make this transition. Knowing how to properly convey intent to intelligent systems becomes a priority. Creating observable architectures that enable agents to receive information required for well-informed decision-making becomes an essential skill [10]. Service-oriented computing programming model research identifies that engineers need to move away from imperative programming models that are concerned with algorithmic control flow toward declarative styles that focus on composition, coordination, and quality-of-service specification.

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Research on service-oriented development practices illustrates that average enterprise applications compose between twelve and thirty-five external services, each with functional interfaces exposed using standard protocols and with internal implementation opacity [10]. The crux problem is to assemble these independently composed services into well-formed applications while addressing non-functional concerns like transactional consistency, security policy enforcement, and error recovery across organizational boundaries. Service composition patterns analysis reveals that engineers need to become experts in orchestration languages for describing intricate workflows with concurrent execution branches, conditionally based on runtime information, and compensation logic for managing partial failures in long-running business processes [10]. Engineers will also need to create frameworks for agent behavior monitoring, ensuring autonomous decisions are consistent with organizational goals and keeping adequate control without excessive micromanagement, while studies have shown that service-oriented architectures need to be equipped with advanced management infrastructure in the form of service registries for capability discovery, policy frameworks for access control enforcement, and monitoring systems for quality-of-service verification [10]. The role of human beings changes from implementer to architect, from firefighter to strategist, with programming model evolution research indicating that abstraction layers that support declarative service composition can cut application development effort by thirty-five to fifty-two percent against low-level integration coding, while at the same time enhancing system maintainability by providing clean separation between business logic and infrastructure issues [10].

Engineering Dimension	Traditional Allocation	Agentic Allocation	Competency Requirement
Operational Tasks	The majority of the effort	Minority of effort	Reactive incident response
Strategic Initiatives	Minority of effort	The majority of the effort	Architecture and planning
SLA Specification	Multiple distinct clauses	Automated compliance monitoring	Formal agreement languages
Violation Cascade Risk	Significant in multi-party scenarios	Agent-mediated prevention	Cross-boundary coordination
Service Integration	Numerous external services	Declarative composition	Orchestration languages
Development Effort Reduction	Baseline implementation	Substantial reduction	Service-oriented abstractions
Training Period	Not applicable	Several weeks of proficiency	Goal-oriented specification
Productivity Improvement	Baseline capacity	Significant increase	Agent collaboration mastery

Table 4. Human Engineering Role Transformation in Agentic Systems [9, 10].

Conclusion

Agentic cloud infrastructure management is a core rethink of how digital systems run and regulate themselves across dispersed conditions. The constraints built into procedural automation—brittle failure recovery, exponential increase in complexity, and ongoing manual maintenance needs—call for a paradigmatic shift toward self-governance. Intelligent agents with learning capabilities, predictive analytics, and adaptive decision-making break reactive operational paradigms by acting proactively, continuously optimizing, and self-improving based on accumulated operational intelligence. Component ranking based on quality allows for intelligent selection of services from large catalogs of options, while placement algorithms based on constraint awareness minimize resource usage based on intricate affinity and capacity constraints. Autoscaling algorithms with self-adaptive features

balance competing goals such as performance, cost minimization, and configuration stability through advanced trade-off analysis. Technical complexity in multi-agent coordination, reinforcement learning, and causal inference allows systems to manage operational complexity beyond human cognitive capabilities in real-time processing environments. Practice in engineering changes equivalently as practitioners shift from applying procedural logic to designing declarative policies, authoring observable systems, and developing governance frameworks that ensure autonomous decisions align with organizational goals. Service-level agreement languages facilitate formal means of expressing quality commitments across organizational boundaries, and service-oriented programming models present compositional abstractions that control distributed application complexity. The symbiosis between human strategic oversight and machine operational autonomy unlocks unprecedented stages of efficiency, reliability, and flexibility. Infrastructure environments are becoming energetic contributors in preserving superior operation in place of passive aid pools waiting for human coaching, marking the emergence of truly intelligent cloud platforms.

References

[1] RAJKISHORE BARIK et al., "Optimization of Swift Protocols," ACM, 2019. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3360590
[2] Markos Viggiato et al., "Microservices in Practice: A Survey Study," arXiv, 2018. [Online]. Available: https://arxiv.org/pdf/1808.04836
[3] Julio Sandobalin et al., "An Infrastructure Modelling Tool for Cloud Provisioning," ResearchGate. [Online]. Available: https://www.researchgate.net/profile/Julio-Sandobalin/publication/316701858_An_Infrastructure_Modelling_Tool_for_Cloud_Provisioning/links/5a1b1b11aca272df080f1629/An-Infrastructure-Modelling-Tool-for-Cloud-Provisioning.pdf
[4] J. Tweedale et al., "Innovations in multi-agent systems," Elsevier, 2006. [Online]. Available: https://www.researchgate.net/profile/Gloria-Phillips-Wren/publication/222939284_Innovations_in_multi-agent_systems/links/5b9d91c1a6fdccd3cb5a75d0/Innovations-in-multi-agent-systems.pdf
[5] Zibin Zheng et al., "CloudRank: A QoS-Driven Component Ranking Framework for Cloud Computing," ResearchGate. [Online]. Available: https://www.researchgate.net/profile/Michael-Lyu/publication/224189968_CloudRank_A_QoS-Driven_Component_Ranking_Framework_for_Cloud_Computing/links/543c63f90cf2c432f74201d5/CloudRank-A-QoS-Driven-Component-Ranking-Framework-for-Cloud-Computing.pdf
[6] Mohammed Rashid Chowdhury et al., "Implementation and performance analysis of various VM placement strategies in CloudSim," Springer, 2015. [Online]. Available: https://link.springer.com/content/pdf/10.1186/s13677-015-0045-5.pdf
[7] Lei Shi et al., "Provisioning of Requests for Virtual Machine Sets with Placement Constraints in IaaS Clouds," ResearchGate. [Online]. Available: https://www.researchgate.net/profile/Brendan-Jennings/publication/261116675_Provisioning_of_requests_for_virtual_machine_sets_with_placement_constraints_in_IaaS_clouds/links/553ff72f0cf29680de9dc204/Provisioning-of-requests-for-virtual-machine-sets-with-placement-constraints-in-IaaS-clouds.pdf
[8] Tao Chen et al., "Self-Adaptive Trade-off Decision Making for Autoscaling Cloud-Based Services," IEEE TRANSACTIONS ON SERVICES COMPUTING, 2015. [Online]. Available: https://arxiv.org/pdf/1608.05917
[9] Surya Nepal et al., "WSLA+: Web Service Level Agreement Language for Collaborations," IEEE International Conference on Services Computing, 2008. [Online]. Available: https://www.researchgate.net/profile/Shiping-Chen-2/publication/4359252_WSLA_plus_Web_Service_Level_Agreement_Language_for_Collaborations/links/0f317531c4ba21bc33000000/WSLA-plus-Web-Service-Level-Agreement-Language-for-Collaborations.pdf

Journal of Information Systems Engineering and Management
2025, 10(61s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[10] Francisco Curbera et al., "Toward a Programming Model for Service-Oriented Computing," Springer, 2005. [Online]. Available: https://www.researchgate.net/profile/Francisco-Curbera/publication/221050852_Toward_a_Programming_Model_for_Service-Oriented_Computing/links/0deec5239ac641aab5000000/Toward-a-Programming-Model-for-Service-Oriented-Computing.pdf