Service Graph Connector

Mohammed Shakeer Bandrevu

Senior Software Developer

Abstract:
Ask any IT operations manager what keeps them up at night and a poorly maintained CMDB will almost certainly come up. Getting that database to actually reflect what is running in the environment, in real time, without a small army of people updating it manually, has been an unsolved problem for years. Service Graph Connector is arguably the most practical answer the industry has produced so far. In this paper, we examine how the SGC works, what makes its architecture different from earlier approaches, and what organizations can realistically expect when they deploy it. We also look honestly at the challenges, because the implementation is not always smooth, and we close with a discussion of where things are likely headed as AI and predictive analytics get folded into the mix.

Keywords: ServiceNow, Service Graph Connector, CMDB (Configuration Management Database), IT Service Management (ITSM).

1. INTRODUCTION

1.1 IT Service Management (ITSM) and Configuration Management Databases (CMDB)

IT Service Management, or ITSM, is best understood as the set of disciplines that govern how an organization plans, delivers, and keeps improving the technology services its people depend on. It is not a single tool or platform. It is a way of thinking about IT as a service function rather than a collection of ad hoc technical tasks. Processes like incident management, change coordination, and service desk operations all fall under its umbrella. The ITIL framework gave practitioners a common vocabulary for this work decades ago, and that vocabulary has stuck, even as the platforms built around it have evolved considerably.

The Configuration Management Database sits at the center of any serious ITSM effort. Simply put, it is a record of everything in the IT environment: servers, applications, network devices, virtual machines, and the relationships between all of them. When the CMDB is accurate, life gets easier. Incidents get resolved faster because engineers can see exactly what is connected to what. Change requests get properly assessed because the blast radius of any given modification is visible. Auditors get satisfied because there is a documented record of the infrastructure. When the CMDB is wrong, all of that falls apart. We have seen this play out in practice, and the cost is real.

1.2 Importance of Data Integration in IT Operations

Here is the fundamental problem. IT environments today are not static. Cloud resources spin up and down on demand. Applications get deployed across multiple vendors. Network configurations change weekly. Keeping a CMDB current under those conditions requires either a very large team or a very good, automated integration process. Most organizations have neither.

When data integration breaks down, the CMDB starts drifting. New assets appear in production without any corresponding record. Old records linger for equipment that was decommissioned months ago. The relationships between components go unmapped. None of this is catastrophic on its own, but it accumulates. Eventually someone makes a change based on what the CMDB says, discovers mid-incident that the CMDB was wrong, and the failure cascades. Good data integration prevents exactly that kind of compounding failure.

1.3 ServiceNow and Service Graph Connector (SGC)

ServiceNow has become the dominant platform for enterprise ITSM, and its CMDB capabilities are among the most mature in the industry. Its Common Service Data Model, or CSDM, gives organizations a standardized framework for classifying IT assets and their interdependencies. That standardization matters more than it might seem at first. Without a shared schema, every integration becomes a custom project.

The Service Graph Connector is ServiceNow's answer to the data integration problem we described above. It pulls CI data from external systems, including cloud platforms, on-premises monitoring tools, and third-party asset management solutions, and loads it into the CMDB automatically. Crucially, it applies validation and transformation logic during that process rather than after, so what ends up in the CMDB is clean, relationship-aware, and compliant with CSDM standards. The result, in theory and in practice, is a CMDB that stays synchronized with the actual environment without requiring constant manual intervention.

1.4 Research Objectives and Scope

Our goal in this paper is to give practitioners a clear-eyed view of what the Service Graph Connector actually does and what deploying it looks like in practice. We cover the architecture and core features in enough detail to be useful to someone evaluating or implementing the tool. We then examine the real-world challenges, which are often underplayed in vendor documentation. The paper ends with a look at the trajectory of the technology, including where AI integration and predictive capabilities are likely to take things. We have tried to be specific rather than general throughout, because vague claims about 'digital transformation' do not help anyone make a better implementation decision.

2. BACKGROUND AND LITERATURE REVIEW

2.1. Evolution of IT Service Management (ITSM)

ITSM did not start out sophisticated. Early IT support was reactive by nature. Teams fixed things when they broke, tracked hardware on spreadsheets, and responded to requests as they came in. That approach worked fine when IT environments were small and relatively stable. It stopped working when organizations began depending on technology in more fundamental ways.

The ITIL framework emerged from that tension. It gave IT organizations a structured way to think about service lifecycles, change management, and problem resolution. Adoption was uneven, and implementation quality varied wildly, but the underlying principles were sound. Then digital transformation arrived and raised the stakes again. Cloud adoption, DevOps practices, and the explosion of SaaS applications created environments that change faster than any manual process can track. Modern ITSM has had to adapt to that velocity, and the CMDB sits squarely at the intersection of that challenge.

2.2 Current Methods for Data Integration into CMDB

For a long time, organizations had three options for populating their CMDBs: manual data entry, batch imports, and custom scripts. None of them were good. Manual entry is slow and error-prone at scale. Anyone who has watched a team of engineers spend two weeks reconciling a CMDB after a data center migration knows how painful that gets. Batch imports are faster but they create predictable staleness windows, which is its own kind of problem. Custom scripts are the most flexible option but they require ongoing maintenance, and they break every time an upstream system changes its API.

API-based integrations helped somewhat. Real-time synchronization became possible, and the volume of manual work dropped. But data quality management still landed on the implementing team. Every organization ended up building its own mapping logic, its own validation rules, its own reconciliation processes. That is expensive to build and even more expensive to maintain. ServiceNow's Service Graph Framework was a direct response to that pattern. The idea was to standardize the integration approach so that organizations could stop reinventing the same wheel.

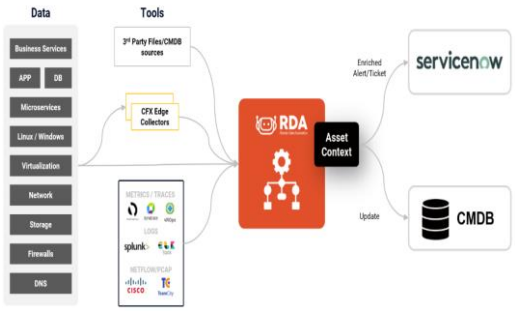

Figure - 1: CMDB Data Ingestion Pipeline Diagram

2.3 Service Graph Framework Introduction

The Service Graph Framework is best understood as a shift in who owns the integration problem. Before it existed, each organization was responsible for building and maintaining its own connectors. After it, that responsibility moves largely to ServiceNow. The framework provides pre-built integration patterns that conform to the CSDM, which means a connector pulling data from AWS produces the same kind of CMDB records as a connector pulling from a network monitoring tool. That consistency is genuinely valuable.

Two components matter most. The Service Graph Connector handles the actual data movement, including discovery, mapping, and validation. The CSDM provides the classification vocabulary that makes all of that output consistent and interoperable. Together, they address both the technical and the organizational sides of the data quality problem. We have found that organizations sometimes focus too heavily on the technical side and underinvest in getting their CSDM alignment right upfront. That tends to cause problems later.

2.4 Related Work on ServiceNow Integration

Research on ServiceNow's CMDB integration capabilities has grown alongside platform adoption. Patel et al. (2019) conducted a detailed study of CMDB architecture in ServiceNow environments and found that data consistency was the primary challenge in multi-source configurations. Their work showed that automated validation reduced reconciliation overhead meaningfully, though they also noted that the initial mapping effort was often underestimated by implementing teams.

Chinthapatla (2024) took a forward-looking angle, examining how AI could be layered onto CMDB management to anticipate configuration drift before it causes operational impact. That work is relevant to our discussion in Section 6. Kario (2018) looked at the user experience side of ServiceNow's cloud solution and noted that reducing manual CMDB maintenance tasks freed IT staff for higher-value work, which is an outcome we think gets too little attention in most ROI analyses. Ale (2024) reinforced a related point from a metrics perspective, demonstrating that dashboard reliability depends directly on CMDB accuracy, an obvious finding perhaps, but one that is easy to lose sight of when the integration work feels abstract.

3. KNOWING SERVICE GRAPH CONNECTOR

3.1 Definition and Purpose of Service Graph Connector

The Service Graph Connector is, at its core, an automated data pipeline between external IT systems and the ServiceNow CMDB. What distinguishes it from simpler integration approaches is that it handles not just the movement of data but the validation and transformation of that data according to CSDM standards. Earlier approaches moved data from point A to point B and left quality enforcement as a separate problem. The SGC treats quality enforcement as part of the pipeline itself.

The practical effect is a meaningful shift in where human effort goes. Instead of spending time on data entry and reconciliation, IT teams spend time on oversight and governance. That is a better use of skilled people, and in our observation it also tends to produce better outcomes because the people doing the oversight have context that a data entry process never has.

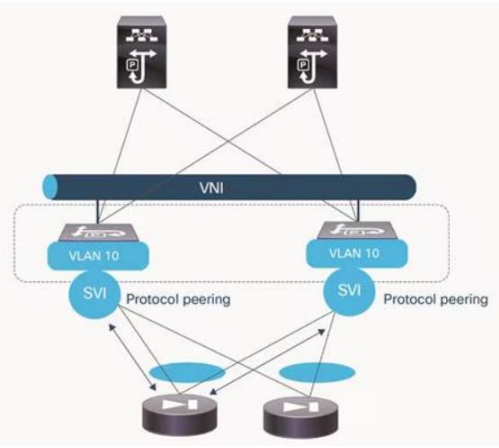

Figure - 2: Service Graph Architecture Diagram

3.2 Key Features and Capabilities of Service Graph Connector

Several things set the SGC apart from simpler connectors. Data validation is the most important. Before any record reaches the CMDB, the connector checks it against CSDM-defined rules and either flags it or rejects it if it fails to conform. This sounds straightforward but it represents a significant departure from how most organizations have historically handled CMDB population, where bad data would make it into the system and cause problems that were only discovered much later (Karimova, 2024).

Automated CI discovery and relationship mapping are also worth calling out specifically. When a new virtual machine spins up in a cloud environment, the SGC detects it, creates the appropriate CMDB record, and maps its relationships to other CIs automatically. The CMDB stays structurally accurate even as the infrastructure shifts. The connector also supports bidirectional data exchange, meaning that corrections made within ServiceNow can flow back to source systems. That closes a loop that many organizations leave open, and it reduces the risk of the same bad data being reingested on the next cycle.

3.3 How SGC Improves Data Quality in CMDB

We think about SGC's data quality contribution in two distinct ways. First, there is the ongoing accuracy benefit. Continuous ingestion means the CMDB reflects the current state of the environment rather than whatever it looked like during the last manual update cycle. For organizations running dynamic cloud environments, the difference between those two things can be enormous. Decisions made on stale data are almost always worse than decisions made on current data.

Second, there is the structural quality benefit. The automated validation catches problems that manual processes typically miss: mislinked CIs, orphaned records, duplicate entries, field formatting inconsistencies. Over time, as the SGC runs repeatedly against the same environment, it tends to surface and correct pre-existing quality issues in addition to preventing new ones. We have seen CMDBs that were in poor shape improve substantially within a few months of SGC deployment, simply from the combination of continuous ingestion and consistent validation.

3.4 Comparison with Traditional Data Ingestion Methods

The core weakness of every traditional data ingestion approach was that it was episodic. The CMDB was accurate right after an update and grew less accurate until the next one. In a stable environment that lag might be tolerable. In a modern cloud environment it is not. The gap between the CMDB and reality could grow large within days.

The SGC addresses this by making ingestion continuous rather than periodic. Validation is built into the pipeline from the start rather than being applied as an afterthought. And since the connectors are maintained by ServiceNow rather than by the customer's internal team, the burden of keeping integrations functional as upstream systems change falls on the vendor. That last point matters more than it might seem. Custom connector maintenance is one of those ongoing costs that organizations frequently underestimate when they build their own solutions (Shukla et al., 2023).

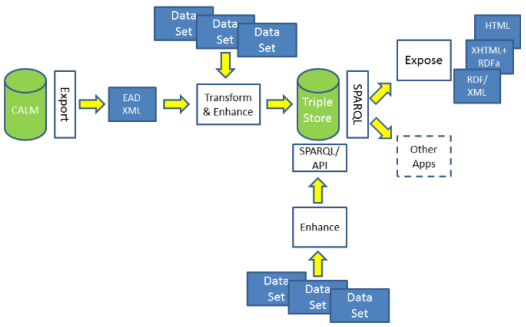

Figure - 3: Service Graph Data Transformation Process

4. ARCHITECTURE AND WORKING OF SERVICE GRAPH CONNECTOR

4.1 Technical Components of Service Graph Connector

The SGC's architecture has three distinct layers and understanding all three matters for implementation planning. The base layer consists of integration adapters, which are pre-built connectors and APIs that handle authentication and initial communication with external platforms. These adapters vary in complexity. A cloud platform adapter like AWS or Azure is considerably more involved than an adapter for a simple on-premises monitoring tool, and that complexity has implications for configuration time.

The middle layer is the Service Graph Framework itself. This is where data models, classification rules, and relationship schemas live. Everything that comes through the base layer gets processed against these standards before it can reach the CMDB. At the top, the SGC connects with ServiceNow's Discovery and Event Management modules, which enables real-time CI detection and relationship resolution. The three layers work together. Understanding how they interact is essential for diagnosing issues when things go wrong (Chinthapatla, 2024).

4.2 Data Ingestion Pipeline and Transformation Process

Data moves through the SGC in three stages and each stage is worth understanding individually. Extraction is first. The connector queries an external system via an API call, a webhook, or a scheduled pull, and retrieves raw data about servers, applications, devices, or services. That raw data usually reflects the source system's own data model, which often looks quite different from the CMDB's.

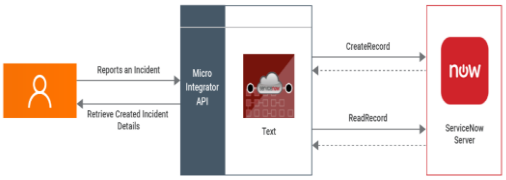

Figure - 4: API Integration with ServiceNow

Transformation is where the heavy lifting happens. The connector maps incoming fields to their CMDB equivalents, normalizes values to match ServiceNow's standards, and enriches records by appending relationship data. That last step, inferring or verifying that a discovered server belongs to a particular business service, is genuinely difficult to get right and is often where implementation teams spend the most time. Loading is the final step: the transformed records are written to the CMDB, updating existing CIs or creating new ones as needed. The whole cycle runs automatically on a defined schedule (Patel et al., 2019).

4.3 External Integration

One of the SGC's practical strengths is its breadth of integration coverage. Cloud platforms like AWS and Microsoft Azure are natively supported. Virtual machines, storage volumes, container clusters, networking components: all of it can be automatically imported into the CMDB without manual discovery cycles. For organizations running significant cloud workloads, this alone justifies the deployment effort.

On-premises systems work differently. Here the connector communicates with monitoring or management platforms installed within the organization's own infrastructure. The SGC supports both push and pull patterns. In a pull configuration, the SGC queries external systems periodically for updated CI data. In a push configuration, external systems send change events to ServiceNow as they occur. Which pattern makes sense depends on the source system's capabilities and how much latency the organization can tolerate. Both approaches work. Neither is universally better.

4.4 API and Data Validation Mechanisms

The API layer handles two-way traffic. In bidirectional configurations, corrections made in ServiceNow can flow back to source systems, which helps prevent the same erroneous data from being reingested. This turns what would otherwise be a one-way pipeline into something closer to a shared, synchronized record between platforms.

Validation happens at ingestion time, not after. This is the early-gate design we mentioned earlier, and it is important. Catching problems before they reach the CMDB is categorically better than catching them afterwards. The rules cover both structural conformances, meaning correct data types, required fields, and valid values, and semantic consistency, meaning that the declared relationships between CIs are logically coherent. When validation fails, the SGC logs the failure and generates alerts. In our experience, teams that review those logs regularly catch integration issues much faster than teams that do not.

5. SERVICE GRAPH CONNECTOR IMPLEMENTATION AND BENEFITS

5.1 Prerequisites and Setup Process

Getting the prerequisites right is where many implementations go wrong. The basics are straightforward enough: a licensed ServiceNow instance with the Service Graph Framework enabled, a CMDB that is structured around CSDM principles. But that second requirement deserves more attention than it typically gets. Deploying the SGC on top of a poorly organized CMDB does not fix the underlying data quality issues. It usually makes them worse by ingesting new data at scale into a system that was already confused (Ale, 2024).

The setup process itself involves establishing secure connections to external systems, which means configuring API credentials, setting up encryption for data in transit, and testing connectivity before any live ingestion begins. Once connectivity is confirmed, administrators define the data mappings and transformation rules. This is the most time-consuming part of any deployment. How long it takes depends almost entirely on how well-defined the source systems' data models are and how closely they align with the CSDM. After configuration and testing, ingestion is enabled and the connector begins its work.

5.2 Use Cases and Real-World Applications

Cloud infrastructure management is the most common use case, and it is easy to see why. Organizations running workloads on AWS or Azure get automatic CMDB records for virtual machines, storage resources, and network components without any manual discovery cycles. That visibility has downstream benefits for capacity planning, cost attribution, and failure impact analysis that most organizations underestimate until they experience a major incident where the CMDB actually helped.

Network operations teams have applied the SGC to similar effect by integrating data from monitoring platforms like SolarWinds or Nagios. When network devices are catalogued automatically and their dependencies mapped, incident response gets faster in a very practical way: engineers can see immediately which upstream device is affecting downstream services and prioritize accordingly. In healthcare and financial services, the SGC has also been deployed to bridge ERP systems and service catalogs, mainly to support asset lifecycle tracking and the audit documentation that compliance frameworks require. These are less glamorous use cases than cloud infrastructure, but they represent significant operational value.

5.3 Benefits in Terms of Data Accuracy

The accuracy improvement delivered by the SGC comes from closing the gap between what the CMDB says and what is actually running. Traditional approaches left that gap open for days or weeks at a time. The SGC closes it continuously. That difference is more significant than it sounds when you consider that most incident response processes depend on CMDB data being correct at the moment an incident occurs, not at the moment of the last scheduled update.

At enterprise scale, this matters enormously. When IT environments span thousands of CIs across dozens of platforms, the accumulated inaccuracy of a batch-update approach becomes substantial. Procurement decisions get made on wrong data. Capacity planning uses figures that do not reflect current consumption. Audit reports describe an environment that no longer exists. Continuous ingestion with built-in validation addresses all of these problems at once, which is part of why the return on a well-executed SGC deployment tends to be higher than organizations initially project.

5.4 Benefits of Automation

The automation benefit is the most immediately tangible. Tasks that used to consume significant IT staff time, importing records from a newly connected cloud account, reconciling data after a hardware refresh, verifying relationship mappings following an infrastructure change, are handled by the connector automatically. In large

organizations, CMDB maintenance has historically absorbed a surprising amount of engineering time that could have been spent on actual service improvement.

There is a less obvious benefit too. Because validation runs with every ingestion cycle, the CMDB does not gradually accumulate the errors and inconsistencies that manual processes inevitably introduce. The quality stays consistent over time rather than degrading between manual cleanup efforts. That consistency, in turn, makes teams more willing to trust the CMDB as a source of truth, which creates a positive feedback loop. A trusted CMDB gets used more. A more-used CMDB reveals its errors faster. Faster error correction produces a more accurate CMDB.

5.5 Decision-Making and IT Governance Benefits

From a governance standpoint, what the SGC really provides is a data foundation that rigorous IT management actually requires but rarely has. Relationship-aware CI data makes impact analysis possible in a meaningful way. When someone proposes a change to a shared network component, you can look at the CMDB and understand immediately which services depend on it and which teams need to be consulted. That kind of visibility supports better change management decisions in a way that is hard to quantify but very easy to feel the absence of (Barczak & Barczak, 2023).

For compliance, the continuous automated ingestion means the CMDB describes the current environment rather than a historical snapshot. Demonstrating configuration control during an audit becomes straightforward rather than a scramble. Undocumented changes, one of the most common audit findings in organizations that rely on manual CMDB maintenance, become much harder to sustain when the system is being updated continuously from authoritative sources.

6. CHALLENGES, FUTURE TRENDS, AND RECOMMENDATIONS

6.1 Common Implementation Challenges

We want to be direct about the challenges here because vendor documentation tends to gloss over them. Integration complexity is real. External systems vary widely in their data structures, API protocols, and update frequencies. Adapting the connector to each source requires careful, time-consuming configuration work. Organizations with a large portfolio of legacy tools often find that building and validating data mappings for all of them takes two or three times longer than the initial estimate.

Data model alignment is a related challenge that deserves its own discussion. The CSDM is a robust framework, but mapping an external system's data model onto it requires thorough familiarity with both sides of the equation. Where existing CMDB data has not been consistently classified, this effort must be preceded by a data cleansing exercise. That exercise is unglamorous work and it is tempting to skip or rush it. In our observation, organizations that skip it consistently run into problems six to twelve months into the deployment that trace directly back to the unresolved data quality issues they chose not to address at the start.

6.2 Limitations and Security Concerns

The SGC's reliance on external APIs creates a structural vulnerability. When an upstream system changes its API, whether through a version upgrade, an authentication protocol change, or an endpoint deprecation, the integration breaks until the connector configuration is updated. This is not a hypothetical risk. It happens regularly in practice. Organizations need monitoring in place to detect integration failures quickly and a clear process for addressing them before significant CMDB staleness accumulates.

Security deserves serious attention too. The connector transmits potentially sensitive configuration data across system boundaries, and the protection requirements are not trivial. Encryption in transit is the minimum. Strong authentication, OAuth 2.0 or SAML-based SSO, is standard practice. Granular access controls on the ServiceNow side limit the blast radius if credentials are compromised. Organizations in regulated industries need to go further, with regular security reviews of both the connector configuration and the upstream APIs it connects to. We have seen organizations treat the security review as a one-time deployment checklist item rather than an ongoing practice. That is a mistake.

6.3 Performance Considerations

Performance becomes a real concern at scale. In large IT environments, high-frequency ingestion from multiple external systems simultaneously places significant load on both the connector and the ServiceNow

instance. When the connector slows down, ingestion gets delayed and the CMDB falls temporarily out of sync. That is generally better than persistent staleness, but in fast-moving operational contexts it can still cause problems.

The standard optimization approach involves tuning batch sizes and ingestion intervals to balance freshness against system load. For very high-volume environments, partitioning data flows across separate connector instances can help. ServiceNow's native load-balancing capabilities are also relevant here. Regular performance reviews are not optional if you care about the quality of what the SGC produces. Assessing ingestion throughput, validation failure rates, and API response times on a consistent schedule lets you catch bottlenecks before they affect operations rather than after (Kario, 2018).

6.4 Future Trends and Improvements

The most interesting development on the horizon is the integration of AI into the SGC's validation and anomaly detection capabilities. Current validation is rules-based: data either conforms to the schema or it does not. AI-augmented validation could go further, identifying records that are technically conformant but statistically unusual given the historical pattern of the environment. A sudden cluster of new CIs that looks different from the usual pattern might warrant human review even if every individual record passes the structural checks.

Predictive analytics is the other frontier worth watching. If you correlate CMDB state with historical incident data over a long enough time horizon, patterns start to emerge. Certain configuration states tend to precede certain failure types. That kind of insight could shift IT operations from reactive to genuinely proactive, which would be a significant change. We are also watching the expansion of integration coverage for IoT devices and edge computing infrastructure, which currently lack the standardized connector support that cloud platforms have but which are becoming increasingly prevalent in enterprise environments.

6.5 Recommendations for Successful Adoption

Start with an honest assessment of your CMDB's current state. If the existing data is poorly structured or inconsistently classified, fix that first. We know that is not what most organizations want to hear because it adds time to the project. But attempting to deploy the SGC on top of a degraded data foundation consistently produces worse outcomes than taking the time to clean things up beforehand. The SGC will amplify whatever it finds in the CMDB, both the good and the bad.

Invest in training, and not just on how to operate the connector day to day. The teams responsible for the SGC need to understand the Service Graph Framework and the CSDM well enough to make judgment calls about validation failures, integration errors, and whether ingested data is actually meeting quality thresholds. Those judgment calls come up regularly. Cross-functional alignment between IT operations, security, and data governance is also essential, since the SGC touches all three domains and decisions made in one area frequently affect the others.

After deployment, build a monitoring practice around the connector. Track ingestion success rates, validation failure trends, and CMDB staleness indicators as ongoing metrics rather than one-off checks. Stay current with ServiceNow's connector releases and CSDM updates. An SGC deployment that is not actively maintained will degrade over time as the environment around it changes. The organizations that get the most long-term value from the tool are the ones that treat it as a living operational capability rather than a project that ends at go-live (Wood, 2016).

CONCLUSION

CMDB accuracy has been a persistent, frustrating problem in IT operations for as long as CMDBs have existed. Manual processes cannot keep up with the pace of change in modern IT environments, and most organizations have known that for years without having a good alternative. The Service Graph Connector is a genuine step forward. By making data ingestion continuous and building quality enforcement into the pipeline itself, it addresses the root cause of CMDB drift rather than just treating the symptoms.

That said, it is not a magic solution. We have tried to be clear throughout this paper about what the implementation actually requires, because the gap between a well-deployed SGC and a poorly-deployed one is large. Organizations that invest in proper preparation, train their teams adequately, and commit to ongoing monitoring will get dramatically better results than those that treat it as a plug-and-play deployment.

Looking ahead, the convergence of the SGC with AI-driven analytics and predictive modeling is genuinely exciting. A CMDB that not only stays current but anticipates where problems are likely to emerge would represent a qualitative change in what IT operations teams can do. We are not there yet, but the trajectory is clear. The Service Graph Connector, properly implemented and continuously maintained, is the foundation that makes that future possible.

REFERENCES:

1. Ale, N. K. (2024). Enhancing quality management with integrated metrics dashboards: A case study on QMO metrics dashboard. European Journal of Advances in Engineering and Technology, 11(1), 64-69. https://www.researchgate.net/publication/383034302
2. Barczak, A., & Barczak, M. (2023). Selected issues of threat management in cyberspace. Studia Informatica: System and Information Technology, 28(1), 5-28. https://czasopisma.uws.edu.pl/studiainformatica/article/view/3531
3. Chinthapatla, Y. (2024). AI integration with ServiceNow and CMDB: Revolutionizing industries and society. Journal of Science & Technology, 5(3), 1-13. https://thesciencebrigade.com/jst/article/view/159
4. Dalfo Ferrer, D. (2020). ServiceNow data extraction system (Bachelor's thesis, Universitat Politecnica de Catalunya). https://upcommons.upc.edu/handle/2117/328799
5. Karimova, N. (2024). Navigating strategic risks: Overcoming challenges in the competitive cloud computing sector. Preprints. https://www.preprints.org/manuscript/347d22092f6e24d5fc050e89111fa014
6. Kario, P. (2018). Service design approach to understanding user experiences on the ServiceNow cloud solution. Theseus. https://www.theseus.fi/handle/10024/155888
7. Naik, N. N., & Bhosale, T. S. (2020). Exam analysis system using ServiceNow based on cloud computing. IRJET, 7(9). https://www.academia.edu/download/64756970/IRJET_V7I9392.pdf
8. Patel, M., Patil, S., Tzanavara, K., Chauhan, M., Wang, Y., Xie, H., & Shi, L. (2019). ServiceNow: CMDB research. Clark University. https://commons.clarku.edu/sps_masters_papers/50/
9. Shukla, A., Patel, J., Panzade, K., & Sardana, H. (2023). Cisco cloud infrastructure. Cisco Press.
10. Wood, M. (2016). Mastering ServiceNow. Packt Publishing.