vFunction

REPORT

Microservices, Monoliths, and the Battle Against $1.52 Trillion in Technical Debt

How Software Architecture Choices Impact Application Scalability, Resiliency, and Engineering Velocity

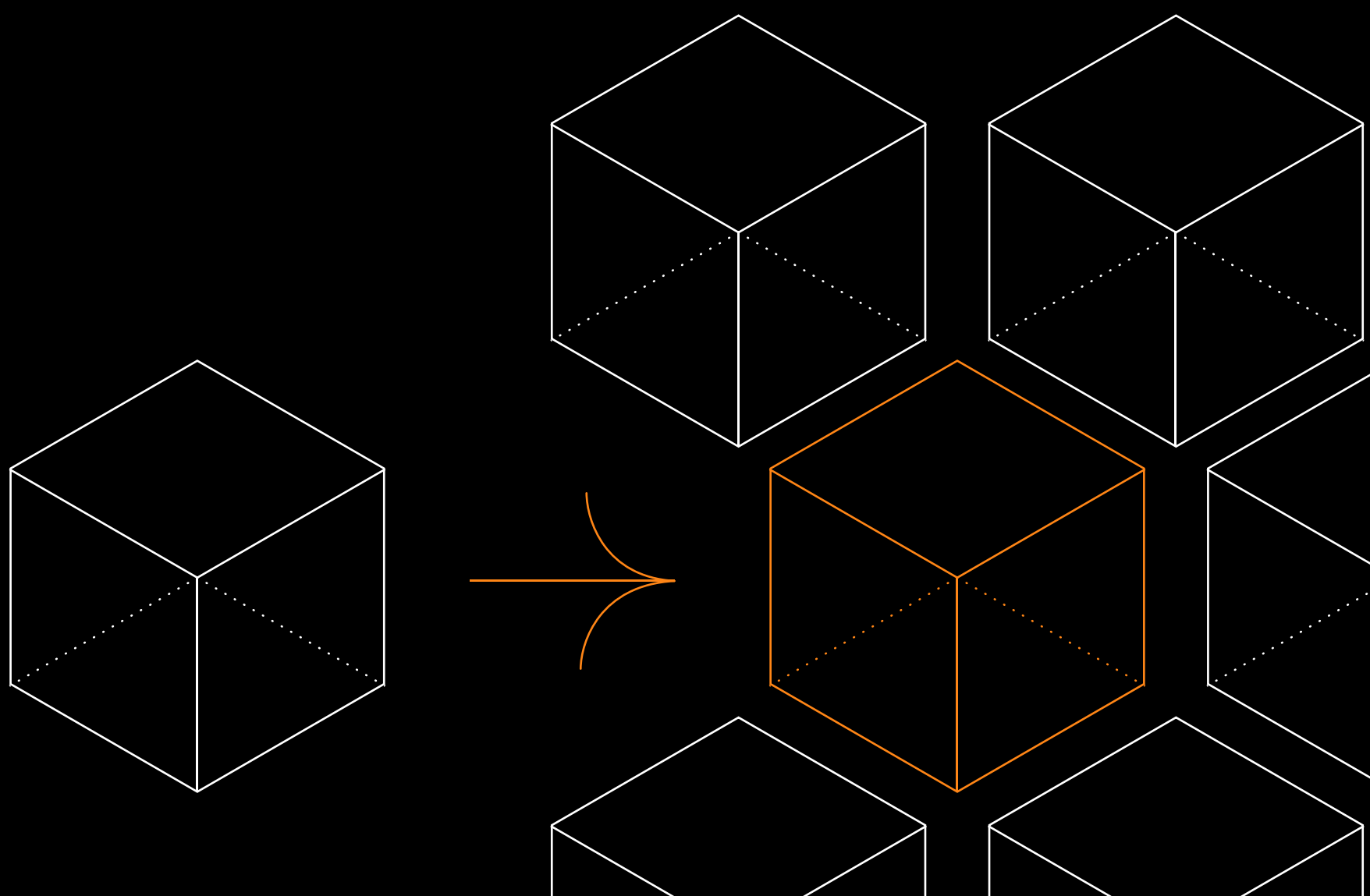
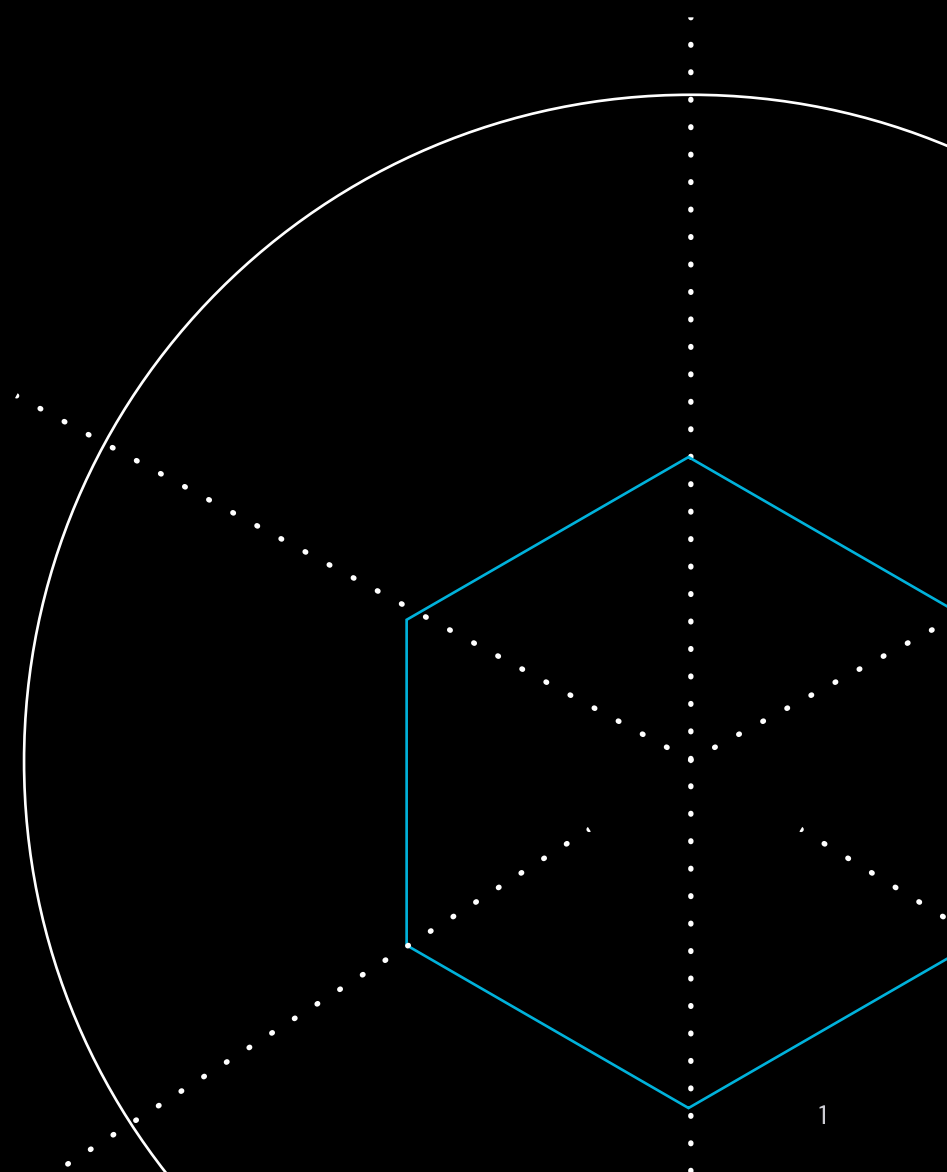


Table of Contents



Executive Summary

Rapid software innovation required to support modern business needs has resulted in increasingly complex software architectures, inefficient operational processes, and the rapid accumulation of technical debt. This debt hampers engineering velocity, limits application scalability, and impacts resiliency. It manifests as cybersecurity breaches and operational failures, resulting in a staggering $1.52 trillion drain on the U.S. economy annually.

A particularly damaging subset of tech debt is architectural technical debt, which increases systems' complexity as companies strive to be competitive by adding new capabilities to their software without sufficient architectural observability tooling.

When ranked, **architectural technical debt was identified as the most damaging type** of technical debt for applications.

Despite leveraging static code analysis, performance and security monitoring, and component mapping tools, teams still rely heavily on manual efforts and fragmented knowledge to assess architectural risk and prioritize remediation. The resulting opacity leads to application scalability and resiliency issues as well as engineering velocity challenges.

Over the last five years, software architects and engineers have grappled with:

44%

Increased complexity in monolithic applications, lack of clear domain boundaries, and decreasing modularity

42%

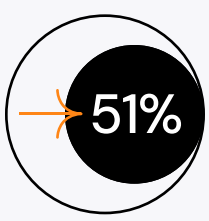
Software architecture complexity due to disparate technology stacks

39%

Lack of visibility into architecture, which makes it hard to know which microservices talk to each other

In addressing these challenges, organizations are implementing enterprise-wide initiatives to combat technical debt, allocating significant portions of their IT/engineering budgets to remediation efforts.

Remediating technical debt



More than half dedicate more than a quarter of their total annual IT/engineering budget to remediating technical debt (i.e., refactoring or re-architecting).

77%

Nearly eight in ten organizations have enterprise-wide initiatives in place to address technical debt.

However, the distribution of responsibility for addressing technical debt across multiple roles and teams highlights the complexity of the issue. A majority of organizations acknowledge multiple stakeholders responsible for tackling technical debt.

Responsibility for addressing technical debt

When asked, "Who owns or is responsible for addressing technical debt in the organization? (Select all that apply)," most organizations selected more than one role/team as being responsible:

48%	Enterprise architect or architecture team	42%	Application architect
47%	Each engineering leader	40%	Each application/product owner
47%	Central or head of all engineering teams	5%	Don't have anyone responsible for technical debt in the organization

Furthermore, the impact of architectural choices on technical debt is substantial, with organizations grappling with the trade-offs between monolithic and microservices architectures.

Those with monolithic architectures tend to incur higher remediation costs and experience more pronounced issues with engineering velocity, scalability, and resiliency compared to those with microservices.

Monolithic vs microservices

57% of organizations with monolithic architectures allocate over a quarter of their IT budget to technical debt remediation, compared to 49% for microservices architectures.

57% monolithic architecture

vs

49% microservices architecture

2x

Companies with monolithic architectures are **2.1 times** more likely to have issues with engineering velocity, scalability, and resiliency compared to those with microservices architectures.

To mitigate these challenges, organizations are exploring innovative approaches such as leveraging generative AI and shifting left by incorporating architectural observability early in the development lifecycle. These strategies aim to enhance application resiliency, reduce outage risks, and optimize performance, aligning with the evolving landscape of software development and architecture.

Mitigation

40%

said shifting left by using architectural observability is the most effective approach to ensuring application resiliency and reducing outage risks.

41%

of respondents plan to leverage generative AI to improve application performance and scalability as part of modernization efforts.

About this study

The research study, "Microservices, Monoliths, and the Battle Against $1.52 Trillion in Technical Debt," surveyed more than 1,000 U.S.-based architecture, development and engineering leaders, and practitioners at large enterprises as well as smaller digital-first companies. It revealed the importance of addressing technical debt, especially architectural technical debt, in organizations, and was conducted to:

- Explore the impact of software architecture on business outcomes, specifically engineering velocity, application resiliency, and scalability, and identify the greatest pain point for enterprises delivering and maintaining applications
- Gather insights on how software development organizations conceptualize architectural debt and manage it
- Provide insight into the role of architects and engineering leaders into reducing technical debt and delivering faster innovation

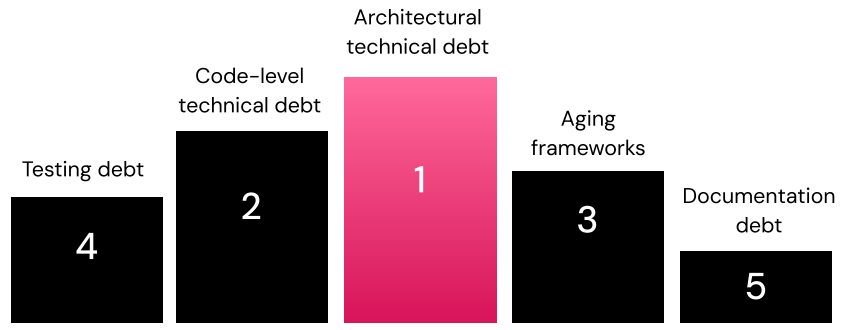
Conquering Technical Debt to Accelerate Growth and Innovation

Effective technical debt management has emerged as a critical imperative. The significant allocation of engineering resources toward technical debt remediation reflects this commitment. Half of respondents (51%) indicated their organizations dedicate more than a quarter of their total annual IT/engineering budget to remediation, including refactoring and re-architecting.

Architectural technical debt (ATD), was identified as a key priority for organizations, with nearly eight in 10 organizations (77%) having enterprise-wide initiatives in place to address technical debt head-on. When asked to rank types of technical debt from most damaging to least damaging for applications, ATD ranks as the most detrimental. This finding carries significant implications, as ATD encompasses structural deficiencies, excessive dependencies, violations of design principles, lack of modularity, and architectural complexity. Failure to effectively manage these factors could hinder organizations' ability to build resilient and scalable software capable of meeting evolving business needs.

As company size increases, ATD becomes a bigger issue than code-level debt. This underscores the importance of managing technical debt effectively as a company scales.

Respondents ranked the types of technical debt from most damaging (1) to least damaging (5) impact on their applications



#1 Architectural technical debt

Structural deficiencies, too many dependencies, violation of principles, lack of modularity, architectural complexity

#2 Code-level technical debt

Poor coding practices, code smells

#3 Aging frameworks

Outdated frameworks, vulnerabilities that haven't been addressed

#4 Testing debt

Lack of automated tests, insufficient test coverage

#5 Documentation debt

Outdated docs, lack of architectural documentation

Prioritizing technical debt remediation with architecture

The survey, with respondents split between organizations with entirely monolithic architectures, a mix of monolithic and microservices, and with entirely or predominantly microservices, revealed a correlation between an organization's software architecture and its approach to technical debt remediation. More specifically, architecture types and company size impact an organization’s prioritization of technical debt in terms of budget allocation and the types of debt considered most damaging.

Organizations that allocate over a quarter of their IT budget to technical debt remediation

57%

Organiztions with monolithic architecture

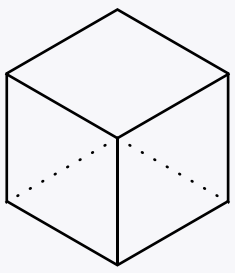
VS

49%

Organizations with microservices architecture

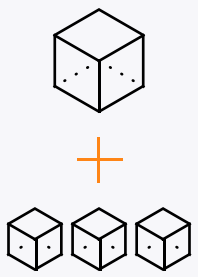
Organizations with monolithic architectures are more likely to allocate a substantial portion of their IT budget to addressing technical debt, with 57% dedicating over a quarter of their total annual budget to these efforts. In contrast, organizations with entirely or predominantly microservices architecture allocate a slightly lower percentage (49%) of their budget to technical debt remediation.

Respondent survey demographic according to their organization’s architecture



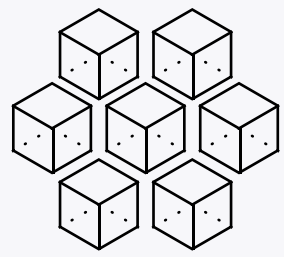
33%

Organizations with entirely monolithic architecture



31%

Organizations with a mix of monolithic and microservices

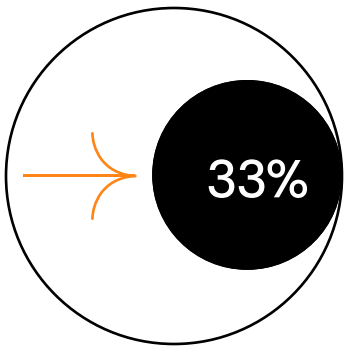


37%

Organizations with entirely or predominantly microservices

Architectural Impact on Engineering Velocity, App Scalability, and Resiliency

An organization's ability to rapidly deliver innovative products and services is paramount to maintaining a competitive edge. However, the research found that architectural choices can profoundly impact key aspects of software development and delivery, including engineering velocity, application scalability, and resiliency.



A third of respondents described their software architecture as having moderate or poor health, with notable deficiencies and technical debt accumulation

This burden of technical debt manifests in multiple structural flaws and a lack of consistency, degrading modularity and necessitating extensive refactoring efforts for even minor feature additions.

The consequences of these architectural challenges are far-reaching, preventing organizations from operating at their full business potential. Delayed projects, productivity losses, missed market opportunities, and increased costs are just a few of the negative impacts cited by respondents.

Findings highlight a stark contrast between organizations with microservices architecture and monolithic architecture. Enterprises with a microservices architecture report better engineering velocity, application scalability, and resiliency than those with a monolithic architecture. Companies with entirely monolithic architecture are more than two times (2.1) more likely to face issues with velocity, scalability, and resiliency compared to those with entirely or predominantly microservices architecture.

That said, while enterprises embracing microservices report enhanced engineering velocity, application scalability, and resiliency, these organizations also suffer from challenges including delayed platform upgrades, missed market and revenue opportunities, and decreased productivity. Both architectural paradigms have their challenges. To meet business objectives, a balanced assessment of architectural choices is crucial.

Compared to organizations with entirely or predominantly microservices architecture, companies with entirely monolithic architectures experience more challenges

1.9x more likely to experience extremely slow or slow velocity

2.5x more likely to have extremely limited or poor scalability

2x more likely to have extremely poor or low resiliency

Respondents were asked to select the top three negative business impacts due to either slow engineering velocity, inadequate application scalability, or application resiliency issues. Challenges vary based on architecture.

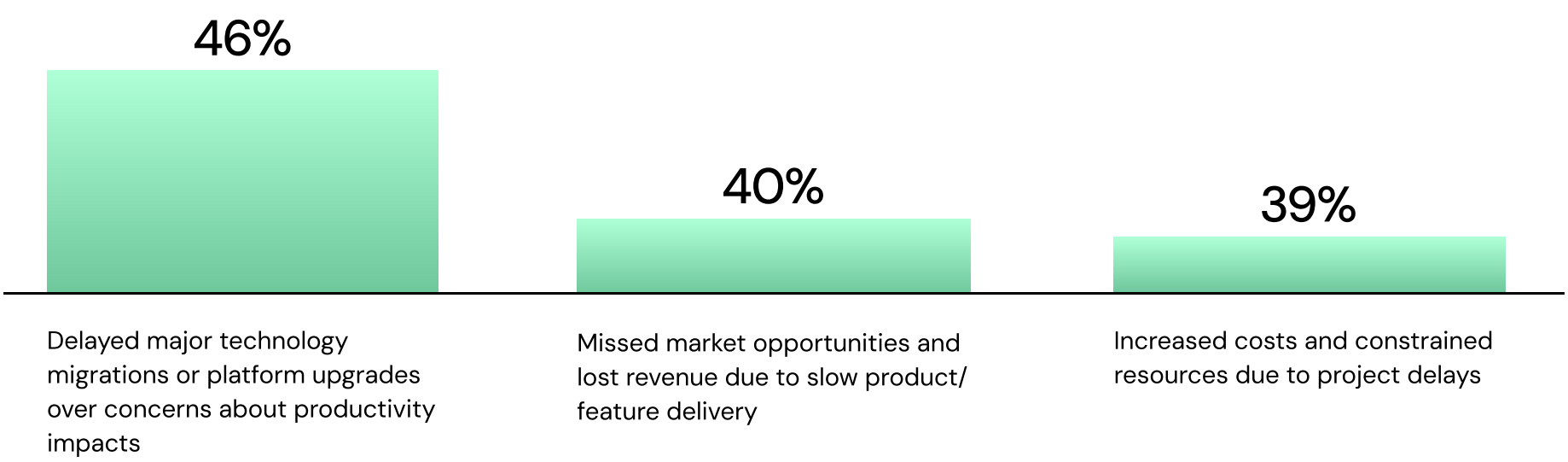
Key aspects of software development and delivery

Architecture type	Top negative business impacts experienced due to slow engineering velocity	Top negative business impacts experienced due to inadequate application scalability	Top negative business impacts experienced due to application resiliency issues
Entirely Monolithic	**44%** Missed market opportunities and lost revenue due to slow product/ feature delivery **36%** Customer churn and loss of market share due to inability to keep up with user demands **34%** Increased costs and constrained resources due to project delays	**37%** Delayed product launches or feature releases due to concerns about system capacity **36%** Increased infrastructure and operational costs to compensate for scalability issues **34%** Customer churn and loss of market share due to poor performance during peak loads	**36%** Customer churn and loss of market share to competitors with more reliable applications **34%** Increased infrastructure spend to over-provision resources as failsafe against outages **33%** Decreased productivity and increased operational costs due to time spent recovering from failures
Mix of Monolithic and Microservices	**48%** Delayed major technology migrations or platform upgrades over concerns about productivity impacts **38%** Increased costs and constrained resources due to project delays **36%** Missed market opportunities and lost revenue due to slow product/ feature delivery	**40%** Increased infrastructure and operational costs to compensate for scalability issues **38%** Delayed product launches or feature releases due to concerns about system capacity **37%** Productivity losses from engineering teams constantly firefighting scalability problems	**44%** Decreased productivity and increased operational costs due to time spent recovering from failures **33%** Increased infrastructure spend to over-provision resources as failsafe against outages **30%** Delayed product innovations or feature releases due to resources spent fixing resiliency problems
Entirely or Predominantly Microservices	**53%** Delayed major technology migrations or platform upgrades over concerns about productivity impacts **45%** Increased costs and constrained resources due to project delays **39%** Missed market opportunities and lost revenue due to slow product/ feature delivery	**39%** Missed revenue opportunities due to inability to handle increased user demand **39%** Customer churn and loss of market share due to poor performance during peak loads **38%** Increased infrastructure and operational costs to compensate for scalability issues **38%** Productivity losses from engineering teams constantly firefighting scalability problems	**40%** Decreased productivity and increased operational costs due to time spent recovering from failures **39%** Delayed product innovations or feature releases due to resources spent fixing resiliency problems **35%** Customer churn and loss of market share to competitors with more reliable applications

Engineering velocity: The pace of innovation

Rapidly delivering new features and capabilities is crucial for organizations to stay ahead of the competition. However, nearly half (48%) of organizations reported extremely slow or moderate engineering velocity, leading to delayed migrations, missed opportunities, and higher costs.

Top three negative business impacts organizations across all architectures have experienced due to slow engineering velocity



Organizations with entirely or predominantly microservices architecture report the best engineering velocity. In comparison, those with a monolithic architecture are more likely to report that they have extremely slow or slow velocity.

Architecture Type	Good/Excellent Engineering Velocity	Slow/Extremely Slow Engineering Velocity
Entirely Monolithic	48%	26%
Mix of Monolithic and Microservices	50%	18%
Entirely or Predominantly Microservices	59%	14%

While organizations with a monolithic architecture are nearly two times more likely to have slow or extremely slow velocity compared to those with a microservices architecture, the latter can also suffer from complexity challenges that slow its engineering velocity such as delayed platform upgrades, increased costs, and missed market opportunities.

Application scalability: Adapting to growth

As businesses expand and customer demand surges, scaling applications seamlessly becomes a critical differentiator. Despite this, 45% of organizations struggle with application scalability, reporting extremely limited, poor, or moderate capabilities. This leads to increased infrastructure costs, delayed product launches, and productivity losses, hindering their ability to effectively meet growing business demands. Furthermore, 40% of respondents identified software architecture scalability limitations as their organization's most pressing pain point, emphasizing its critical challenge and urgent need for resolution.

Which of the following pain points, if any, currently pose the biggest challenge, are urgent to address, or have the greatest negative impact for your organization?

40%	Software architecture scalability limitations
29%	Application resiliency
19%	Slow engineering velocity/inability to deliver software/ features quickly
11%	Their organization is not facing any significant challenges

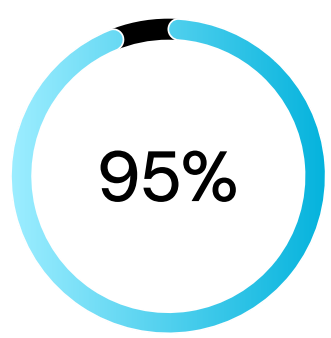
Top three negative business impacts organizations have experienced due to inadequate application scalability

#1	Increased infrastructure and operational costs to compensate for scalability issues - 38%
#2	Delayed product launches or feature releases due to concerns about system capacity - 36%
#3	Productivity losses from engineering teams constantly firefighting scalability problems - 35%

Microservices organizations have a clear advantage compared to monolithic organizations which are more likely to report extremely limited or poor scalability.

Architecture Type	Good/Excellent Scalability	Extremely Limited/Poor Scalability
Entirely Monolithic	48%	27%
Mix of Monolithic and Microservices	51%	16%
Entirely or Predominantly Microservices	64%	11%

Monolithic architectures are nearly 2.5 times more likely to have extremely limited or poor scalability compared to microservices architectures, a challenge that resonates with 95% of enterprises ($500M–$5B) who report that their architecture impacts application scalability.



of enterprises with revenues between $500 million and $5 billion cite that their architecture impacts application scalability

System resiliency: Withstanding failures

Application resiliency is essential for maintaining uptime, ensuring customer satisfaction, and minimizing the impact of failures. The fact that 42% of organizations experienced extremely low or moderate application resiliency is a cause for concern as it leads to decreased productivity, customer churn, and delayed innovation.

The data further highlights that organizations with a monolithic architecture are nearly two times more likely to have extremely poor or low resiliency compared to those with microservices architecture.

The 42% who cited extremely poor, low or moderate resiliency within their organization experienced the top three negative business impacts due to application resiliency issues

39% Decreased productivity and increased operational costs due to time spent recovering from failures

33% Customer churn and loss of market share to competitors with more reliable applications

33% Delayed product innovations or feature releases due to resources spent fixing resiliency problems

Architecture Type	Good/Excellent Resiliency	Extremely Poor/Low Resiliency
Entirely Monolithic	57%	19%
Mix of Monolithic and Microservices	55%	13%
Entirely or Predominantly Microservices	63%	9%

Architectural choices — one size does not fit all

As businesses strive to remain competitive and meet ever-evolving customer demands, the significance of architectural choices becomes increasingly evident yet nuanced. While businesses often face the imperative to modernize and embrace scalable architectures like microservices, the decision is not a one-size-fits-all solution. It hinges on the alignment with specific business goals and considerations.

Monolithic applications, for instance, remain viable options, especially in scenarios where lack of deployment complexity is critical. However, regardless of the architectural approach chosen, addressing technical debt proactively is essential. By carefully balancing architectural decisions and addressing the unique challenges of each approach, organizations can unlock greater engineering velocity, seamless scalability, and resiliency – paving the way for sustained innovation and growth.

Bridging the Gap Between Software Architects' Role and Technical Debt Remediation

Despite many enterprise-wide initiatives in place, there is a lack of alignment in which role is responsible for technical debt remediation.

When asked, "Who owns or is responsible for addressing technical debt in the organization? (Select all that apply)," most organizations selected more than one role/team as being responsible:

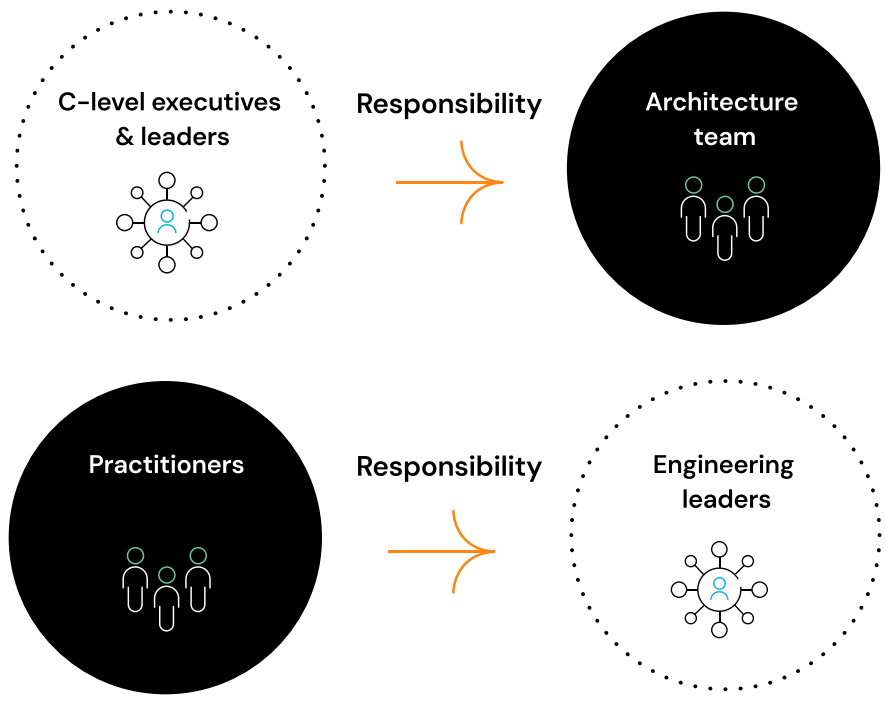
- Enterprise architect or architecture team (48%)
- Each engineering leader (47%)
- Central or head of all engineering teams (47%)
- Application architect (42%)
- Each application/product owner (40%)
- 5% said they don't have anyone responsible for technical debt in the organization

According to C-level executives and leadership roles, the enterprise architect/architecture team is at the top of the list as the primary owner responsible for the organization's technical debt. In contrast, practitioners place engineering leaders at the top and architects closer to the bottom. A clear delineation of responsibilities remains elusive.

Notably, despite their central role as custodians of long-term code quality and architecture, the study uncovered that software architects are disconnected from the CI/CD process. Over a third (37%) report architects are involved in upfront design but have limited involvement in CI/CD.

The reasons behind this disconnect are multifaceted, with a lack of processes, software engineering bandwidth concerns, and fears of architects being a bottleneck among the reasons preventing deeper architect integration in CI/CD.

Who owns or is responsible for addressing technical debt in the organization?



Of the 28% who said software architects don't participate in CI/CD or have very little to no involvement in CI/CD pipelines and release processes, the primary reasons included:

46% There is a lack of processes/ mechanisms/tools to involve architects in CI/CD

43% Architects provide guidance, but engineering owns CI/CD

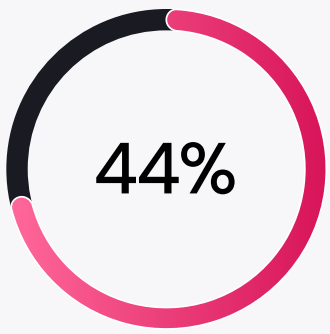
42% Architects are only focused on upfront design, not implementation or head of all engineering teams

41% Architects don't have enough software engineering bandwidth

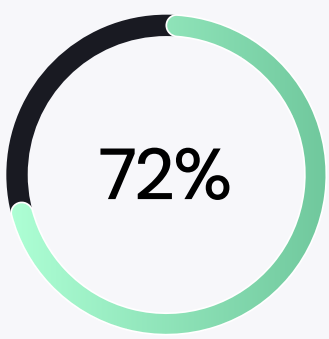
34% There are concerns about architects being a bottleneck

Nevertheless, the data strongly suggests the critical role of software architects in ensuring architectural resilience.

Confidence in architecture resiliency based on software architecture involvement



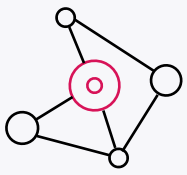
When architects had limited involvement in CI/CD, only 44% of respondents reported confidence in their architecture's resiliency.



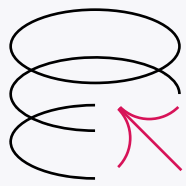
In contrast, when architects were fully involved in the CI/CD process from planning through deployment, a striking 72% expressed a high degree of confidence in their architecture's resiliency.

This disparity indicates that having architects closely engaged throughout the entire software development lifecycle, from initial design to final deployment, leads to more robust, fault-tolerant architectures – a critical foundation for long-term success.

Architectural challenges software architects and engineers have faced over the last five years



Increased complexity in monolithic applications, lack of clear domain boundaries and decreasing modularity (44%)



Software architecture complexity due to disparate technology stacks (42%)

Lack of visibility into architecture, which makes it hard to know which microservices talk to each other (39%)

Insufficient tooling and capabilities exacerbate architectural visibility. When asked if the engineering team, developers, or architects have the tools and capabilities needed to understand and visualize the current state of their software architecture in production as the codebase evolves, more than half (53%) said their architecture visualization tools are limited in scope or they don't have sufficient tooling.

In addition, when asked if their team has an effective way to visualize and understand the potential business impact of architectural changes to applications after implementing them, 49% said their ability to visualize change impacts is limited and inconsistent, or they don't or aren't sure if they have an effective way to do so.

The consequences of this architectural opacity are far-reaching.

When asked "How does an insufficiency of visualizing architectural drift impact or affect your engineering organization? (Select all that apply)," respondents said:

Rework caused by unanticipated change impacts (53%)

Longer change review and approval cycles (51%)

Difficulties in prioritizing changes based on their potential business value (48%)

Increased risk of damaging critical business capabilities (46%)

Architectural Observability and GenAI are Paving the Way for Modernization

As organizations grapple with the complexities of modern software architectures and the perpetual challenge of technical debt, the survey findings point to architectural observability as a promising path forward.

What is architectural observability?

Architectural observability is the ability to analyze an application statically and dynamically to understand its architecture, observe drift, and find and fix architectural technical debt. It enables the understanding of software architecture, helping teams to continuously observe and understand the following each and every release:

- Define domains with dynamic analysis and AI and understand their modularity
- See class and resource dependencies and cross-domain pollution
- Find high-debt classes
- Improve modularity
- Identify dead code and dead flows based on production data
- Identify circular dependencies between services and libraries
- Understand and improve the cloud suitability of the application or domains within it

Architectural observability uncovers class complexity, dead code, and long dependency chains across resources, classes, database tables, and more.

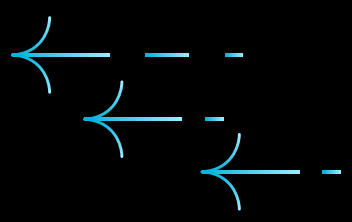
After being provided with the definition of architectural observability, 80% acknowledged that having these capabilities within their engineering organizations would be extremely or very valuable. This resounding consensus among respondents stresses the necessity of having tools and practices to observe and understand the architecture of their software in real-time.

When asked "Over the next year, which of the following potential benefits of architectural observability would be most valuable to you? (Select all that apply)," respondents said:

52%	Assessing architectural complexity and identifying technical risk areas
49%	Visualizing system architecture and domain/ resource dependencies in real-time
48%	Analyzing change impact across architectural components
47%	Identifying the architectural root causes of issues like outages
44%	Detecting architectural drift and violations as the system evolves from release to release
43%	Guided remediation to manage and fix technical debt

The potential impact of architectural observability extends beyond mere visibility; it offers a pathway to more resilient and scalable architectures.

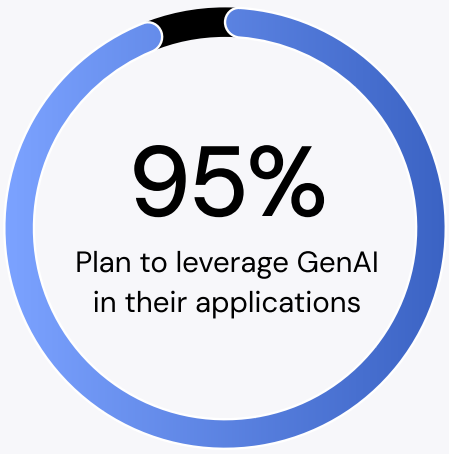
Shift Left



When asked about the most effective approaches to ensuring application resiliency, 40% of respondents advocated for "shifting left" – leveraging architectural observability to proactively address resiliency concerns earlier in the development lifecycle, thereby reducing the likelihood of outages occurring.

Additional approaches to ensuring application resiliency include:

- Relying primarily on application performance management and other observability tools to identify and prevent outages when they occur (30%)
- A combination of both approaches – using architectural observability to improve resiliency early on, and APM/observability tools to identify and prevent outages (29%)

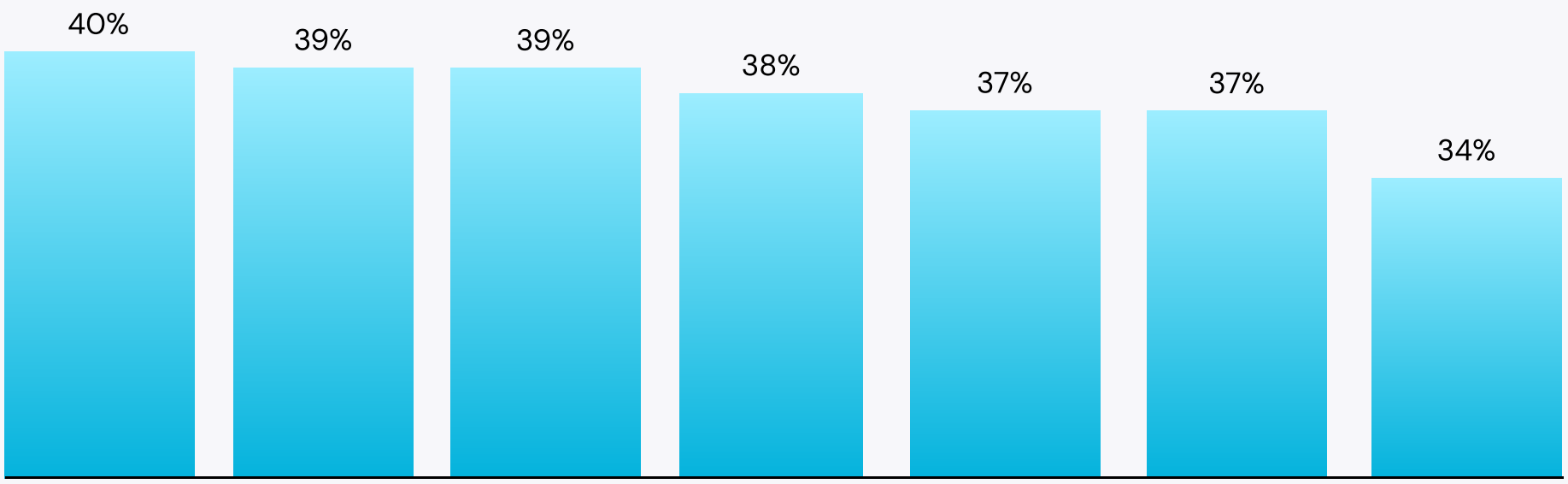


While a small percentage (5%) do not plan to leverage GenAI or are still evaluating potential use cases for GenAI in their applications, the overwhelming majority (95%) recognized the transformative potential of this emerging technology.

Alongside architectural observability, organizations also see generative AI as playing a pivotal role in application modernization efforts. Readiness to adopt generative AI increases with company size, with 44% of companies with $10 billion or more in revenue who say their applications are fully ready, compared to 25% of companies with $100-499 million in revenue.

When asked how organizations intend to use generative AI to modernize applications, 41% indicated they would leverage GenAI to improve application performance and scalability, underscoring the potential synergies between observability, AI, and architectural optimization.

Beyond performance and scalability, respondents highlighted a range of additional use cases for GenAI in application modernization



As organizations navigate the complexities of modern software development, architectural observability and generative AI emerge as powerful allies, offering a path to more resilient, scalable, and effective architectures. By harnessing the insights gleaned from real-time architectural visibility and leveraging the capabilities of AI, organizations can unlock new opportunities for innovation, optimization, and competitive advantage – paving the way for a future where software is not only functional but truly adaptive, intelligent, and aligned with evolving business needs.

Conclusion

The effective management of technical debt, particularly architectural technical debt, has emerged as a critical imperative for organizations striving to stay competitive and meet constantly evolving business demands. The impact of architecture types and company size on challenges resulting from technical debt highlights the need for organizations to adapt their strategies as they scale. The research confirms that software architecture can significantly impact performance.

While there's no universally perfect solution, organizations embracing modern, scalable architectures such as microservices demonstrate advantages in engineering velocity, scalability, and resiliency, enabling them to operate more efficiently and respond to market demands more effectively. Conversely, organizations with monolithic architectures are more likely to face challenges in these areas, resulting in delayed projects, productivity losses, and missed market opportunities.

Architectural choice depends heavily on aligning with specific business objectives and circumstances. Monoliths, for instance, remain viable, particularly when simplifying deployment. By carefully weighing architectural decisions and addressing the unique challenges each approach presents, organizations can foster sustained innovation and growth.

The survey reveals a lack of alignment regarding the roles responsible for technical debt remediation, with software architects often disconnected from the crucial CI/CD process. This disconnect not only hinders the creation of resilient architectures but also underscores the need for better integration of architects throughout the software development lifecycle.

Findings shed light on promising solutions, such as architectural observability and GenAI. Architectural observability offers real-time visibility into software architectures, enabling organizations to assess and alleviate architectural complexity, visualize dependencies and facilitate guided remediation of technical debt. Meanwhile, GenAI is poised to play a transformative role in application modernization, with organizations recognizing its potential to improve performance, scalability, security, and user experiences, among other benefits.

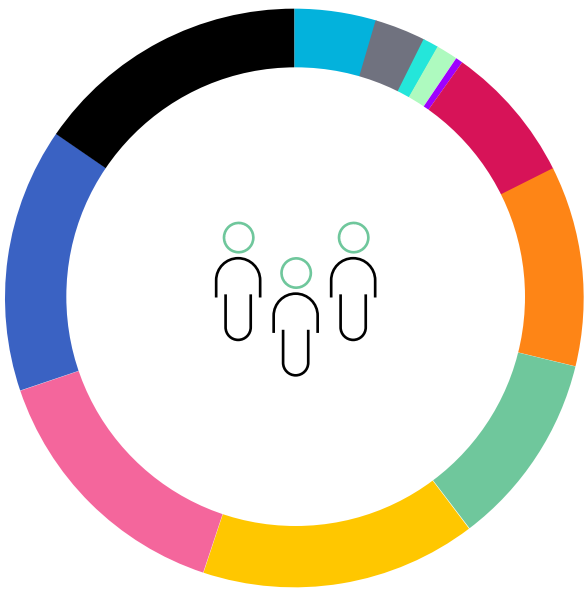
As organizations navigate the complexities of modern software development, embracing these technologies and prioritizing the remediation of architectural technical debt will be critical. In addition to harnessing the power of architectural observability and GenAI, integrating software architects throughout the development lifecycle will be important for organizations to unlock new opportunities for innovation, optimization, and competitive advantage.

About vFunction

vFunction, the pioneer of AI-driven architectural observability, delivers a platform that increases application resiliency, scalability, and engineering velocity by continuously identifying and recommending ways to reduce technical debt and complexity in applications. Global system integrators and top cloud providers partner with vFunction to assist leading companies like Intesa Sanpaolo and Trend Micro in discovering their architecture and transforming applications to innovate faster and change their business trajectory. vFunction is headquartered in Menlo Park, CA, with offices in Israel, London and Austin, TX. To learn more, visit www.vfunction.com.

Survey Respondent Demographics

1,037 U.S.-based respondents were surveyed



Primary role

4.6%	Chief Technology Officer
2.9%	Chief Information Officer
0.9%	Chief Software Architect
1.2%	Chief System Engineer
0.4%	Chief Development Officer
7.8%	Head of Architecture
11.2%	Head of Engineering / VP of Engineering
10.8%	Head of Product
15.4%	VP Application Development
14.7%	Lead Software Architect / Software Architect
14.8%	Developer Experience
15.4%	Product Owner

Industry

3.6%	Healthcare:
6.8%	Financial Services
12.7%	Manufacturing
4.4%	Education
38.5%	Computer Software/Hardware
2.0%	Media & Entertainment
2.5%	Marketing & Advertising
1.3%	Hospitality
0.7%	Public Administration/Government
1.9%	Energy
8.1%	Retail & CPG
3.9%	Transportation
3.3%	Insurance
7.0%	Telecommunications
3.4%	Other

Company size based on the total annual revenue for the last fiscal year