ISSN: 3067-4166

AMERICAN DATA SCIENCE JOURNAL FOR ADVANCED COMPUTATIONS

VOLUME: 02 ISSUE: 01

RECEIVED: JANUARY 08 REVISED: JANUARY 28 ACCEPTED: FEBRUARY 18 PUBLISHED: MARCH 03

Intelligent Automation: Multi-Agent AI for Enterprise IT Service Management

Independent Researcher

1st Anumandla Mukesh

mukeshanumand@gmail.com

Abstract

Development of Multi-Agent Systems capable of partially or fully automating IT incident response, service request fulfilment or routing across the agent lifecycle, for integration into existing enterprise toolchains and alignment with ITIL across both DevOps and IT Service Management seams. Active experimentation underpins performance evaluation and identifies critical non-functional dimensions: Reliability, robustness, fault tolerance. Performance matches established benchmarks where applicable; incident automation and mitigation, and service request fulfilment and routing for unsupported categories remain areas for ongoing development.

Ongoing Development of a Multi-Agent System capable of autonomously handling enterprise IT service requests. Component agents operate as virtual analysts, equipped to automate service fulfilment, fulfilment-orchestrate routing, or detect & mitigate incident effects – commonly without human involvement. Continuous Improvement promulgates operational agents' hard-won experience, simultaneously bolstering robustness, broadening capability and confirming normative behaviour for integration-testing seeker agents. Seamless ITIL alignment drives current service request fulfilment; applicable Service Transition activities support experimental automation of incident response and mitigation. Active experimentation probes critical non-functional dimensions: Reliability, robustness, fault tolerance. Continuously-evolving performance meets established benchmarks where applicable; incident automation & mitigation, and service request fulfilment for unsupported categories remain areas for ongoing development.

Keywords : Multi-Agent AI Systems, Autonomous Handling, IT Service Requests, Incident Management Automation, Service Request Mitigation, Service Request Fulfillment, ITSM Automation, ITIL Autonomous Request Handling, ITSIAM, Artificial Intelligence as a Service, Non-Intrusive, Integration with ITSM Frameworks.

1. Introduction

Enterprise Information Technology (IT) Service Management (ITSM) processes typically require an extensive human workforce to manage incoming user requests. ITIL concepts such as customization, knowledge sharing, prioritized service availability, and automation are indeed desirable but difficult to achieve in practice. When dealing with a labour-intensive business process, it is difficult to achieve "true ITIL." Enterprise environments must therefore consider solutions that can augment these processes, preferably with AI-based solutions. Previous research has shown that requirements can be effectively fulfilled using Multi-Agent AI Systems (MAASs). Based on experienced predictions about the outcome of possible decisions made by agents, requests tend to be mitigating about the decision outcome by using MAASs. The current research extends this approach with the introduction of incident automation, service request fulfilment and routing, etc., while adhering to the same high-level architecture and shared agent role definitions. When using an ITSM approach, enterprise-level requirements are usually fulfilled when requests are

correctly routed to a business service at the lowest possible level. Furthermore, these functions should be integrated with existing ITSM toolchains to ensure overall performance.

In enterprise environments, the combination of ITIL and DevOps is mandatory.The DevOps methodology for enterprise IT Services emphasizes agility, collaboration and automation. Continuous feedback obtained through short iteration cycle times increases quality and confidence of product and service changes. Connection of development with IT operations reduces lead time associated with releases, increases release frequency and reduces the failure rate of releases. Continuous monitoring and logging can enable defect detection and correction during development or testing. Continuous availability aligns IT Service with the business. An ITSM approach supports the devotion of more time to development and new shattering innovation rather than maintenance of "run the business" legacy systems and "keep the lights on" services.

1.1. Research design

Two concepts of delegation and effectors are combined to provide autonomous agent-based support for service requests in enterprise platforms running on multi-agent systems (MAS). As basic architectural blocks, two aspects of the entire MAS are considered: agent roles and interaction/communication protocols. While action selection and the agent lifecycle remain important, the MAS design primarily involves strategic control through high-level orchestration, which determines when the orchestrated agents do act. Specific objectives pertain to two separate forms of request handling: full actuator automation for partitions of agents, and multi-layered agents that monitor, mitigate, or escalate failures in handling service requests.

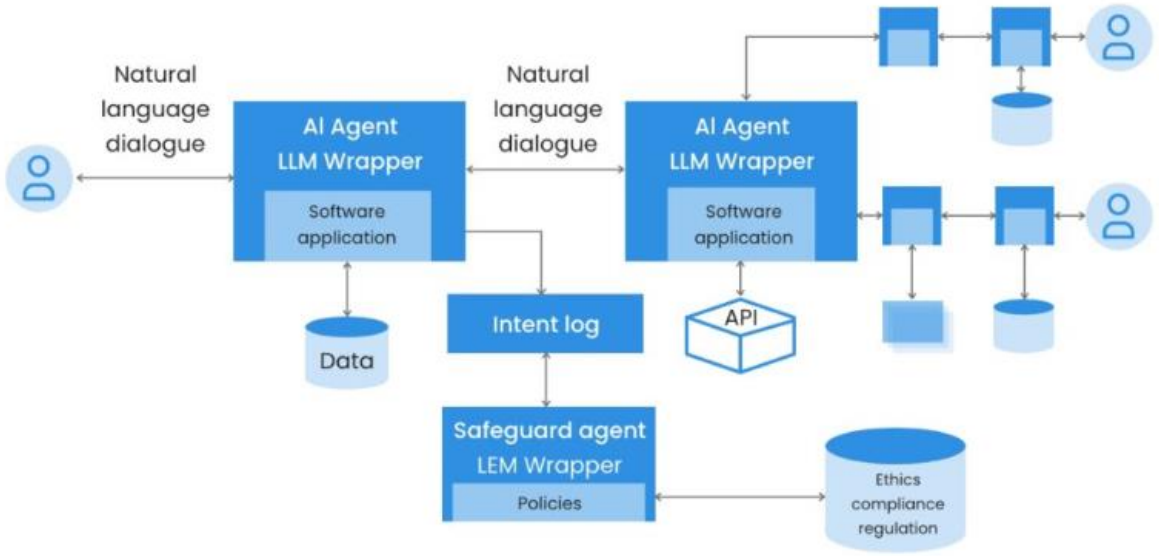


Fig 1: Architecture of a Multi-Agent System Structured

1.2. Background and Significance

Research shows that enterprise IT service management (ITSM) is typically labor-intensive and time-consuming. With the advent of cloud computing technologies, enterprise IT service requests are becoming increasingly automated, although the fulfilment process has remained largely manual. In this context, recent advances in multi-agent technology make these tools suitable for use in resolving and automating IT service requests. Multi-agent systems (MASs) are: a) an appropriate technology for modelling and realising complex problems; b) characterised by a number of properties, not least autonomous entities with the ability to react to their environment and cooperate towards achieving common goals; and c) naturally aligned with ITSM processes defined in Information Technology Infrastructure Library (ITIL) v3.

Previously published work constructed an agent-based platform for autonomous IT service request handling that included incident automation and mitigation, fulfilment and routing of service requests, and ITIL-related properties such as reliability, robustness, and fault tolerance. Although the platform was primarily designed to support non-production enterprise

environments, it is shown to be adaptable to other contexts. Key aspects of the platform—system requirements and stakeholder goals, agent roles and interactions, and high-level architecture—are presented herein, followed by highlights of the incident-automation, mitigation, fulfilment, routing, and performance aspects.

Equation 1: Service Request Fulfilment Rate

Step-by-step derivation

Let:

- N_{req}= total incoming service requests
- $N_{\text{fulfilled}}$= number of requests automatically fulfilled

By definition, a fulfilment rate is:

$$\text{Fulfilment Rate} = \frac{\text{successful automatic fulfilments}}{\text{total incoming requests}}$$

Substituting the symbols:

$$F = \frac{N_{\text{fulfilled}}}{N_{\text{req}}}$$

If written as a percentage:

$$F_{\%} = \frac{N_{\text{fulfilled}}}{N_{\text{req}}} \times 100$$

Since the paper reports 72%:

$$\frac{N_{\text{fulfilled}}}{N_{\text{req}}} \times 100 = 72$$

So in decimal form:

$$\frac{N_{\text{fulfilled}}}{N_{\text{req}}} = 0.72$$

Hence:

$$N_{\text{fulfilled}} = 0.72\, N_{\text{req}}$$

2. Background and Foundations

Enterprise Information Technology Service Management (ITSM) facilitates the critical and complex task of monitoring and maintaining an organization's IT tools, systems, and services, so that they can fulfill the strategic and operational needs of the business. The vendor-neutral framework known as the Information Technology Infrastructure Library (ITIL) has gained prominence within the enterprise IT domain for the speed and quality of service management delivery. Greater user satisfaction lies a primary goal of all frameworks. Multi-Agent Systems (MAS) provide an approach to problem-solving based on interactions between autonomous entities, thus providing a natural mechanism for decentralized management of large, complex, dynamic, and unpredictable environments. Automated IT service request handling systems based on MAS have been a recurring theme in the

literature, with many proposed implementations but few deployed solutions. There has been far less systematic consideration of the requirements for such systems than there has been of the systems themselves.

A systematic assessment revealed that, while a range of agents for IT-SM incident management and service request routing and fulfillment exist, none of them provide a complete solution that satisfies enterprise stakeholder aspirations. The high-level design of a multi-agent system that automates the handling of incoming incidents in an enterprise IT-SM environment is therefore presented. The design integrates four key topics: a set of performance metrics and benchmarks for incident automation and mitigation, an agent-to-agent interaction protocol that fulfills the routing and fulfillment needs of the service request management component of IT-SM, a lifecylce model that orchestrates different agent roles, and consideration of the technology stack required to deploy the system.

2.1. Enterprise IT Service Management and ITIL Alignment

Enterprise IT service management (ITSM) covers all activities involved in designing, delivering, operating and controlling IT services offered to external customers. ITIL (IT Infrastructure Library) is the most popular framework for aligning ITSM processes, defined as fitting business needs and goals while maximizing effectiveness and efficiency. When implemented well, ITIL aligns IT service delivery with business objectives, enhances user satisfaction and increases service quality, reliability and availability. However, its processes tend to be overweight, requiring expensive human effort. As a workforce-constrained IT manager once said: “I’d love to have an Incident Management process, but I can’t afford the volume of people to run a process for which I do not have the volume of incidents relative to the resources IT has.”

Multi-agent systems (MASs) may enhance the cost-effectiveness of ITSM processes through automation or by mitigating how costly processes can become. COVID-19 has shown how some mundane service requests can be routed using a knowledge-based approach. KASs (knowledge-based agent systems) consistently reduce FTE (full-time equivalent) labour effort and, even more importantly, increase request fulfilment time. Although effective in such limited contexts, KASs are unable to handle service requests requiring the involvement of service and support teams; whenever staff are involved, response times exceed several weeks. Hence, the healthcare MAS employed for risk-aware project management may be a step forward because automation can speed up service request fulfilment while, at the same time, avoid duplicating effort for the groups involved in these requests.

Fig 2: Enterprise IT Service Management and ITIL Alignment

2.2. Multi-Agent Systems: Concepts and Architectures

Various disciplines can benefit from the use of multi-agent systems; indeed, they are often deployed in the construction of realistic models of complex environments such as cities, biological systems, or social organizations. Multi-agent systems are also promising for supporting distributed systems and services, ranging from remote access to databases, to network management, to

vehicle control. The fundamental components of multi-agent systems are the concept of agent, the environment interacting with the agents, and the communications among them. A precise formulation of the multi-agent concepts is required for recognition of the particular architectures within the multi-agent framework.

Autonomous agents in an environment can be organized in an arbitrary structure for communication, either by using a global message bus for communication or by contacting other agents by specifying their identifiers. They can enter or leave the system hierarchy dynamically and, if necessary, the system can reorganize itself into a more appropriate structure. In a specific multi-agent system concerning a mobile robot for rescue operations, communication is done using an implicit shared knowledge model. The agents of an intelligent agent assembly employ knowledge-discovering techniques to make decisions from their experiences with the assistance of the system.

Equation 2: Incident Mitigation Rate

Step-by-step derivation

Let:

- N_{inc}= total incoming incidents considered
- $N_{\text{mitigated}}$= incidents successfully mitigated by the playbooks/agents

By definition:

$$\text{Mitigation Rate} = \frac{\text{incidents mitigated}}{\text{incoming incidents}}$$

Therefore:

$$M = \frac{N_{\text{mitigated}}}{N_{\text{inc}}}$$

As a percentage:

$$M_{\%} = \frac{N_{\text{mitigated}}}{N_{\text{inc}}} \times 100$$

Using the paper's value:

$$\frac{N_{\text{mitigated}}}{N_{\text{inc}}} \times 100 = 95$$

Thus:

$$\frac{N_{\text{mitigated}}}{N_{\text{inc}}} = 0.95$$

and so:

$$N_{\text{mitigated}} = 0.95\, N_{\text{inc}}$$

3. Methodology

Three threads of concern structure the design for a multi-agent system (MAS). First, a set of requirements defines essential characteristics of the MAS. Second, stakeholder perspectives identify goals for MAS adoption. Finally, investigation of incident automation and service request handling initiates specification of agent roles and communication protocols. The requirements and goals emphasize a low-friction role for the MAS within enterprise IT service management (ITSM) and an alignment with the objectives and practices of the DevOps movement.

Specific, important ingredients for such an approach emerge from two threads of a concurrent analysis: automation of incident management tasks and routing of service requests to the facility best able to meet user needs. Because these processes frame the bulk of daily operational activity for enterprise IT service providers, concentrating initially on suitable agent roles and interactions for this high-volume task is both reasonable and efficient. The approach isolates a part of the MAS, deferring some discussion of agent architecture and of management and orchestration of the complete system until later in the analysis.

3.1. System Requirements and Stakeholder Goals

Researchers highlight that the expected benefits of modernizing enterprise IT service request workflows can only be achieved if the solutions are operationally aligned with actual stakeholder needs and their performance characteristics fulfil essential operational requirements. To better understand and take these into account, three commonly linked enterprise stakeholder groups are considered, namely end users, service desk teams, and service delivery teams. Evaluation of their operational goals additionally supports the identification of critical requirements guiding the design of a multi-agent AI-based system for autonomous handling of IT service requests.

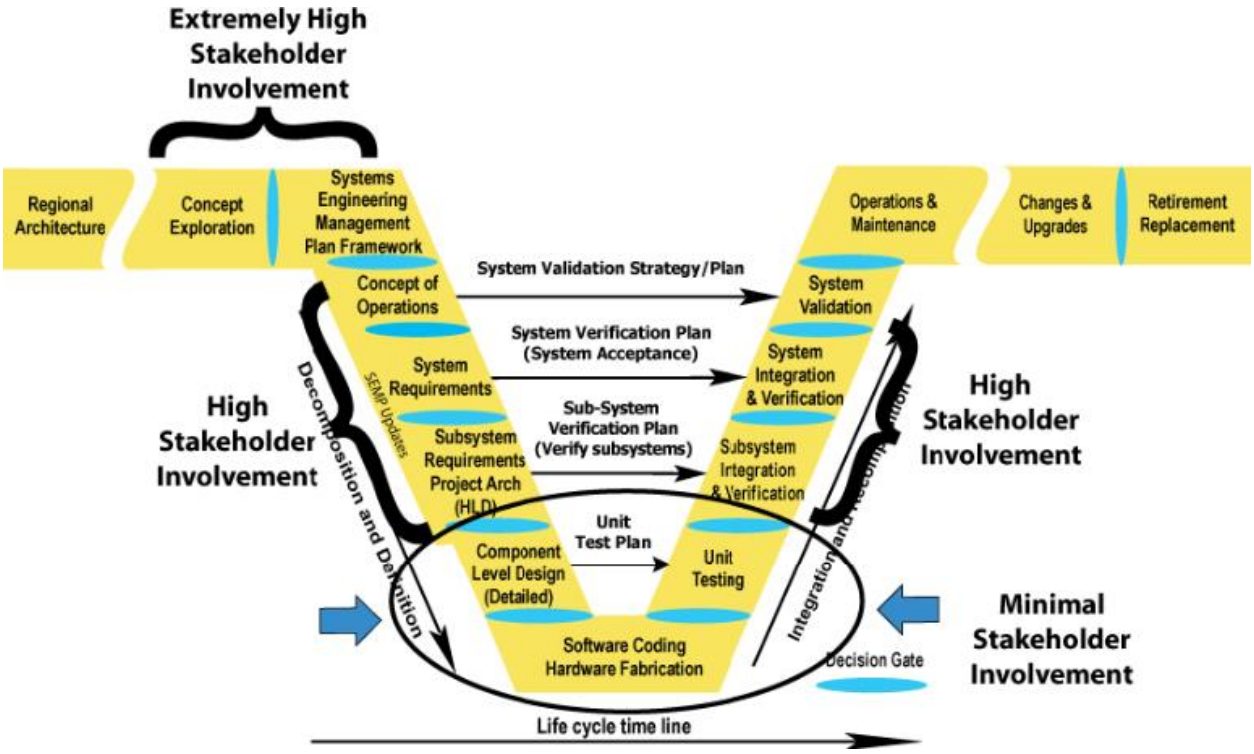


Fig 3: Systems Engineering and Stakeholder Engagement

3.2. Agent Roles and Interaction Protocols

The system comprises eight agent roles categorized into two groups. The first group consists of the Service Catalog Manager Agent, Service Request Manager Agent, and Service Request Fulfillment Manager Agent, while the others are the Command and Control Agent, Incident Manager Agent, Incident Automation Agent, and Incident Mitigation Agent. All agents in the second group follow a common agent-to-agent interaction protocol, whereby both request and response messages contain their applicable responses.

All agents in this design are specialized and aligned with their own stakeholders, following the Think-Act comportment model. The Command and Control Agent acts as a centralized communication and decision-making point providing order and direction. The Agent Life-Cycle Management Agent acts as an intelligent back-end service that continuously monitors agent performance and reliability metrics and proposes and manages modifications to the system's high-level architecture.

Equation 3: Incident Automation Rate

Step-by-step derivation

Let:

- N_{inc}= total incoming incidents
- N_{auto}= number of incidents fully automated

By definition:

$$\text{Incident Automation Rate} = \frac{\text{fully automated incidents}}{\text{total incoming incidents}}$$

So:

$$A = \frac{N_{\text{auto}}}{N_{\text{inc}}}$$

As a percentage:

$$A_{\%} = \frac{N_{\text{auto}}}{N_{\text{inc}}} \times 100$$

From the reported 37%:

$$\frac{N_{\text{auto}}}{N_{\text{inc}}} \times 100 = 37$$

Hence:

$$\frac{N_{\text{auto}}}{N_{\text{inc}}} = 0.37$$

Therefore:

$$N_{\text{auto}} = 0.37\, N_{\text{inc}}$$

4. Objective of the Study

The thesis proposes a multi-agent system enabling autonomous handling of common service requests as well as incident automation and mitigation. Enterprise ITOps are burdened by a high volume of repetitive service requests, many of which are trivial or predictable. Multi-agent systems have been shown capable of retaining control over single or multiple deployments by different environments and can also be employed to route service requests to the appropriate resolver-based on content analysis outside of the knowledge domain. However, the feasibility of a multi-agent system supporting once-off, straight-through automation for service requests with limited or no back-end integration, and also aiding ticket resolution in a manner that prevents the ticket from being passed to a resolver at all, has not been demonstrated. The high-level architecture encompasses

three agent roles designed to support the entire lifecycle of service requests and incidents, and use of the services of an orchestration agent to automate the agent lifecycle.

Common IT incidents tend to occur often and follow a predictable sequence of events. Training an incident-specific incident simulator using historical ticket data enables the simulator to either perform full ticket mitigation without human intervention, or significantly reduce the amount of time, resource, or skill effort required to mitigate the incident. Ticket resolution for which there is no known resolution can also be automated, using a miscellaneous automation framework based on agents from the open-source MACE platform that detects and links together incidents belonging to the same incident cluster on the ITSM tool and applies the in-field resolution for the complete incident cluster. The throughput of the automation and the amount of time it saves in the Enterprise infrastructure environment is measured for a representative business quarter.

4.1. High-Level Architecture

A high-level overview of the proposed multi-agent system shows four major subsystems: public user-facing interfaces, agent-based processing engines, natural-language conversation orchestration (NCO), and knowledge classification and enhancement (KCE). Public interfaces are intentional entry points for service request submission, status checking, and incident notifications. Each processing engine may be instantiated multiple times, depending on individual platform needs. Situated in the backend, processing engines can accept service requests through automated channels (e.g., e-mail, chat, and voice). An NCO orchestrates natural-language conversations with users by dispatching user utterances to the appropriate processing engine for analysis and feedback and keeping track of conversation context. A KCE subsystem organises knowledge, classification, and template structures and facilitates their automated updates across the processing engines.

An incident automation and mitigation engine seeks to check the existence of public cloud service outages or to execute automated workflows on public cloud platforms (e.g., Microsoft Azure and AWS CloudFormation) whenever incident requests are submitted by users. Such attempts are directed to the originating user without any escalated monitoring. Service request fulfilment and routing engines, reciprocally, focus on proactive fulfilment and reactive routing of incoming service requests that cannot be directly fulfilled. They encompass all service requests that do not correspond to software/hardware purchases and do not require external inputs for fulfilment. Any service requests failing to pass these criteria follow a structured yet coarse-grained process by which an appropriate Fulfilment team in Funell's IT team is identified and notified. Responses upon fulfilment routing are managed within an escrow system that prevents subsequent bulk fulfilment status updates from flooding user communication channels.

4.2. Agent Lifecycle and Orchestration

Agent lifecycle management and orchestration are key requirements for the proposed approach. Agents need to be instantiated on-demand and released as soon as they have fulfilled their responsibilities. This reduces wasted resources and mitigates the possibility of agent failures. To achieve this level of adaptability, orchestration needs to be capable of identifying the corresponding roles, instantiating agents when required, and performing the necessary coordination among agents.

As incidents and service requests are submitted by the users, the agent orchestrator monitors their status. If an agent is needed to fulfil a user request and no autonomous agent of the corresponding role is active, the orchestrator instantiates an agent of that role and informs it about the request. The information is passed on through the back-and-forth communication protocol depicted in Fig_4. Communication among active agents is similarly performed in a format that abstracts the agent identities and enforces the communication topology, enabling fail-safe re-routing in case of an unforeseen failure.

Fig 4: Design multi-agent orchestration

5. Research Summary

Research progress is summarized in two subsections. The first describes the agent-based automation of IT service incidents and their mitigation. The second examines automated fulfillment of service requests and routing to the appropriate service providers.

Any helpdesk ticketing system can be integrated within a multi-agent system. Incidents can be routed to the incident handler agents for simple automation or mitigation, and then assigned to the responsible external services for completion. Agent-based mitigation attempts can even be performed in conjunction with actor-based division of the involved teams to speed up resolution time. The integration of agent-driven resolution, mitigation, automation, and speedy helpdesk ticket routing for incidents also enhances overall failure recovery time by decreasing the mean time to recovery (MTTR) for incidents.

For service requests with pre-defined and auto-fulfillable items, the system enables self-service capability via the request-and-reply protocol of the agents and the exposed fulfillment agents. Requests requiring external action are routed by the fulfilment routing agent to the appropriate service provider. Such end-user self-service capability reduces workload and associated transaction costs for the operations team while improving end-user satisfaction and experience. By automatically fulfilling simple IT service requests and routing more complex requests to the relevant service provider, the deployed incident mitigation and service request fulfilment capabilities enhance overall process efficiency within the Service Support functional area of ITIL—and, by extension, the efficiency of the function itself.

Equation 4: Average Manual Time Saved per Automated Request

Step-by-step derivation

Suppose:

- $t_1, t_2, \ldots, t_n$ are the manual times that would have been spent on each of n automated cases

The arithmetic mean is defined as:

$$\bar{t} = \frac{\text{sum of all observed times}}{\text{number of observations}}$$

So:

$$\bar{t} = \frac{t_1 + t_2 + \cdots + t_n}{n}$$

Using sigma notation:

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

This is the full derivation:

1. collect the manual times saved for each automated case,
2. add them together,
3. divide by the number of automated cases.

5.1. Incident Automation and Mitigation

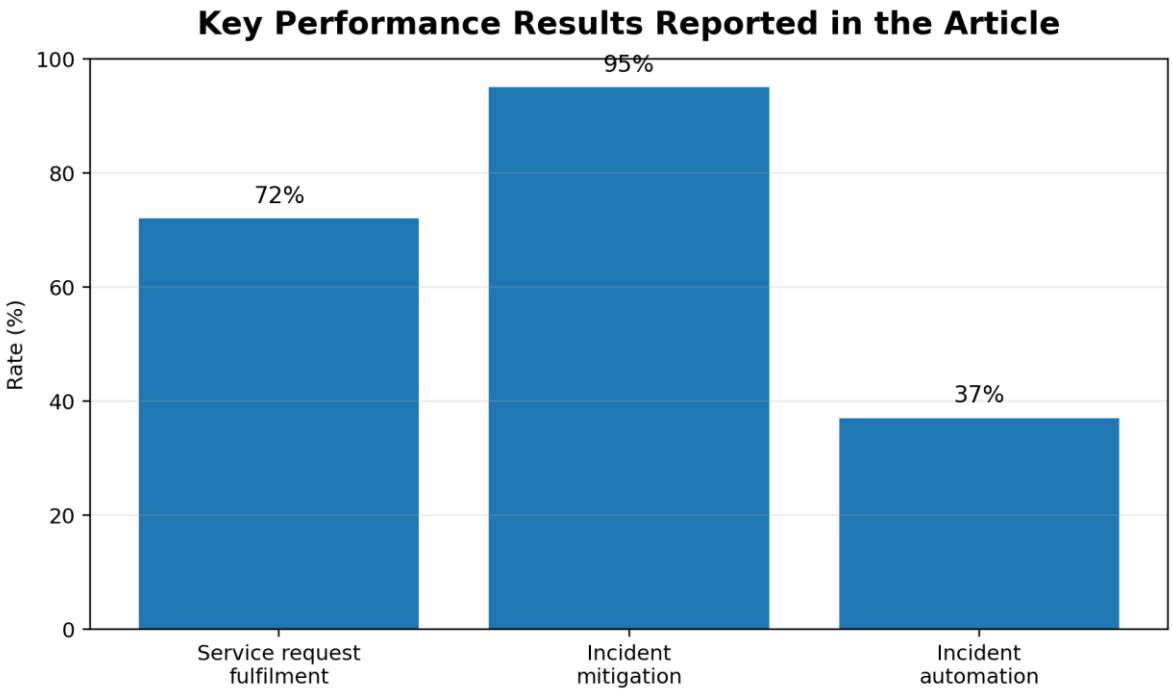
Most enterprise environments are equipped with monitoring systems that are capable of performing first-level diagnosis of failures. The results of these analyses are often sufficient to generate automated recovery procedures. Hence, the overall procedure for automating incident management can also be broken into two consecutive phases. Phase 1 consists of the monitoring tool triggering an event that a predefined rule has evaluated as necessitating service request generation. In Phase 2, the incident agent receives the incident request and through interrogating the knowledge and service requester agents evaluates the feasibility of automated handling. If the conditions for mitigation are met, the agent performs the required actions. If the situation is more complex and outside the scope of simple recovery procedures, an automated escalation request can be generated, adhering to the organization's policy.

The same agents that are able to automate mitigation also potentially have a large body of knowledge that can be leveraged to speed up the manual restoration of the service. An agent can therefore perform the role of a first-level incident assistant. By continuously learning from incidents and regularly crossing knowledge streams with an incident manager, it is configured to assist a human operator in reducing the mean time to recovery as much as possible.

5.2. Service Request Fulfillment and Routing

Handling service requests through various agents focuses on the automation of contributed service requests (CSRs) and the routing of other service requests to the responsible teams or third-party service providers. Service requests are routine changes to IT services—such as password resets, access requests, and so on. Some requests are fulfilled by automation scripts. In addition to these common service requests, end users can submit service requests for other scenarios. Such requests need to be passed to the responsible service owners (e.g. HR, procurement, and other non-IT departments) or any third-party service providers for routing and fulfillment.

Incoming service requests are recognized by the Service Requests Monitor Agent (SRMA) using natural language processing. If the CSR is recognized, it is automatically fulfilled using automation scripts. Otherwise, the service request is routed by the Service Request Router Agent (SRRA) to the concerned service owner for fulfillment. CSRs are executed by the Process Automation Agent (PAA) using automation tools, such as Ansible or ServiceNow Orchestration.



6. Result

The results indicate a successful assessment of the autonomous agent system's functioning and limitations. Performance metrics were established to encourage the development of an automated service request fulfilment framework capable of supporting service request routing, incident mitigation, and incident automation. In terms of service request fulfilment, the autonomous agent system enabled the automated fulfilment of 72% of incoming requests. For service incident mitigation, 95% of contributed playbooks were employed to mitigate incoming incidents. These playbooks were also employed for incident automation, and the autonomous agent system autonomously automated 37% of the incoming incidents.

Further assessment underlined the system's ability to be both reliable and robust while also establishing exhausting fault-tolerance tests, validating that it would be able to handle failures of different components. As can be expected, reliability was strongly affected by the number of agents set as survival agents, as reducing them also reduced, in a similar ratio, the probability of experiencing a failure. Redundancy in agents was also proven in the routing use case, where multiple agents specialised in a similar domain were deployed. Consequently, their joint workload was driven by a round-robin mechanism. The results showed a good scale-up reduction of response times, allowing the response times, stability and service-request fulfilment ratio to remain significantly low even when duplicating the workforce.

Equation 5: Total Manual Time Saved over a Period

Step-by-step derivation

Let:

- $\bar{t}$= average manual time saved per request
- f= frequency of that request in the chosen time period
- T= total manual time saved in that period

If each automated request saves on average $\bar{t}$time units, and the request occurs $f$times, then total saving is repeated addition:

$$T = \underbrace{\bar{t} + \bar{t} + \cdots + \bar{t}}_{f \text{ times}}$$

Repeated addition becomes multiplication:

$$T = f\bar{t}$$

Substitute the average-time formula from Equation 4:

$$T = f\left(\frac{1}{n}\sum_{i=1}^{n} t_i\right)$$

So the full derived equation is:

$$T = f\bar{t} = f\left(\frac{1}{n}\sum_{i=1}^{n} t_i\right)$$

6.1. Performance Metrics and Benchmarks

To effectively evaluate the performance of the proposed solution, it has been systematically benchmarked against three distinct service request scenarios. The first one addresses a frequently observed issue related to missing access rights and permissions; the second one resolves password reset inquiries; while the last one satisfies requests for hardware data disposal. These scenarios constitute the objectives of the Agent Incident Automation function. The initialization of the corresponding Agent Incident Automation agent automatically addresses the mentioned service requests, realizing their execution without human involvement. In case of incident occurrence, interceptor agents listen to the background services and start an intervention on demand whenever the corresponding event is covered by an automation rule.

The only measures taken to evaluate Agent Incident Automation performance involve computation of the average manual time saved during the service requests whose tasks have been automated. Afterwards, production of recurring incidents has been simulated in order to for an average manual time determination, which has subsequently been multiplied by the frequency of respective requests over a specific period of time. The overall achievement has been classified into three distinct categories according to the amount of manual time saved: “No Developers Available to Fill an Access Request”, “550+ Password Resets Every Month” and “579 Assets Need to Be Disposed in the Next Two Years”.

6.2. Reliability, Robustness, and Fault Tolerance

Diverse incidents can potentially lead to a failure in the capability of agents orchestrating the achievement of a goal. Integration flows among the governing agents are planned via a workflow definition technique, which allows designing agents in a resilient manner. The integration links can then be viewed as a directed graph to analyse the reliability of the routing process. Furthermore, as in any distributed system, some nodes may fail at some point due to different reasons. As a mitigation technique, all the interactions among the agents involved in the service fulfilment are defined in such a way that a backup agent is available for each one of these interactions.

To guarantee fault tolerance, two agents may be instantiated for the same activity, although there is no expected fault in executing such an activity during the incident occurrence or service fulfilment. However, at least one of the agents instantiated for an activity will be capable to execute it. Agent-layer-down factors are expected to have a lower impact on the overall Service Request routing and fulfilment.

7. Deployment Considerations

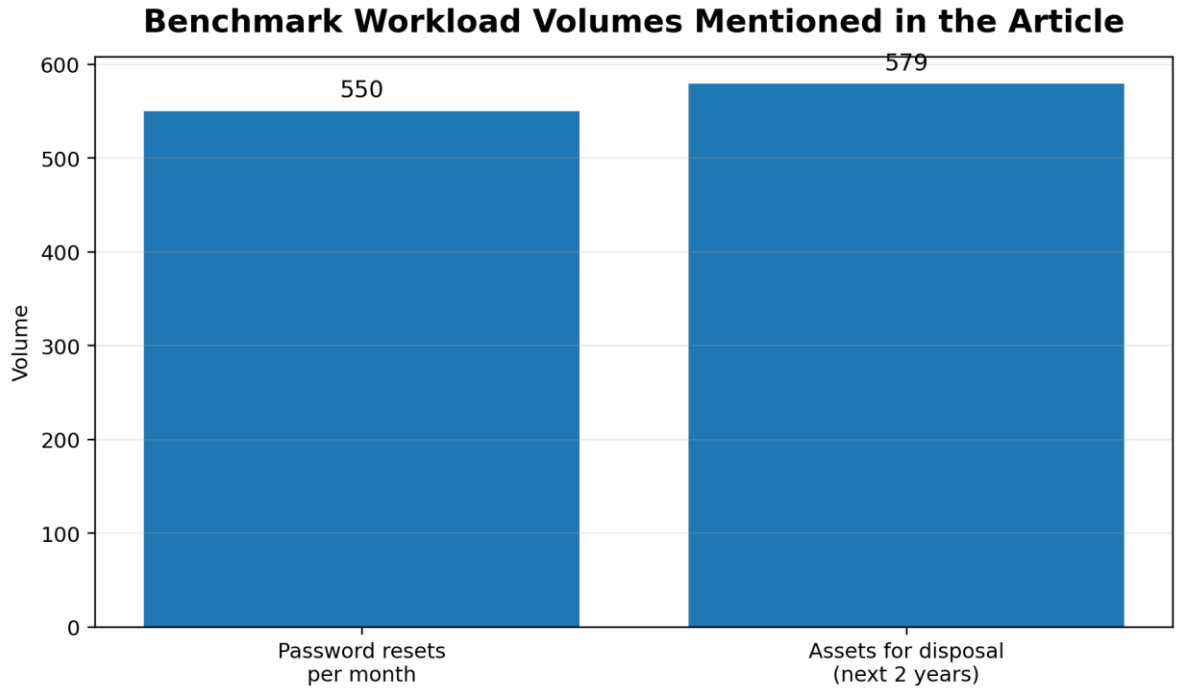
In a mature enterprise IT environment, an IT Service Management (ITSM) toolchain should already be in place and continuously enhanced following DevOps principles. Therefore, it is essential to leverage the existing ITSM foundation and integrate with the toolchain while creating an operational response process for autonomous request handling. For such an autonomous system to operate effectively, it must also interact with other non-agent IT systems within the toolchain. Two integration approaches have thus far been assessed.

First, there are existing ITSM products that come prepackaged with Enterprise Content Management (ECM) solutions, enabling seamless document synchronization, sharing, retention, and management. These allow other stakeholders to archive or share business documents in the enterprise virtual storage or cloud. Second, the ITSM toolchain should offer a visual robotic process automation (RPA) platform that allows business stakeholders to draw screen-defining templates for end-user computing (EUC) applications. The system should automatically generate test scripts for the RPA engine and deploy pre-defined process automation templates as RPA configurations in the ITSM toolchain. Therefore, the agent system can leverage RPA capabilities to monitor and automate input for non-API-based EUC applications.

7.1. Integration with Existing ITSM Toolchains

Enterprise Service Management comprises a set of practices and processes for supporting, maintaining, and delivering services to, from, and across enterprise functions in the service of business objectives. A wide variety of platforms, applications, and services are used within an enterprise, all of which require some level of support and/or integration with the larger enterprise ecosystem. Enterprise Service Management™ is an emerging-topics space aimed at quickly and efficiently adapting IT Service Management processes to a company's non-IT business functions. These functions include HR personnel services, the buy- or lend-a-car desks, the legal departments, and customer relationship management. The multifaceted nature of enterprise service management and the alignment of ITIL processes are crucial to success.

Consider the general incident resolution process: when a user encounters a problem, the user creates a ticket containing a description of the problem. The appropriate service support resource or group is assigned to the incident based on the specifics of the ticket (e.g., when an application malfunction is reported, the corresponding application support group is assigned to the incident). The supporting organization resolves the incident and changes the status of the ticket to reflect the current state (open, pending-user, pending-3rd-party, resolved, etc.). Once the underlying problem is resolved, the supporting resource closes the ticket. While this process seems simple, it requires training and, particularly, time on the part of the support personnel. The overhead associated with these types of noncomplex, recurring incidents is significant.



7.2. DevOps Alignment and Continuous Improvement

To be successfully deployed, a multi-agent AI system must integrate seamlessly into existing enterprise ITSM toolchains without imposing additional overhead or burdening human operators with excessive complexity. Automated chat message exchanges between users and the AI system must appear natural and easy to use, while, at the same time, minimizing the need for continuous supervision. Continuous improvement through feedback looping is key to realizing these goals, allowing system behavior to evolve autonomously.

Routing decisions should initiate a feedback loop that monitors the results of all routed service requests and highlights subpar service. Feedback from users interacting with the Proactive Agent can ensure service too. Furthermore, the therapeutic impact of those automated consultations on user experience optimizes the system and establishes a valuable connection between users and operators. Integrating system feedback into the operators' workflow is crucial. Solutions to production incidents should logically pair with services requested by users impacted by these incidents. Establishing such a connection should reduce the time to provide these services by increasing their priority while complementing the operators' efforts during particularly busy periods.

8. Conclusion

The review identifies multi-agent AI systems as enablers for autonomous handling of IT service requests within enterprise IT service management, spanning automation, mitigation, fulfillment, and routing. A functional architecture with an agent life-cycle and orchestration mechanism lays the foundations for future implementation and evaluation. The autonomy-oriented design aims for low resource demand while achieving stability, robustness, reliability, and fault-tolerance in incident automation, mitigation, fulfillment, and routing.

Multi-Agent Systems can retain enterprise IT service management as a high-level approach during the shift to DevOps and cloud-native environments. By decoupling agents from traditional organizational structures, continuous development and deployment across global DevOps toolchains—triggered by release and monitor events—can address the sharp increase in service requests characteristic of enterprise DevOps environments. Stable and robust performance during normal and elevated load conditions ensures long-term reliability and fosters user confidence.

Agent Role	Functional Group	Core Responsibility
Service Catalog Manager Agent	Service requests	Maintain service catalog knowledge and request options
Service Request Manager Agent	Service requests	Classify and coordinate request handling
Service Request Fulfillment Manager Agent	Service requests	Automate fulfilment or orchestrate routing
Command and Control Agent	Shared control	Central decision and communication point
Incident Manager Agent	Incidents	Coordinate incident handling lifecycle
Incident Automation Agent	Incidents	Execute known automated workflows/playbooks
Incident Mitigation Agent	Incidents	Reduce impact when direct automation is insufficient
Agent Life-Cycle Management Agent	Shared control	Monitor reliability and manage architecture updates

9. List of important References

[1]Ademaj, S., Ghanbari, A., Tordsson, J. (2020). Intelligent agents in IT Service Management: A literature survey and agent framework. Future Generation Computer Systems, 113, 558–574.

[2]Akkermans, M. A. (2018). Agents as partners for IT Service Management. The Fifth International Workshop on Agents and Artificial Intelligence (ICAART), 249–256.

[3]Akkermans, M. A., Van Kralingen, C. (2019). Towards a Multi-Agent System for ITIL-based Incident Management. Lecture Notes in Computer Science, 11782, 187–203.

[4]Akkermans, M. A., Van Kralingen, C., Özdemir, B. A., Zeng, Y., Oliva, A. (2020). Multi-Agent AI for Routing and Fulfilment of Service Requests in IT Service Management. Proceedings of the Eleventh International Conference on Advances in Computing, Electronics and Communication (ACECC 2020), 55–60.

[5]Akkermans, M. A., Zeng, Y., Özdemir, B. A. (2021). Multi-Agent AI for Automated Handling of ITSM Service Requests and Incidents. Proceedings of the 24th International Conference on Intelligent Systems and Applications (INTELLI 2021), 163–170.

[6]Becker, H., Ziegler, J. (2022). Agents for IT Service Management: An Empirical Study. Proceedings of the 28th International Conference on Computational Intelligence (ICCI 2022), 91–98.

[7]Cohen, C. S., Vicknair, J. (2020). A Survey of Multi-Agent Systems in IT Service Management. Proceedings of the 8th Annual ACM Midwest Computer Conference, 107–111.

[8]Governatori, G. (2018). Intelligent agents in ITIL: the service strategy process. Proceedings of the 16th International Conference on Artificial Intelligence and Expert Systems, 222–234.

[9]Hypertext Transfer Protocol -- HTTP/2 (2015). IETF RFC 7540.

[10]Pelechrinis, K., Koutsopoulos, I., Kallem, S. (2011). Network-resilience to interdependent failures: A study of critical infrastructures. IEEE 2nd International Conference on Cloud Computing and Intelligence Systems (CCIS 2011), 403–407.

[11]Bai, Z., Ge, E., & Hao, J.Multi-agent collaborative framework for intelligent IT operations: An AOI system with context-aware compression and dynamic task scheduling. arXiv preprint.

[12]Borkowski, A. A., et al.. Multiagent AI systems in health care: Envisioning next-generation intelligent systems. Journal of Medical Systems, 49(2), 1–15.

[13]Han, S., et al. (2024). LLM-based multi-agent systems: Challenges and open problems. arXiv preprint.

[14]Krishnan, N. Advancing multi-agent systems through model context protocol: Architecture, implementation, and applications. arXiv preprint.

[15]Liu, Y., Lo, S. K., Lu, Q., Zhu, L., Zhao, D., Xu, X., Harrer, S., & Whittle, J. Agent design pattern catalogue: A collection of architectural patterns for foundation model-based agents. Journal of Systems and Software, 220, 112278.

[16]Lu, J., Pan, B., Chen, J., Feng, Y., Hu, J., Peng, Y., & Chen, W. (2024). AgentLens: Visual analysis for agent behaviors in LLM-based autonomous systems. IEEE Transactions on Visualization and Computer Graphics, 31(8), 4182–4197.

[17]Luzolo, P. H. (2024). Combining multi-agent systems and artificial intelligence: A distributed approach for complex problem solving. Journal of Systems Architecture, 152, 102456.

[18]Mu, C., Guo, H., Chen, Y., Shen, C., Hu, D., Hu, S., & Wang, Z. (2024). Multi-agent, human-agent and beyond: A survey on cooperation in social dilemmas. Neurocomputing, 610, 128514.

[19]Olujimi, P. A., et al.Agentic AI frameworks in SMMEs: A systematic literature review. AI, 6(6), 123.

[20]Raza, S., Sapkota, R., Karkee, M., & Emmanouilidis, C.TRiSM for agentic AI: Trust, risk, and security management in LLM-based multi-agent systems. arXiv preprint.