Cloud Chemistry: Making SaaS solutions play nice with your application to ensure Enterprise IT Quality

Nikita Deepak Davda
nikki.d.davda@hotmail.com

Ying Ki Kwong
ying.ki.kwong@pnsqc.org

Abstract

The global Software as a Service (SaaS) market size is projected to grow from $273.55 billion in 2023 to $908.21 billion by 2030 [1]. SaaS has revolutionized the way businesses operate by providing lower upfront costs, scalability and on-demand access to software applications. However, test strategies must adapt with a focus on cloud-based models and ensuring Enterprise level IT quality is still maintained. In this paper, we will discuss SaaS integrations and how to overcome common challenges. This paper will analyze enterprise SaaS integration from the perspective of the four pillars of Enterprise level IT quality [2]:

- Data Quality – How to ensure data synchronization between your application and the SaaS solution.
- Application Quality – How to ensure Functionality, Usability and Security
- Infrastructure Quality – How to ensure Reliability, Scalability and Security
- Enterprise Quality Management Systems

We will discuss lessons learned from various incidents and their implications to overall quality of systems that utilize third-party SaaS services. The lessons learned include keeping up with rapid development cycles of SaaS companies, third-party integration, security concerns, and making sure applications are running as expected to meet business needs during operational support & maintenance.

Biography

Nikita Deepak Davda is a Lead Test Software Engineer with over a decade of experience in designing, implementing and managing test plans to build high quality software. She believes that quality plays an important role in every stage of SDLC, right from when the Requirements are written all the way until Post deployment. She has filed a patent with USPTO with a solution for “Uninterrupted Usage and Access of Physically unreachable managed handheld devices.” She is passionate about testing applications on the Cloud and enjoys the challenge of testing integration between applications.

Ying Ki Kwong is an independent consultant. In the public sector, his roles with the state of Oregon included: E-Government Program Manager, Statewide QA Program Manager, IT Investment Oversight Coordinator, and Project Office Manager of the Medicaid Management Information System. In the private sector, his roles included: CEO of a Hong Kong-based internet B2B portal for trading commodities futures and metals, program manager in the Video & Networking Division of Tektronix responsible for worldwide applications & channels marketing in the video server business, and research engineer in Tektronix Labs. In these roles, Dr. Kwong managed software-based business operations, systems, products, and business process improvements. He received the doctorate in applied physics from Cornell University and was adjunct faculty in the School of Business Administration at Portland State University. He holds certifications in project management (PMP), ITIL, and IT Service Management. He has served on the Board of PNSQC since 2021.

1. Introduction

Software as a Service (SaaS) is a way of delivering enterprise applications remotely over the internet instead of local installations. Before SaaS, software was typically purchased or licensed by an enterprise customer; often installed on its own servers and client devices. In SaaS models, software or online services are rendered as a set of capabilities that an enterprise customer subscribes to, with pricing options at different service levels for different numbers of authorized users. Many SaaS vendors offer training to help enterprise customers to make the most of their investments. Overall, SaaS simplifies software acquisition for enterprise customers, providing cost-effective and convenient ways for an enterprise to access powerful applications without the complexities in traditional software deployment.

From a financial accounting standpoint, traditional software acquisition models are typically treated as capital investments, while SaaS models are typically treated as expenses. SaaS models are often preferred by the executive management of an enterprise – because initial investment is lower, even if total cost of ownership may end up being higher over time.

2. How does SaaS work?

When an enterprise customer subscribes to SaaS software, its users typically log in through a web browser. The software runs on the service provider's servers, which handle all the processing and data storage. This allows the users to access the software needed from any authorized device with an internet connection and appropriate information security safeguards – providing flexibility and mobility. The SaaS vendor manages software maintenance, including updates, security, and backups, so enterprise customers have the latest features and security patches. From a technical perspective, SaaS vendors may use multi-tenant architecture, meaning a single instance of the software serves multiple customers. This approach optimizes resources and reduces costs, as infrastructure and maintenance expenses are shared across many customers. SaaS applications are often architected to be highly scalable, allowing enterprise customers to easily adjust their subscription levels based on their business needs, whether that means adding features, increasing storage, supporting more users, or other quality of service criteria.[3][4]

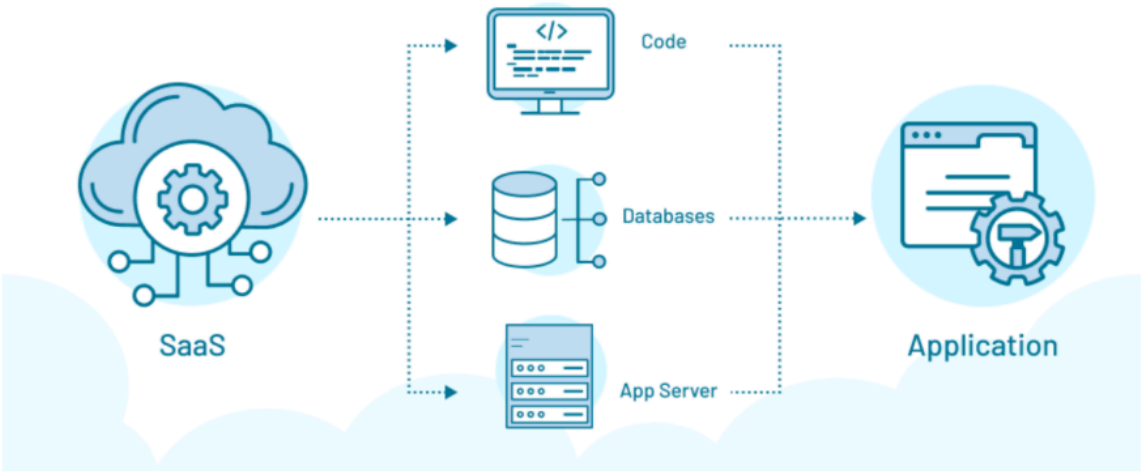

Figure 1. *SaaS Architecture.*

In this paper, we use the editorial “you” to denote an enterprise customer of SaaS vendors and “we” to denote the authors of this paper.

3. Types of Application integrations

To highlight how SaaS differs from other integration patterns, we will group application integrations into three broad types, with Type 3 being SaaS in nature. This classification is conceptual and not based on a specific industry standard or generally accepted definitions..

Type 1: Integration with a privately hosted enterprise system or private cloud

Here, each enterprise customer operates its own fully dedicated application stack — including software, database, and infrastructure — either on-premises or in a private cloud. For example, a bank or a government organization may run Workday HR in an isolated environment, separate from other Workday customers. This model gives organizations maximum control: independent scheduling of software upgrades and patches, flexibility in system configuration and integration with legacy systems, customer-specific performance tuning (such as elastic scaling within a dedicated pool of servers), and stronger compliance and audit capabilities.

Analogy: This is like owning or leasing your own house or building — you have maximum control over how it is managed, used, maintained, and remodeled.

Type 2: Integration with API-based third-party services

An enterprise application connects to an external service in the cloud through standard APIs. The enterprise customer subscribes to the service and accesses its data or functionality programmatically. Examples include using a payment gateway like Stripe or a mapping service like Google Maps.

Analogy: This is like subscribing to a utility service such as electricity or water. You don’t own the infrastructure — you just pay for access and use the service when you need it.

Type 3: Integration with Software-as-a-Service (SaaS)

In this model, the enterprise customer integrates with a solution hosted by a SaaS vendor. SaaS may be deployed in either a single-tenant or multi-tenant form.

In a single-tenant SaaS model, each enterprise operates in its own logically separated tenant, with isolated application and data environments but shares infrastructure, patching, and upgrade schedules as managed by the SaaS vendor. In a multi-tenant SaaS model, many customers share the same application and infrastructure, with logical data separation. Common SaaS like Salesforce can be subscribed in single-tenant or multi-tenant models. Single-tenant SaaS offers stronger data segregation and may be chosen for easier compliance with standards such as HIPAA in healthcare or FERPA in education. Multi-tenant SaaS, by contrast, is usually more cost-effective and so more widely adopted.

Analogy: A multi-tenant SaaS model is like renting in an apartment building where many residents share the same structure and common utilities, a swimming pool, and other

facilities; a single-tenant SaaS model is more like having your own townhouse in a managed community — you share services like roads and landscaping, but your unit is separate, secure and is used only by you.

4 What are the advantages & disadvantages of SaaS?

Because SaaS facilitates remote application hosting and delivery, its key advantage is easy access across locations and devices. Although not widely talked about, there are also disadvantages in using SaaS models.

Feature	Advantage	Disadvantage
Pay only for what you use	Automatically scales up or down based on usage.	Subscription payments can add up to be expensive over time.
No need for local installation	SaaS apps have the option to be run on the browser thereby installation is no longer mandatory.	You cannot control versions you want to use. You can only use the latest available version.
Lower initial costs, Rapid development	You generally don't have to pay for licenses, hardware and infrastructure.	You must pay for data migration and consultation for setup.
Easy Onboarding	SaaS vendors offer documentation and training.	You'll be tied to the platform they provide, at least for some time. It could be very time-consuming and costly to switch down the road.
Frictionless upgrades and updates	Enterprise customers have access to latest software releases, with more features and enhanced security, frequently and fast.	You cannot control versions you want to use and there might be regression bugs that get introduced.
Increased customization	Developers often design SaaS applications to be customizable. So, you can tailor them to your needs. SaaS vendors often offer APIs for integrations as add-ons to enterprise customers.	Developers can only pick from the customization available.
Advanced security	Most SaaS vendors invest heavily in privacy and security.	It is your responsibility as a client to verify that the SaaS vendor has all the necessary certifications in place and that partnering with them will not put your data under risk.
Built-in redundancy	With cloud-based SaaS, backups are frequent and automatic.	N/A

Offline functionality	Many solutions have an offline mode that works regardless of internet access. When connectivity returns, data automatically updates.	Data is stored on the software provider's servers, and as a customer, you must trust that the vendor maintains confidentiality, integrity, and security of the digital information.
Actionable intelligence	Some SaaS solutions ingest data in real time and from many sources. The result is more accurate analytics and reporting.	N/A

Table 1. *Advantages and disadvantages of SaaS services in mission critical systems.*

5 Software Engineering Quality Considerations

Integrating with SaaS software provides many advantages as seen above. However, with great power comes great responsibility. There are many quality considerations that need to be taken into account, to avoid pitfalls while integrating with these solutions. Let's analyze common pitfalls using our four pillars of Enterprise IT Quality by reviewing some sample incident scenarios.

5.1 Data Quality pitfalls during integration

During onboarding and data migration, it is important to ensure no data lost between the systems.

- *Sample Incident Scenario*: When we integrated our product with a SaaS solution, we designed the system to transfer data without errors. However, we saw data quality pitfalls like data sync mismatches, delayed updates and data transformation errors.
    - *Lesson Learned*: When two systems have to work as one solution, it is important to maintain near real-time updates between them.
    - *Recommended Practices:* Automated Integration testing between the systems is a great way to ensure seamless data flow and thorough integration. Change Data Capture (CDC) streamlines data integration processes by identifying and extracting only the changes since the last update, eliminating the need for bulk data transfers.

5.2 Application Quality during integration

When SaaS vendors update their system, one of two possible types of regression defects may occur. The SaaS solution may contain defects, or updates may break integration with an enterprise customer's systems.

- *Sample Incident Scenario*: When SaaS vendors roll out a new update, there is often a chance of regression defects being introduced. In some cases, existing functionality may be removed or modified. The customer's application quality takes a hit when they don't catch it in time and the existing functionality their customers love, no longer exists or is broken.

- *Lesson Learned*: API service contracts have to be defined and maintained. Contract testing can easily catch contract breaking changes.
- *Recommended Practices:* Automate Continuous Regression testing to ensure that existing functionalities remain unaffected.

5.3 Infrastructure Quality during integration

Some SaaS solutions provide both “out-of-the-box” functionality and third-party vendor integrations to choose from. For example, eCommerce SaaS vendors usually provide third party integrations like tax calculator APIs as a part of their contract.

- *Sample Incident Scenario*: While integrating with one such eCommerce solution, the enterprise customer chose to use the tax calculator API that the SaaS vendor integrated with. Right before a critical launch, the terms and conditions (T&Cs) of third-party tax calculator API chosen in the integration changed. Since the enterprise customer did not have enough time before launch to switch to a new tax calculator solution, there were additional costs to maintain the integration.
  - *Lesson Learned*: SaaS contracts should include any third-party integration T&Cs. SaaS vendors must disclose their contract terms with the third-party vendor with details like costs and when their contract ends.
  - *Recommended Practices:* Have a fallback plan for when a third party service becomes unavailable, to ensure system performance and to avoid any downtime.

5.4 Enterprise Quality Management during integration

At an enterprise level, SaaS integrations should take into account all the metrics of quality. In particular, Performance testing and Load testing. Customers must define Service Level Agreements (SLAs) with SaaS vendors about Application performance metrics to match the uptime of their product. If this is not specified clearly in the contract, the SaaS vendor becomes a bottleneck in the customer’s application performance.

- *Sample Incident Scenario*: The peak traffic days vary by industry. For an eCommerce domain in the United States, the “Cyber 5” weekend is a peak traffic period when the maximum number of customers shop online. SaaS solutions supporting these customers may not operate in the same domain, but have to be prepared to support the load their customers could face.
  - *Lesson Learned*: SLAs and preparation for the peak traffic days can play a critical role. Advance Performance and Load testing on both the systems is necessary.
  - *Recommended Practices:* Once the SaaS integration is complete, during a maintenance window, SaaS vendors will often switch to on-demand support. However, during peak traffic days, the support contract must be revised so the providers are available for support around the clock to mitigate any customer blocking integration issues.

6 Operations & Maintenance Quality Considerations

As described in earlier sections, SaaS adoption has the potential to offer benefits in scalability, availability, and cost-effectiveness. However, these advantages are accompanied by new responsibilities for operations, support, and maintenance teams. These teams must assure and sustain consistent IT service quality over time despite diminished control over the complete stack of services. SaaS-based systems or system components must be continuously monitored and managed across organizational and technical system boundaries. This includes not only internal systems and infrastructure, but also external vendor services that may update, scale, or fail independently of an enterprise's internal systems and their controls.

In this section, we examine how the four pillars of Enterprise IT Quality — data, application, infrastructure, and management systems — manifest in operational settings. We will use real-world scenarios experienced by the authors, as summarized in Table 2 and discussed in greater details below, to explore implications for monitoring, incident response, and continuous improvement that are integral to today's IT service management (ITSM). These scenarios are illustrative and do not span the universe of possible incidents in production environments.

Enterprise IT Quality Pillar	Sample Incident Scenario	Lessons Learned
Data Quality	SLA compliance gaps with insufficient monitoring	Internal & external monitoring required to verify SLA compliance
Application Quality	SaaS performance degradation under load	Monitor performance trends and test under realistic usage conditions
Infrastructure Quality	DR failure from WAN infrastructure without sufficient redundancy	Ensure telecom redundancy with physically separate carriers
	SaaS outage misdiagnosed due to failure of internal DNS or other systems	Monitor both SaaS and internal services for accurate RCA
	Vanity domain names and TLS certificate not well maintained	Establish governance and automate tracking of TLS certificates and vanity domain names
Enterprise Quality Management Systems	SaaS update and related prohibitions during enterprise blackout windows	Enforce change control procedures and related governance – especially compliance with blackout dates, as well as related SLAs and T&Cs (terms & conditions) in the contract

Table 2: *Sample Incident Scenarios and Lessons Learned in Operations & Maintenance, through the lens of the four pillars of Enterprise IT Quality in Reference 1.*

6.1 Data Quality in Operations & Maintenance

Operational risks to data quality include sync mismatches, stale data, and monitoring gaps that delay detection of failed integrations. In a multi-vendor SaaS environment, the enterprise often lacks direct visibility into the data handling practices of external systems or services.

- *Sample Incident Scenario*: SLA defines Severity 1 Outage to be sustained outage of an application or a website for 4 minutes or longer; with attendant notification and remedy requirements. However, internal monitoring was set up to ping at a frequency of once every 10 minutes at 3 locations outside enterprise firewalls only. SLA compliance requires improved monitoring of SaaS services in order to adopt a posture of "trust but verify."
    - *Lesson Learned*: Internal monitoring in place may not be frequent enough to ascertain SLA compliance. Monitoring needs to include dependent internal services to differentiate internal outages from SaaS vendor service outages. Also, monitoring should be deployed inside and outside the enterprise firewall.
    - *Recommended Practices:* Monitor data sync success and freshness at both boundary and system-of-record levels. Cross-reference data latency and integrity against SLA requirements and contract terms. Implement internal and external monitoring probes to validate performance and availability commitments.

6.2 Application Quality in Operations & Maintenance

SaaS applications can exhibit performance issues under peak usage conditions, including instability in load balancing and degraded response times that impact user experience.

- *Sample Incident Scenario*: Performance of websites and apps provided by SaaS vendors have load time and load balancing stability issues, as observed during traffic surges that cause significant slowdowns.
    - *Lesson Learned*: The enterprise needs to proactively monitor application performance, especially under variable load conditions, to detect emerging issues.
    - *Recommended Practices*: Perform load testing aligned with real-world usage patterns. Monitor response time trends and transaction drop-offs. Use synthetic monitoring to simulate high-load scenarios and identify thresholds and bottlenecks.

6.3 Infrastructure Quality in Operations & Maintenance

SaaS vendors often manage infrastructure across multiple layers and partners. This introduces risk where dependency chains are obscured, poorly understood, or insufficiently redundant.

- *Sample Incident Scenario*: An unexpected DR (disaster recovery) failure occurred when the SaaS vendor's primary data center suffered a fiber breach, which disabled

primary and secondary connection because both relied on the same physical infrastructure.
    - *Lesson Learned*: SaaS services should be hosted in data centers with WAN/telecom circuits provided by two different providers using physically separate routes including power feeds.
    - *Recommended Practices*: Confirm physical and provider diversity in vendor DR infrastructure.
- *Sample Incident Scenario*: Unexpected SaaS service outage was traced to an internal DNS failure and not the SaaS vendor's platform.
    - *Lesson Learned*: Enterprises must monitor both SaaS availability and dependent internal services to avoid misattribution and speed up root cause analysis. Tools like New Relic or FireEye can be helpful.
    - *Recommended Practices*: Implement layered observability that includes DNS, routing, and app-level checks.
- *Sample Incident Scenario*: Information security hygiene gaps were found involving vanity domain names and TLS certificate management.
    - *Lesson Learned*: Outages or near outages can be caused by abandoned or expired domain name registrations and expired TLS certificates. Enterprises must manage these dependencies with clear governance and should consider third-party scanning and related maintenance services.
    - *Recommended Practices*: Automate renewal tracking for vanity domain names and TLS certificates. Use external scanning tools to assess information security posture for critical dependencies. Develop a cross functional team within the enterprise and with relevant contractors that meet regularly.

6.4 Enterprise Quality Management Systems (EQMS) in Operations & Maintenance

SaaS operations challenge traditional IT service management processes and workflows by introducing asynchronous changes and opaque vendor practices. Aligning operational governance across internal and external boundaries is essential.

- *Sample Incident Scenario*: An unexpected hardware outage during a scheduled master template update occurred, and the SaaS vendor neglected to check enterprise blackout dates. This disrupted operations during a prohibited change window.
    - *Lesson Learned*: Backout dates and maintenance windows must be clearly governed and communicated per ITSM best practices using approved change procedures.
    - *Recommended Practices*: Incorporate vendor change activity into internal change calendars. Require confirmation of blackout windows in service agreements. Integrate vendor roles into incident response and root cause analysis protocols. Conduct joint postmortems for major service interruptions, paying close attention to recurring issues.

7 Conclusion

In this paper, we have reviewed a number of important concepts when an enterprise uses SaaS to support its business needs. While there are advantages and disadvantages that each enterprise must balance (Section 4), we have discussed important lessons learned during system integration and ongoing support & maintenance.

Overall, SaaS is a transformative force but requires new thinking in software QA, as well as software quality & risk management during the lifecycle of an enterprise system. Enterprise IT Quality can and must adapt from test execution to integration assurance and a range of IT services management (ITSM) considerations. Crucially important are proactive application monitoring, robust administration of SaaS contracts, and coordination and related expectation setting with internal and external stakeholders. Future directions in the use of SaaS in enterprise system integration include AI-assisted integration testing, zero-trust validation models, and lifecycle considerations.

In conclusion, test strategies must adapt to support cloud-based models. Software quality & risk management during development and ongoing support & maintenance cannot be done as if everything is within the control of executive management. Software QA must now be part of overall change management, incident management, and ITSM of an enterprise for which SaaS services are an integral part of the modern enterprise.

8 Acknowledgement & Disclosure

The authors thank John Cvetko for helpful discussions during the planning, preparation, and review of this paper. ChatGPT was used to edit certain sections of this paper for self-consistency and readability.

References

1. https://www.cloudwards.net/saas-statistics/ - last retrieved on 9/1/2025.
2. For one description of Enterprise IT Quality, see https://www.pnsqc.org/docs/PNSQC_Brief_on_Enterprise_IT_Quality.pdf - last retrieved on 9/1/2025.
3. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas - last retrieved on 9/1/2025.
4. https://www.velvetech.com/blog/saas-architecture/ - last retrieved on 9/1/2025.
5. https://www.salesforce.com/saas/ - last retrieved on 9/1/2025.