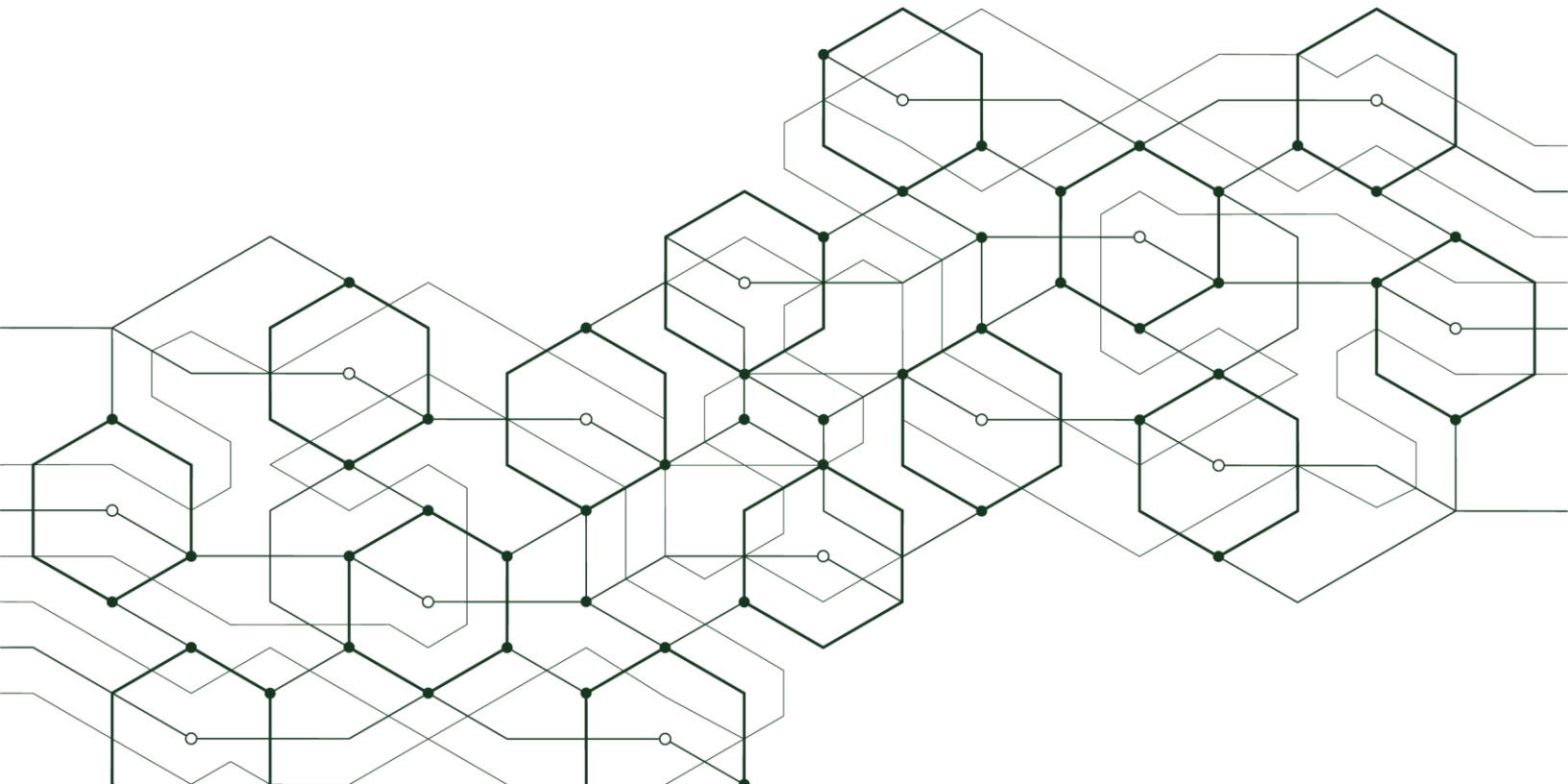
VDC|Strategy

May 2026
IoT, Edge & Embedded Technology

Memory-Safe Languages Offer Protection for Devices and TCO

The Case for Improved Software Quality

by **Chris Rommel**, *Executive Vice President,* with **Dan Mandell**, *Director*



Key Takeaways:

- >85% of engineers report that language choice can impact safety and security
- SPARK, Rust, and Ada are rated as the best languages to address safety and security
- Ada users are 4X more likely to complete their projects ahead of schedule
- Memory-safe languages can save 60% of lifecycle software maintenance costs

Executive Summary

85% of engineers report that programming language choice directly impacts safety and security. Memory-safe languages such as Ada, SPARK, and Rust help reduce dangerous safety and security vulnerabilities, simplify verification, and mitigate long-term corporate risk. These languages were designed with stronger type systems and controlled memory management that can significantly reduce the risk of common memory-related issues.

These risks are now changing both engineering practices and public policy. While these risks are not new, they have been haunting software developers for decades, and the fact that systems are becoming increasingly complex and interconnected is triggering change. The potential consequences stemming from errors such as buffer overflows, invalid pointers, and data leakage prompted the White House Office of the National Cyber Director to recommend broader adoption of memory-safe languages. Moreover, new requirements and regulations such as the Cyber Resilience Act (CRA) and "Secure by Design" initiatives from CISA and the NSA are also driving adoption of memory-safe languages.

- **More engineers are adopting memory-safe languages**
  - **Ada steadily increasing in use** – Ada has proven itself in high-integrity systems over decades. While the use of some software development languages has declined, Ada continues to see use in a range of devices, reflecting its long-standing role in safety-critical environments.
  - **Rust set to double in 3 years** – Rust adoption is expected to increase significantly from 6.8% of current projects to 14.0% in three years, often as a replacement for or addition to non-memory-safe alternatives, such as C or C++.
  - **Ada, SPARK, and Rust strengthen AI-generated code** – Memory-safe languages help reduce defects from AI-generated code and provide additional verification capabilities. Users of memory-safe languages report a higher level of trust in AI-generated code than those using C or C++.
- **Leveraging memory-safe languages drives ROI**
  - **Lower development cost** – Companies leveraging memory-safe languages report significantly lower development costs, with reductions of up to 60% in some cases.
  - **Reduce time to market** – Ada users are 4X more likely to complete their projects ahead of schedule than are those that rely on C or C++.
  - **Promote focus on innovation** – Organizations using memory-safe languages devote 1.7X more of their development effort on the customer value-adding areas of analytics and AI.

Background on VDC's Research

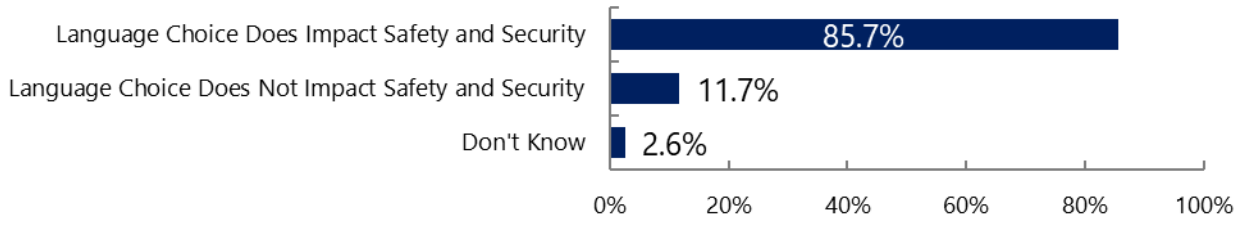
VDC has covered engineering and industrial technology markets since 1971. The analysis and supporting discussions in this paper are based on VDC's ongoing research in the embedded engineering market and by findings from a survey of more than 500 decision makers familiar with embedded software engineering within their respective organizations. The survey included engineers and development managers responsible for embedded software development. This survey offers insight into leading business and technical trends affecting engineering organizations as well as the best practices implemented to address them. Respondents span a range of industries including automotive, aerospace, defense, energy and utilities, industrial, and medical devices, among others.

Introduction

Memory-safety vulnerabilities remain one of the most significant sources of software defects. Many of the most serious software vulnerabilities originate from memory management errors such as buffer overflows and invalid pointer access. Already, more than 85% of engineers report that programming language choice can impact safety and security [See Exhibit 1]. Languages such as Ada, SPARK, and Rust feature stronger type systems and controlled memory management, which can significantly reduce the likelihood of memory-related defects compared with traditional memory-unsafe languages.

Exhibit 1: Consideration of Software Development Language Choice Impact on Product Safety and Security

(Percentage of Respondents)

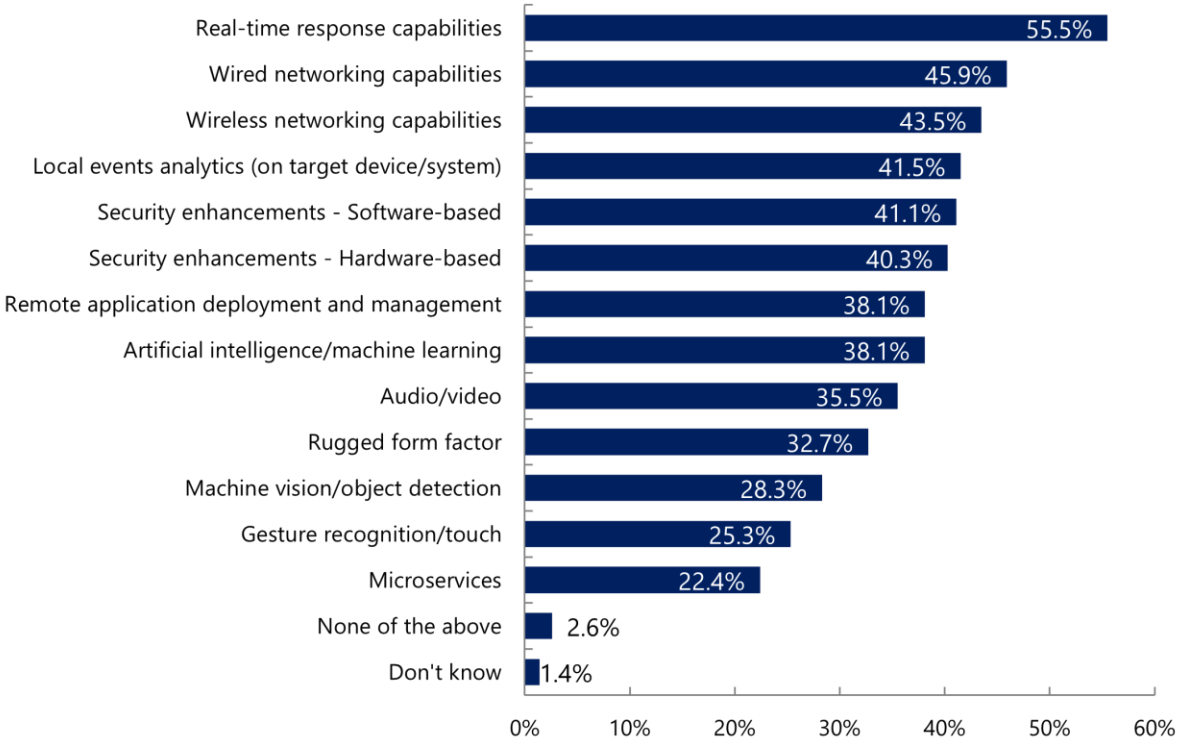


Recognition of these risks has also driven public policy changes. In 2024, the U.S. Office of the National Cyber Director urged engineering organizations to select memory-safe languages. The impact of these capabilities extends beyond safety alone, however. New requirements and regulations such as the Cyber Resilience Act (CRA) and “Secure by Design” initiatives from CISA and the NSA are also driving adoption of memory-safe languages.

In safety-critical and high-integrity embedded systems, the consequences of system failure are dire. Most embedded systems from industries such as aerospace, automotive, industrial automation, and medical devices must already operate under strict reliability and safety requirements. In fact, 55% of engineers report real-time response requirements on their current project [See Exhibit 2]. Adding additional layers of technology such as AI, machine vision and enhanced security on top of already complex systems places greater demands on the development teams. When implemented using existing languages, the added complexity can increase exposure to memory safety risks as well as create backlash if misalignment with regulatory guidelines emerge.

Exhibit 2: Capabilities/Features of Current Project

(Percentage of Respondents)



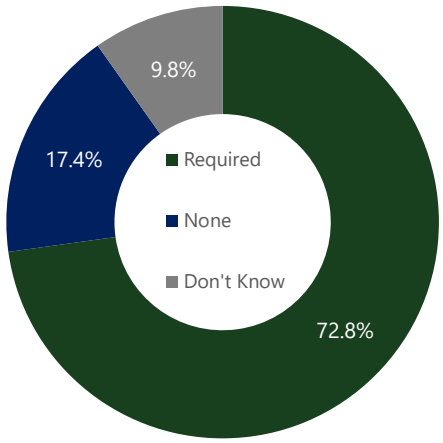
Within these environments, software defects can lead to operational downtime, certification delays, regulatory exposure, tort liabilities, and increased lifecycle costs. The selection of software development language can materially impact schedule adherence and the long-term operational costs associated with the deployed embedded systems. These dynamics can damage vendor reputations with customers and end users, require significant time and effort to repair, and may prompt overcorrections in the software development process. Engineering organizations must therefore evaluate software development language both as a mechanism to reduce risk and to improve development costs, schedule, and lifecycle software maintenance.

Safety and Security Require Careful Language Choice

More than 70% of engineering projects must align with a safety-critical process standard [See Exhibit 3], including DO-178 for avionics, IEC 61508 for industrial, and ISO 26262 for automotive. These standards impose strict requirements for verification, traceability, and defect mitigation. Combined with the consequences of system failure, they necessitate the careful evaluation of technologies used to design, develop, and deploy them.

Exhibit 3: Safety-Critical/Process Standards Required for Current Project

(Percentage of Respondents)



Software development practices are also evolving within safety-critical device markets. Traditionally-developed, in-house software now represents less than one-third of production codebases [See Exhibit 4]. The increasing reliance on third-party software and AI-generated code introduces additional quality assurance challenges. In fact, more than 55% of the engineers surveyed expect the amount of AI-generated code to increase in their next project, with most expecting growth of more than 25%. Trust in that AI-generated code remains uneven, however. A majority of respondents cite concerns with safety/compliance (55%) and security (59%), reinforcing core concerns within this domain.

Despite its productivity benefits, AI code generation often introduces a high volume of new errors. The volume of code produced and developers' growing distance from traditional programming practices actually increases the need for time-intensive testing and validation. Memory-safe languages can play a valuable role in mitigating these issues. For example, SPARK's formal verification capabilities can accelerate the verification process and thereby increase confidence in AI-generated code. In fact, memory-safe language users report a higher level of trust in AI-generated code than developers using C or C++, demonstrating a growing recognition that languages' core characteristics improve reliability [See Exhibit 5].

>70% of projects must align with a safety-critical process standard

Exhibit 4: Percentage of Total Software Code in Final Design Coming from Various Origins

(Average of Responses)

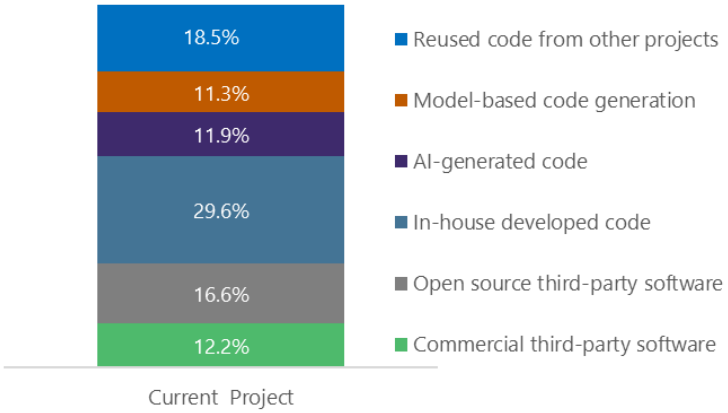
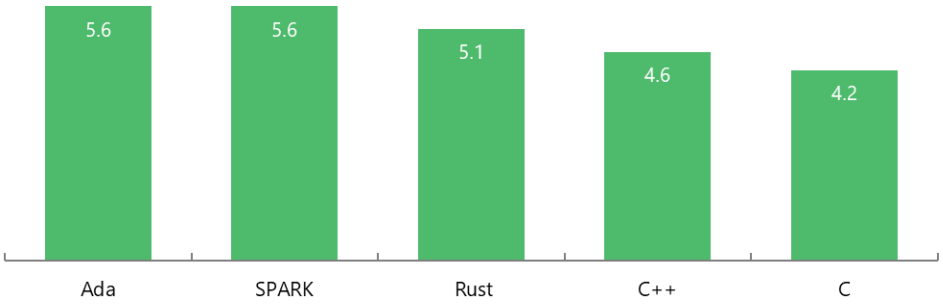


Exhibit 5: Level of Trust of AI-Generated Software/Code

(Average of Responses; 1=Not Trustworthy at All; 4=Low Trust; 7=Extremely High Trust)



Already, more than 85% of engineers recognize embedded software development languages as a key mechanism to improve system security and safety. Memory-safe languages emerged as the recognized leading choice to address these issues. Among the dozens of options for embedded software development language, engineers ranked Ada, SPARK, and Rust as the leading languages for addressing safety and security requirements [See Exhibit 6]. These findings indicate growing alignment between language choice and safety requirements in embedded software development.

Exhibit 6: Perceived Ability to Best Address Safety and Security Requirements

(Ranking out of all 26 languages rated)

1. SPARK
2. Rust
3. Ada

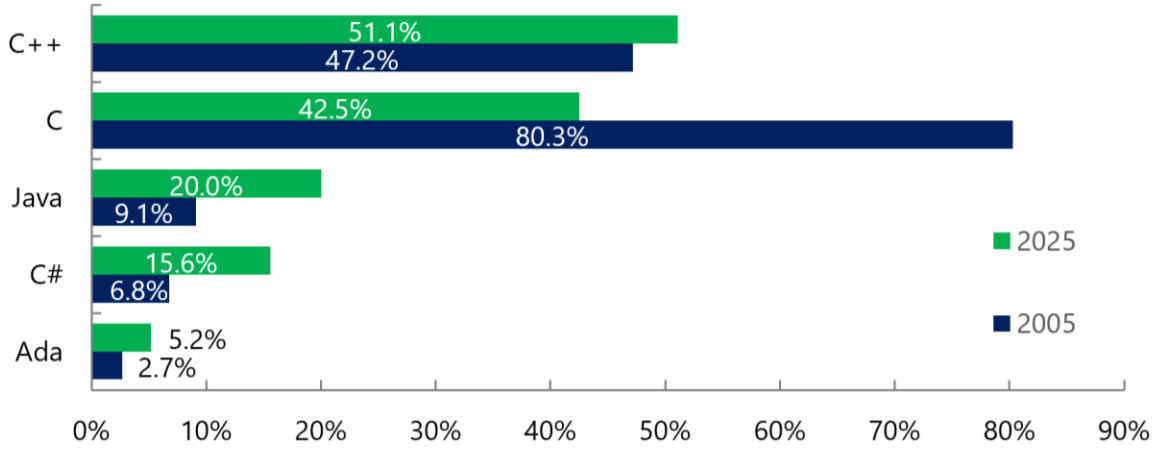
SPARK, Rust, and Ada are rated as the best languages to address safety and security

Evolution of Language Choice

Embedded software development traditionally relied on C and other lower-level languages. Over time, increasing system complexity, resources, and application functionality requirements gradually drove many organizations to evaluate new alternatives. For some applications, higher-level, object-oriented platforms such as C++ and Java gained adoption, while scripting languages, such as Python, gained favor for other workloads, such as analytics. However, some systems, especially safety-critical ones, necessitate a different, and proven set of choices that can ensure deterministic functionality and reliability. Rust has gained attention for its memory safety guarantees. Ada, and its formally analyzable subset SPARK, have long provided memory safety and remain widely used in safety-critical systems that require strong verification and certification support. To this end, the use of Ada has doubled over the past two decades of VDC tracking [See Exhibit 7].

Exhibit 7: Current Project Embedded Software Development Language

(Percentage of Respondents)



**Not all software development languages shown.*

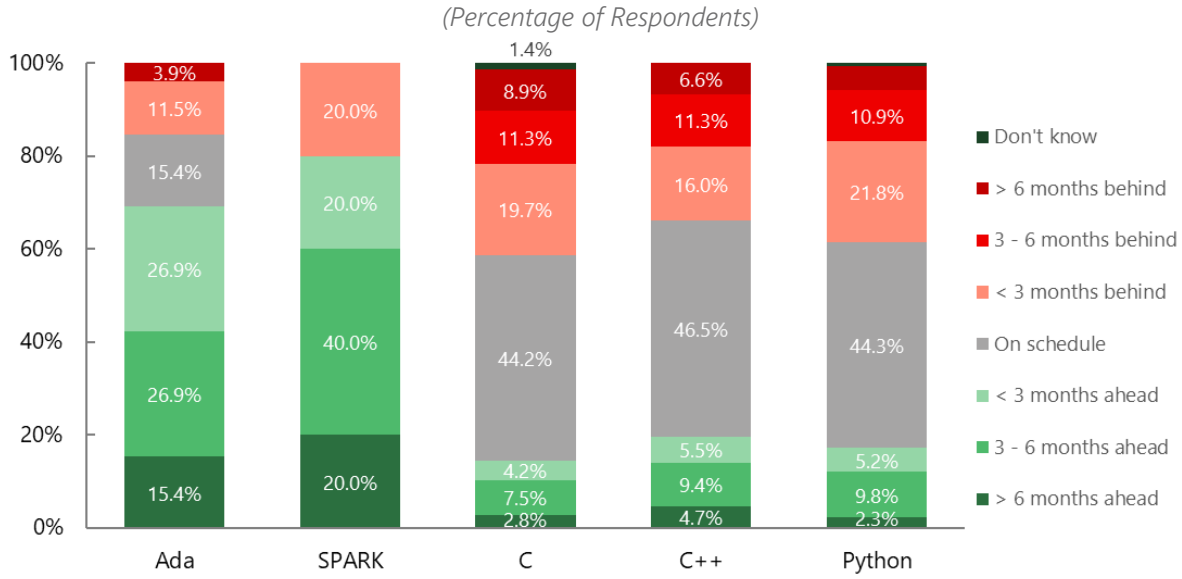
While C use has declined from prior highs, Ada has remained a trusted, reliable option for engineers in safety- and mission-critical markets. Adoption has also accelerated for other memory-safe languages overall. For example, 14.0% of respondents expect to use Rust in three years, an increase from 6.8% of respondents in current projects. Emerging engineering challenges will drive many organizations to reevaluate new – and, in some cases, proven – technology to meet the range of development and financial requirements facing engineering organizations. In this environment, software development language is not a neutral choice. It can materially affect defect rates, verification effort, schedule adherence, and long-term operational costs.

14% of developers expect to use Rust in three years

On-Time Delivery with Memory-Safe Languages

29% of projects are reported as behind schedule. New development and capacity challenges are forcing many product development organizations to seek proven ways to improve time-to-market and schedule adherence. This environment underscores the need for solutions that improve development efficiency while maintaining reliability. While agentic AI may provide efficiency gains in code generation, it can also cause additional software testing burden due to the volume of code and errors produced. Moreover, any schedule delays translate into added operational costs and missed sales and/or market windows. Beyond direct cost impacts, on-time deliveries strengthen customer trust and organizational agility, allowing internal resources to be reallocated to new priority projects. Primary software development language choice remains a key lever to achieve these goals and reduce schedule risk. Memory-safe languages like Ada and SPARK can help reduce the time spent on QA and improve schedule predictability [See Exhibit 8].

Exhibit 8: Current Project Schedule Adherence, Based on Current Project Language Use



The benefits of memory-safe language use extend far beyond safety and security. In fact, schedule performance of Ada and SPARK projects is substantially better than that of projects using other languages. For example, Ada projects are:

- **> 4X** more likely to be ahead of schedule than C projects (e.g., 69% for Ada; 15% for C)
- **> 3X** more likely to be ahead of schedule than C++ projects
- **> 1.7X** more likely to be ahead of schedule than Python projects

Ada projects are 4X more likely to be ahead of schedule than C/C++

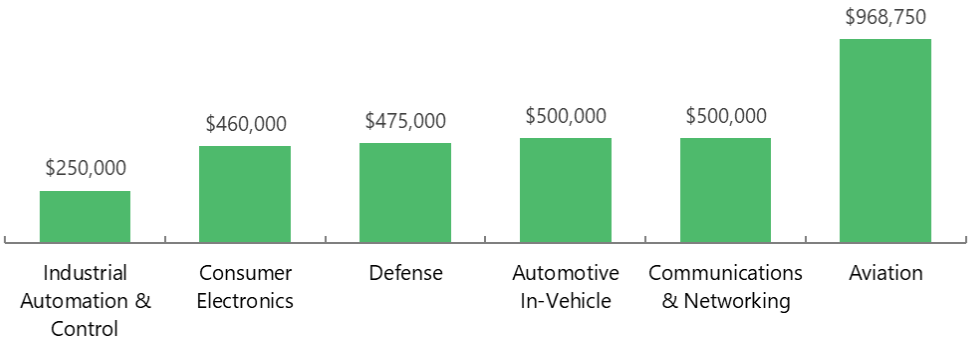
While not as widely used, SPARK projects show even stronger results. In fact, 80% of SPARK users reported projects ahead of schedule – representing the strongest schedule performance observed in the survey.

De-Risking Development and Driving Long-Term Cost Savings

Project development costs, including development labor, tools, and licensing, vary significantly based on factors such as footprint, processing needs, and industry, among others [See Exhibit 9]. With industry project requirements likewise dictating many components and development solutions, engineering organizations must identify ways to control development costs.

Exhibit 9: Total Cost of Development, Based on Industry

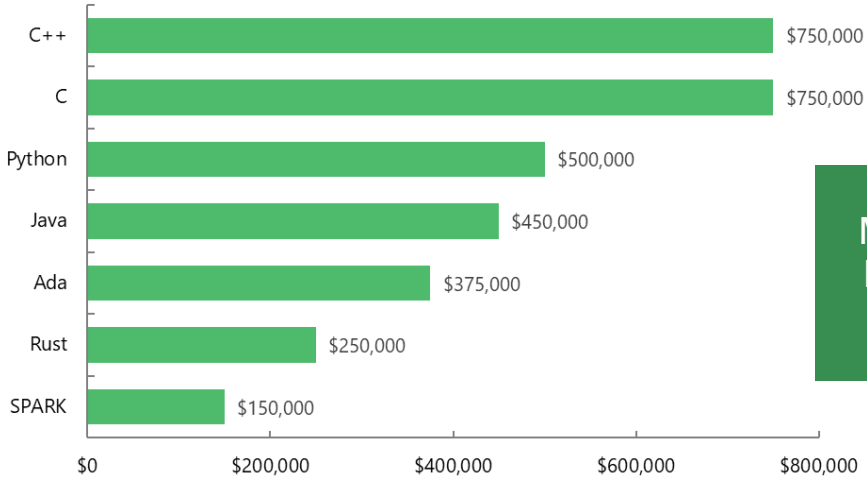
(Median of Responses, USD)



Programming language choice correlates with significant differences in development cost outcomes. When asking engineers to estimate their current project development costs, the use of memory-safe languages emerged as a key variable in lowering costs. In fact, projects using memory-safe languages have median development costs 2X to 4X times lower than projects using C and C++ [See Exhibit 10].

Exhibit 10: Total Cost of Development, Based on Current Project Programming Language

(Median of Responses, USD)



Memory-safe languages projects have costs 2X to 4X times lower than C and C++ projects

Memory-Safe Languages Reduce Long-Term Operational Costs

Embedded devices often have deployment lifecycles of 5 to 10 years or more. In addition to managing up-front costs and improving schedule adherence, organizations must identify ways to manage the deployed devices' long-term operational costs. Decisions made during the development cycle can have lasting operational and financial impacts.

Use of memory-safe languages can save 60% or more in patch and defect remediation costs

While IT systems are generally updated and patched at regular, sometimes automatic intervals, patch management for embedded devices represents a considerable challenge for deploying organizations (and the device makers themselves). System updates must be carefully planned and coordinated to minimize operational downtime. Patches are challenging to manage due to poor device visibility, the availability of patches from vendors, and various internal stakeholders. Technologies that reduce the frequency and magnitude of post-deployment fixes can help minimize both operational costs and customer disruption.

Exhibit 11: Software Defect and Patch Costs per Project, Based on Current Project Language Use

	Ada	SPARK	Rust	C	C++	Python
A. Defects Reported by Customer per Year	6	6	9	9	8	6
B. Number of Software Patches Required by Customer per Year	4	4	5	5	5	5
C. Total Combined Person-Hours of IT and Engineering Time Required for Each Defect Remediation	15	5	12	20	17	20
D. Total Combined Person-Hours of IT and Engineering Time Required for Each Software Patch	12	9	25	20	20	20
Fully-burdened FTE Cost*	$130,000	$130,000	$130,000	$130,000	$130,000	$130,000
E. Hourly Cost*	*$65*	*$65*	*$65*	*$65*	*$65*	*$65*
Yearly Cost	$8,970	$4,290	$15,145	$18,200	$15,340	$14,300
Difference Compared to Ada		-52%	69%	103%	71%	59%
F. Deployed Years in Field*	7	7	7	7	7	7
Operational Cost Impact	**$62,790**	**$30,030**	**$106,015**	**$127,400**	**$107,380**	**$100,100**
Difference Compared to Ada		*-52%*	*69%*	*103%*	*71%*	*59%*

**Based on median global responses and 2,000 working hours per person per year.*

*Operational Cost Impact = [(A*C*E)+(B*D*E)]*F*

Differences in patch and defect remediation effort drive significant lifecycle cost differences. Over the average 7-year deployment lifecycle, the costs of addressing defects and patches compound, amplifying the impact of programming language choice. While the number of defects and patches varies only modestly based on language use, the engineering effort required to resolve each issue differs substantially. Based on this dynamic, projects using memory-safe languages are associated with total cost differences approaching or exceeding 60% in some cases [See Exhibit 11].

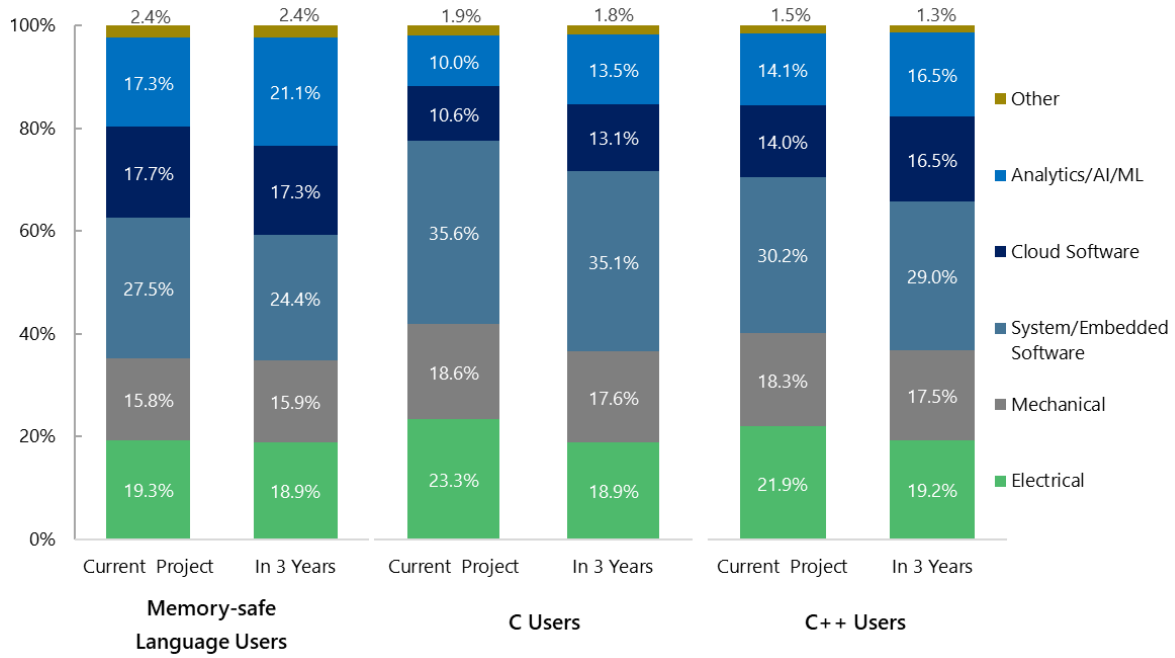
Memory-Safe Languages Enable Greater Focus on Differentiation

Programming language choice impacts the allocation of project costs across engineering domains. The predictability and efficiency provided by memory-safe languages enable engineering organizations to allocate more resources to increasing product value and to the development of a differentiated software stack versus lower-level software development and verification tasks [See Exhibit 12].

For example, memory-safe language users reported 17.3% of their project costs allocated to Analytics/AI, while the average C projects allocated just 10.6%. Moreover, this dynamic is expected to accelerate over the next three years, with developers spending 38.4% of their development costs on AI or Cloud software. The differences in the application of engineering resources on value drivers impact not only today's projects but also allow organizations to develop the in-house IP and expertise that will pay dividends in future product generations.

Exhibit 12: Estimated Allocation of Engineering Development Costs by Domain

(Average of Responses)



Conclusion: How Memory-Safe Languages Can Help

As safety and security requirements become increasingly critical, engineering organizations must place greater emphasis on development technologies that can improve software quality while controlling project costs. The use of memory-safe languages, such as Ada, SPARK, and Rust, helps prevent defects earlier in the development cycle and is associated with improved schedule and cost outcomes. To achieve comparable project and safety outcomes, other widely-used languages, such as C and C++, often require additional effort, tooling, and processes to reduce risk and compensate for the inherent weaknesses of the language.

- **Ada is designed for high reliability –** Ada's semantics reinforce deterministic execution and timing behavior, enabling many issues to be identified at compilation instead of runtime. Its concurrency models and absence of garbage collection help accelerate verification and validation. In addition, its ecosystem of qualified tools, compilers, and certification artifacts can reduce effort for compliance with safety-critical process standards.
- **SPARK is well suited to support AI-generated code workflows –** The subset of Ada enables higher assurance through formal verification and development. It can help verify the absence of certain runtime errors. This can also help "shift verification left" to reduce development timelines and downstream risk. Moreover, security can also be improved through the reduction of these same issues. These same traits drive value within agentic AI workflows, by allowing engineers to focus on specifications while SPARK verifies implementation correctness.
- **Ada and SPARK are proven in the next-generation safety-critical device classes –** While widely used in safety-critical markets such as aerospace, we expect Ada and SPARK's use – and the recognition of their value – to continue to expand into new markets. For example, AdaCore and NVIDIA recently published an ISO 26262-focused reference process that discusses the role and value of Ada and SPARK for automotive software development, while Zenseact is using SPARK for ASIL components. In addition, outside of embedded development, financial services firm BNP Paribas used Ada for its real-time pricing systems, showing the language's utility for a wide range of use cases.
- **Rust's growing adoption offers new developers proven benefits –** Like Ada and SPARK, Rust is designed to prevent many memory-related errors at compile time. Moreover, interest in Rust is expanding, with surveyed respondents expecting an increase from 6.8% of current projects to 14.0% in three years as more engineers look for languages capable of meeting their next-generation development goals.
- **Memory-safe languages drive measurable cost and productivity gains –** The benefits of memory-safe languages extend beyond safety. Organizations using those languages report lower development costs, fewer defects, and reduced lifecycle maintenance costs by 60% or more. For organizations developing safety-critical systems, Ada, SPARK, and Rust provide a mature ecosystem of tools, certification support, and verification capabilities that can help engineering teams improve reliability while focusing on higher-value work.

About the Authors

Chris Rommel
Executive Vice President,
IoT & Industrial Technology

For two decades, Chris has helped clients drive growth in operational and industrial technology businesses, from start-ups to middle-market companies to global enterprises. He has helped a wide variety of clients respond to and capitalize on diverse opportunities in AI, security, IoT, Industry 4.0, the intelligent edge, value-added hardware, semiconductors, engineering solutions, industrial and operational cloud computing and more. Chris has extensive growth strategy consulting expertise including new market assessment, product strategy, M&A diligence, partner and ecosystem development, thought leadership content creation, and more. A frequent speaker at major industry events, Chris has written and published extensive research and thought leadership content on many dynamic technology markets. Chris holds a B.A. in Business Economics and a B.A. in Public and Private Sector Organization from Brown University.

Email Chris at **crommel@vdcstrategy.com**

Dan Mandell
Director,
IoT & Embedded Technology

Dan supports a variety of syndicated market research programs and custom consulting engagements in the IoT and Embedded Technology practice. He leads VDC's annual research services for embedded processors, boards, integrated systems, IoT gateways, and other computing hardware. Dan's insights help leading technology providers align their go-to-market planning and competitive strategies with the dynamic embedded landscape and its constantly evolving buyer behaviors, technology adoption, and application requirements. His working relationship with VDC dates back to 2005 and includes time supporting Business Development as well as the AutoID practice. Dan holds a B.S. in Information Systems Management from Bridgewater State University.

Email Dan at **dmandell@vdcstrategy.com**

About VDC Strategy

Founded in 1971, VDC Strategy provides in-depth insights to technology vendors, end users, and investors across the globe. As a market research and consulting firm, VDC's coverage of AutoID, enterprise mobility, industrial automation, and IoT and embedded technologies is among the most advanced in the industry, helping our clients make critical decisions with confidence. Offering syndicated reports and custom consultation, our methodologies consistently provide accurate forecasts and unmatched thought leadership for deeply technical markets. Located in Southborough, Massachusetts, VDC prides itself on its close personal relationships with clients, delivering an attention to detail and a unique perspective that is second to none.

508.653.9000 | **info@vdcstrategy.com**