VDC Research
Insights for the Connected World

Controlling Costs with Software Language Choice

How Ada Can Help

License to Distribute: AdaCore

By Chris Rommel, Executive Vice President

TABLE OF CONTENTS

Introduction 3

IoT Forcing Reevaluation of Development Economics 6

Total Cost of Development 7

- VDC's TCO Calculator
- Example Total Cost of Development Calculations
- So Why is Ada Saving Organizations Development Cost?
  - 1. Lower Cost Development Resources
  - 2. Average Ada Projects are Shorter in Duration
  - 3. Ada Projects are Reported as More Likely to Be on Schedule
  - 4. Focused Resource Application
  - 5. Software Development Cost Savings

"How" Ada Can Help 13

- 1. Reliability
- 2. Reusability
- 3. Flexibility and Scalability
- 4. Ease of Use

Conclusion 14

About VDC Research 14

LIST OF EXHIBITS

Exhibit 1: Primary Factor Driving Respondent's Organization to Utilize/Deploy IoT Capabilities

Exhibit 2: Capabilities or Features Included in Current Project

Exhibit 3: Types of Processors Used on Current Project

Exhibit 4: Largest Overall Challenge for Respondent's Organization in Developing IoT Solutions

Exhibit 5: Potential Software Development Costs Change per Device, Communications/Networking/x86-based Project

Exhibit 6: Potential Software Development Costs Change per Device, Aerospace and Defense /ARM-based Project

Exhibit 7: Potential Software Development Costs Change per Device, Automotive/ARM-based Project

Exhibit 8: Total Project Length in Calendar Months (Time from Initial Specification to Shipment), Segmented by Software Development Language Use

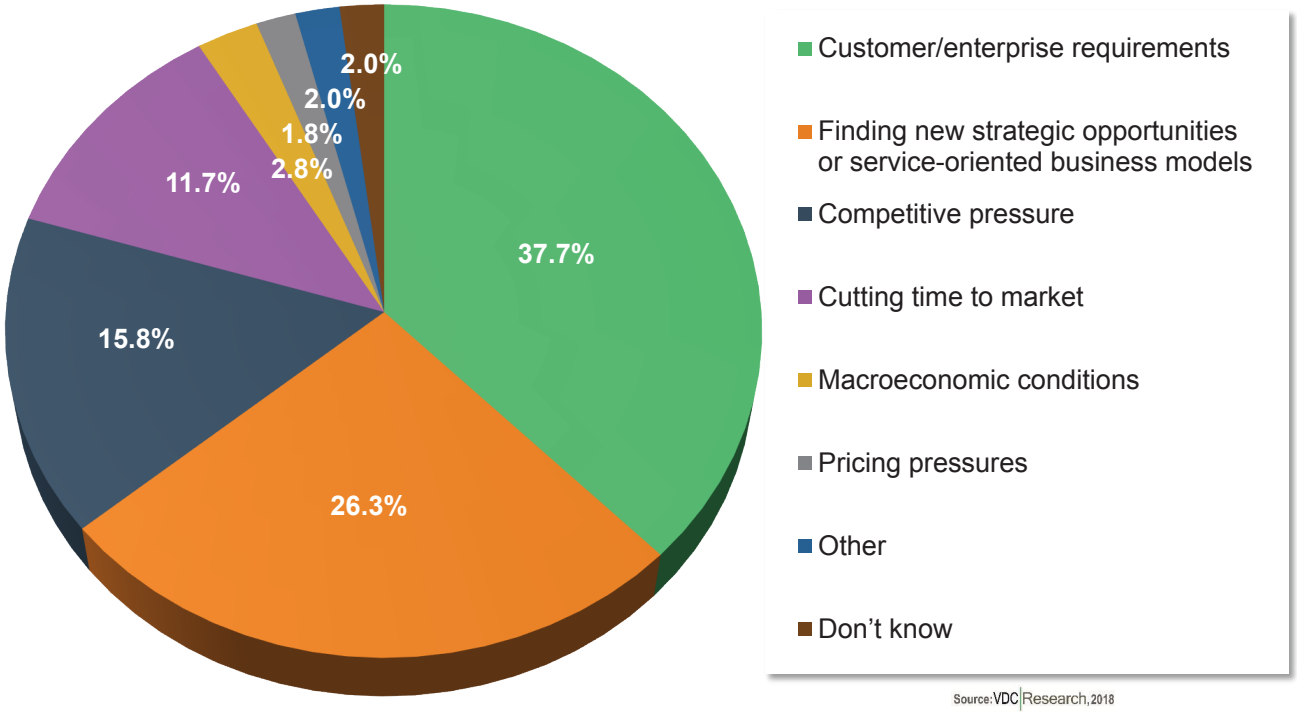
Exhibit 10: Estimated Distribution of Development Costs on the Project, Segmented by Software Development Language Use

Exhibit 11: Estimated Total Cost of Development for the Project, Segmented by Software Development Language Use

INTRODUCTION

Much has been written about impact of IoT. There are new customer requirements, new business models, and competitive pressure all driving change within engineering organizations. These shifting requirements have catalyzed more and more need for connectivity middleware, remote management capabilities and increasing interest in technologies like machine vision and energy harvesting.

Exhibit 1: Primary Factor Driving Respondent's Organization to Utilize/Deploy IoT Capabilities (Percentage of Respondents)



Source: VDC Research, 2018

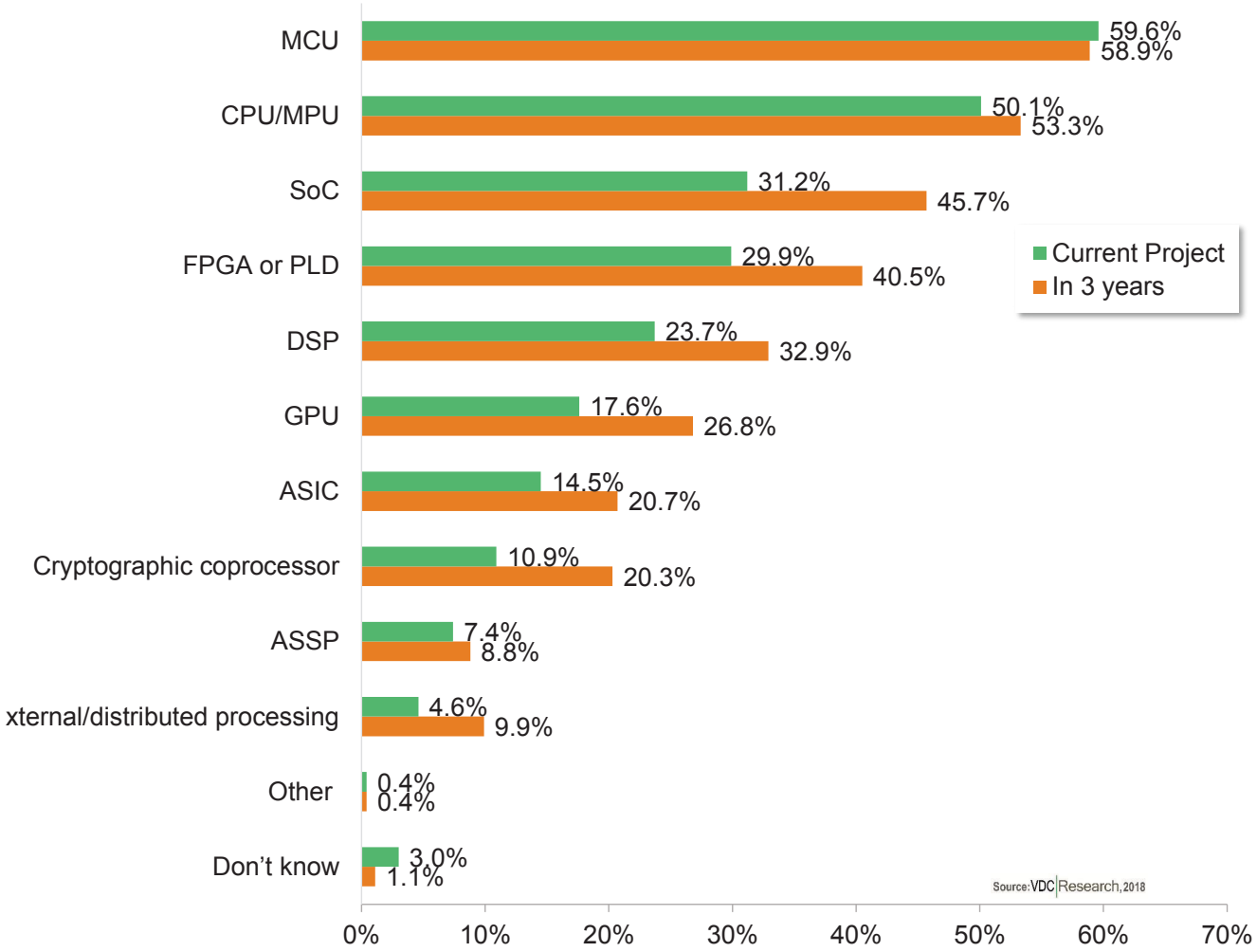
As much hype and promise surrounds the IoT, connected embedded systems are not new. Initiatives to derive more synergies through connected factories and manufacturing environments have been around for decades. Cars and trucks with telematics and services such as OnStar are likewise old concepts. However, the industry has unequivocally reached a new point of concavity in its pairing of increasingly connected systems with corporate business directives and business models. The combination of these technical and business forces has created a self-reinforcing cycle, driving greater adoption and investment in new technology. Despite many organizations and solutions vendors' pursuit of new IoT-driven opportunities and functional requirements, these new goals have not diminished the challenges associated with traditional embedded engineering requirements. For example, there is an unabating need for solutions catering to real-time and safety-critical applications across a number of embedded device classes. In fact, nearly half of the respondents to our survey indicated that their current project included real-time response capabilities. In most cases, the expectations for ubiquitous connectivity and situational awareness and adaptation only augment the risks associated with the failure of time-sensitive or safety-critical systems.

Exhibit 2: Capabilities or Features Included in Current Project
(Percentage of Respondents)

Note: Percentages sum to greater than 100% due to multiple responses.

These feature set changes are also accelerating the adoption of new hardware and software products. For example, OEMs are integrating more robust (often memory-hungry) operating systems and connectivity stacks, with a plurality of respondents to our survey now using Linux-based OSs. Additionally, new designs are incorporating an increasing number of complex, multi-processor/SoCs. Five years ago, we saw a shift and adoption of CPU+MCU (e.g. big.LITTLE, etc.) designs to optimize function and power performance. Now, new technologies such as machine learning, vision processing and security requirements are accelerating this change already underway due to the increasingly complex software and business requirements. Engineers are responding, increasing their utilization of more complex SoCs, GPUs, FPGAs and crypto processors to tackle the computationally-intensive workloads of today's connected devices (See Exhibit 3). In isolation, any new technology adoption can cause development challenges. The combination of multiple technology shifts and new business goals can make these issues all the more difficult to navigate. As a result, engineering organizations must be sure to evaluate solutions not only to address functionality gaps in their current products, but also to help them derive new development efficiencies.

Exhibit 3: Types of Processors Used on Current Project
(Percentage of Respondents)

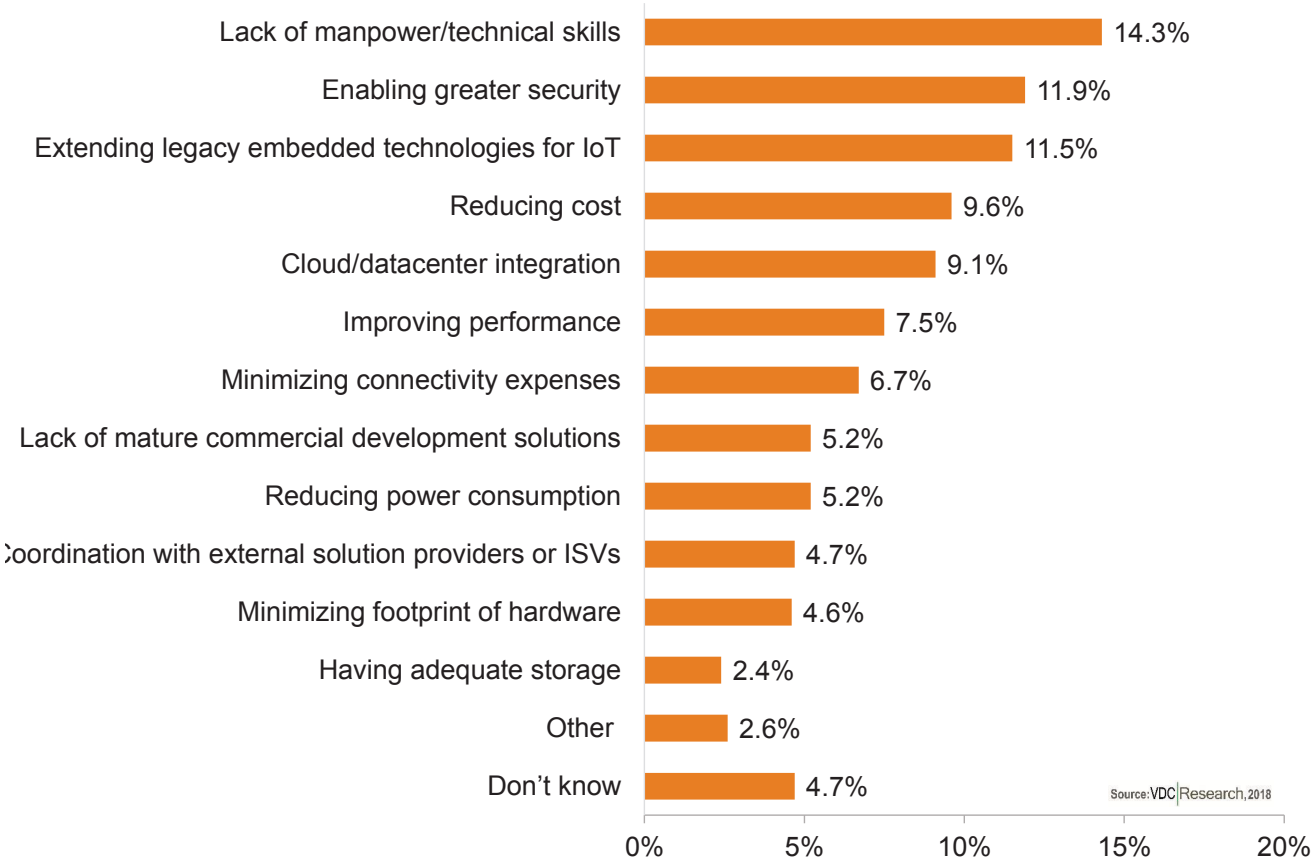


Note: Percentages sum to greater than 100% due to multiple responses.

IOT FORCING REEVALUATION OF DEVELOPMENT ECONOMICS

IoT development is moving many organizations into uncharted territory where they lack the resources and/or expertise to efficiently bring IoT solutions to market. Now, the connectivity demands associated with the IoT bring with them new, or augmented, challenges such as 'enabling greater security' and 'improving cloud/IT integration' (See Exhibit 4). Despite these new variables, development resources and cost are still recognized among the largest challenges for IoT solution development. New IoT-specific concerns exacerbate theses traditional issues.

Exhibit 4: Largest Overall Challenge for Respondent's Organization in Developing IoT Solutions (Percentage of Respondents)



In the midst of this evolving landscape of obstacles, schedule and time-to-market challenges have been expanding as organizations face new development and capacity challenges. Project schedule adherence remains a significant obstacle for product development organizations, with over one-third of projects already reported as late. Not only do these excessive delays create added operational cost, but they can also impact revenue opportunities through missed market windows and sales. Clearly, there is inherent value in predictability, which can earn trust with clients and allow additional investments based on improved visibility into operational expenditures. Despite the relatively high rate of project delays, performance against this metric has improved over the past few years. Part of this improvement is certainly from the adoption of new development technologies and methodologies. However, the magnitude of IoT-fueled business and development requirement changes necessitates additional adaptations to maintain and/or improve upon current schedule adherence levels.

VDC's TCO Calculator

VDC's IoT & Embedded Engineering Survey had over 700 respondents from a range of industries. Those results were used to build a total cost of ownership calculator.

The Total Cost of Ownership calculator uses a variety of statistics in its computation such as: devices per project, Bill of Material costs, distribution of development costs by engineering discipline on current project, number of engineers per project, fully-loaded labor cost average per engineer, project length, product's average years of useful life once deployed in field by the end client, estimated total number of defects or software patches customers report/require per deployed year in field, estimated combined IT and engineering time (hours) required for each patch or defect remediation, and estimated percentage of devices that will become inoperable and require repair or replacement each year.

Our comparative cost calculations emphasize the results from projects from the same vertical market using the same software development language as well as those from projects from the same vertical market using the same processor architecture. Additionally, we also take into account results for projects using the same processor architecture, same language, and same vertical market.

The IoT/Embedded market is incredibly complex and heterogeneous. The wide range of features and form factors necessitated across the various embedded vertical markets lead to equally wide number of factors that drive selection of specific hardware and software components. Furthermore, technology selection is influenced not only by current requirements, but also by investments made during past projects whose substitution can lead to additional labor and cost considerations. Given the multivariate evaluations undertaken by engineering organizations when selecting technology, it is important for decision makers to understand as much about a technology's potential impact and return on investment as possible.

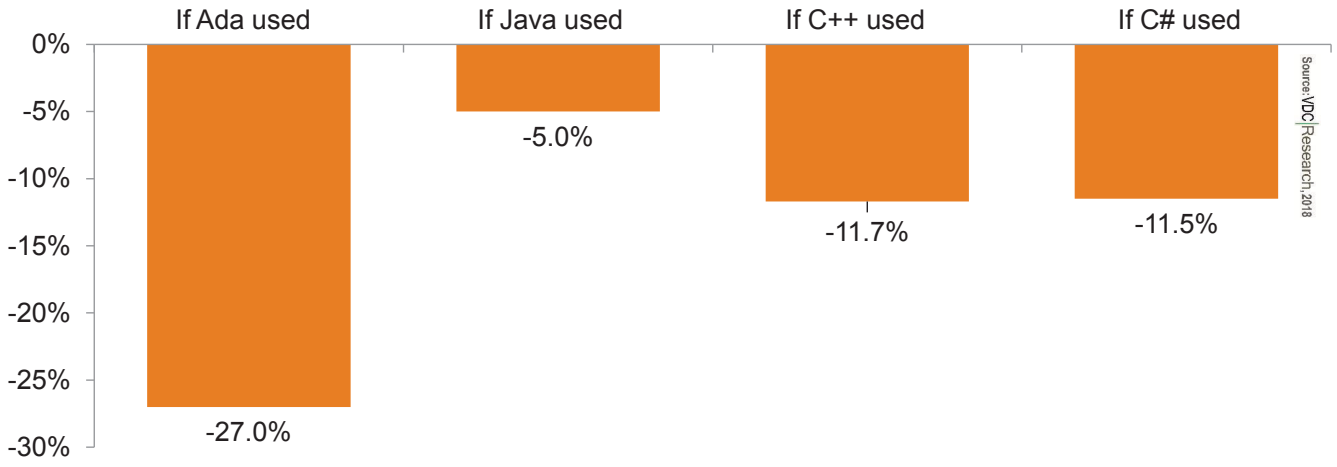
Beyond any organizational or project-specific requirements, certain trends and inferences can be drawn when comparing similar projects. There are a number of different costs that impact the Total Cost of Ownership (TCO), such as Bill of Material costs, development costs, and operational, defect and repair rates. Some of these statistics are strongly influenced by industry and/or demographics, but some technology choices stood out as having significant effect on project outcomes. One such area was software development language choice.

While there are a number of factors that can drive organizations to select a particular programming language – from memory requirements to in-house expertise to existing software assets – there are many projects when companies could or should have those decisions influenced by TCO considerations. Our research shows that software defect rates and costs remained fairly consistent between similar projects, with the exception of Java- and C#-based projects, which tended to have more software defects and patches reported post-deployment. Cost of development varies widely, however, with project length, team size, and average developer cost demonstrating the greatest impact on the results. In order to more effectively measure the impact of technology choices on organizational costs, VDC created its Total Cost of Ownership Calculator. When looking at our most recent research results, Ada was one language that stood out in its impact on various project success metrics.

Example Total Cost of Development Calculations

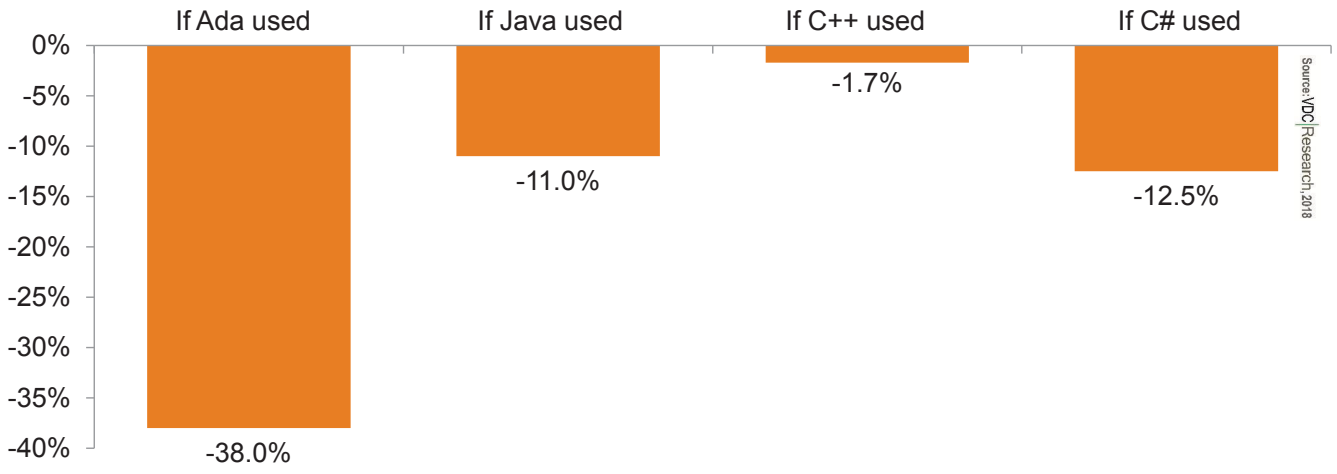
To illustrate the impact of language choice on total cost of development, we have included a few example scenarios for and outputs from our Calculator. The first example evaluates a communications/networking project with an x86 processor that produces 10,000 units and for which C was used for the software development. In this calculation, Ada demonstrates the best possible savings of the highlighted languages, offering a potential 27% reduction in software development costs.

Exhibit 5: Potential Software Development Costs Change per Device, Communications/Networking/x86-based Project (Percentage Change in Costs Versus Current Use of C)



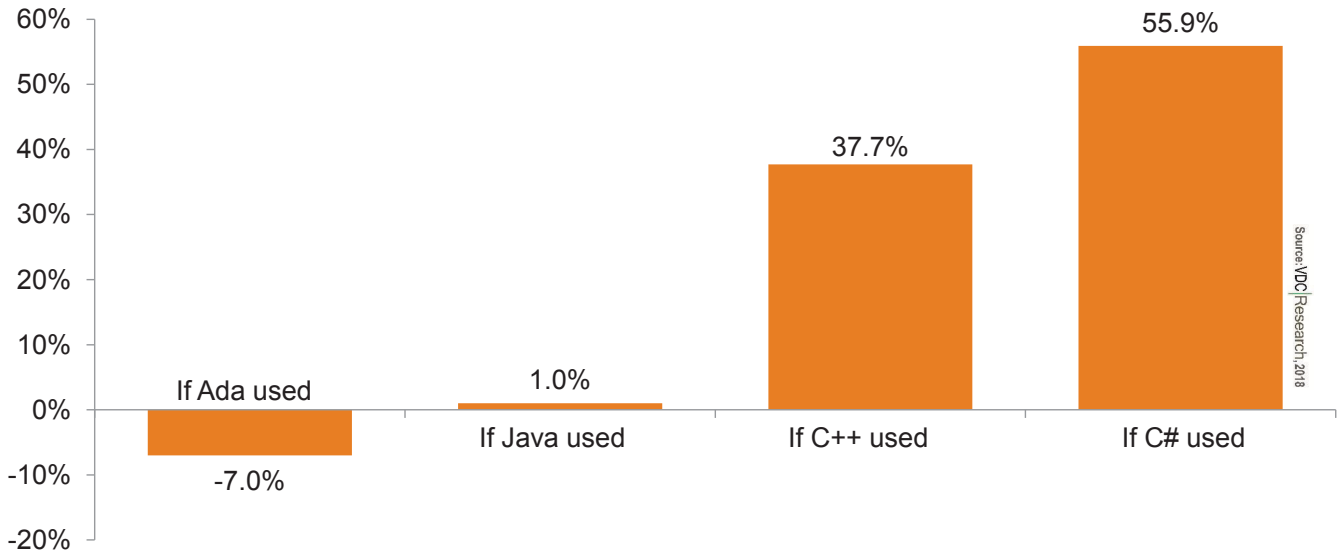
As you may know, aerospace and defense is a longtime user of Ada, but that industry, too, is quickly changing. Designs in that sector were once heavy users of PowerPC processors and are now shifting to ARM, with systems integrators placing greater emphasis on COTS-driven SWaP-C considerations. Engineering organizations have also been placing a new focus on enabling agility through new methodologies and tooling. In the past, methodical and serial development was often mandated based on the belief that such processes would lead to more predictable outcomes. Clearly, in an industry already wrought with long development cycles, understanding what technology choices can drive further improvements in efficiency is critical. When evaluating a 5,000 unit project in this sector using an ARM processor and C as a programming language, Ada again demonstrates the best possible savings of highlighted options, offering a potential 38% reduction in software development costs – which translates to a potential project savings of over $500,000.

Exhibit 6: Potential Software Development Costs Change per Device, Aerospace and Defense /ARM-based Project (Percentage Change in Costs Versus Current Use of C)



Obviously, the results for our cost of ownership calculator vary from vertical to vertical, based on the range of variables and considerations in each sector. And, in some cases, Ada is not the best option. The benefits might not be as pronounced or another language could even be a better choice. In most cases, however, Ada still generally represents a compelling option. For example, in this Automotive/ARM/C project, C is already a rather efficient choice, even if the development cost savings potential may not be high enough to justify changes and risk development organization unease.

Exhibit 7: Potential Software Development Costs Change per Device, Automotive/ARM-based Project (Percentage Change in Costs Versus Current Use of C)



So Why is Ada Saving Organizations Development Cost?

In an effort to better understand the underlying drivers of the cost savings in our TCO calculation, we identified five significant factors contributing to Ada's favorable positioning:

1. Lower Cost Development Resources

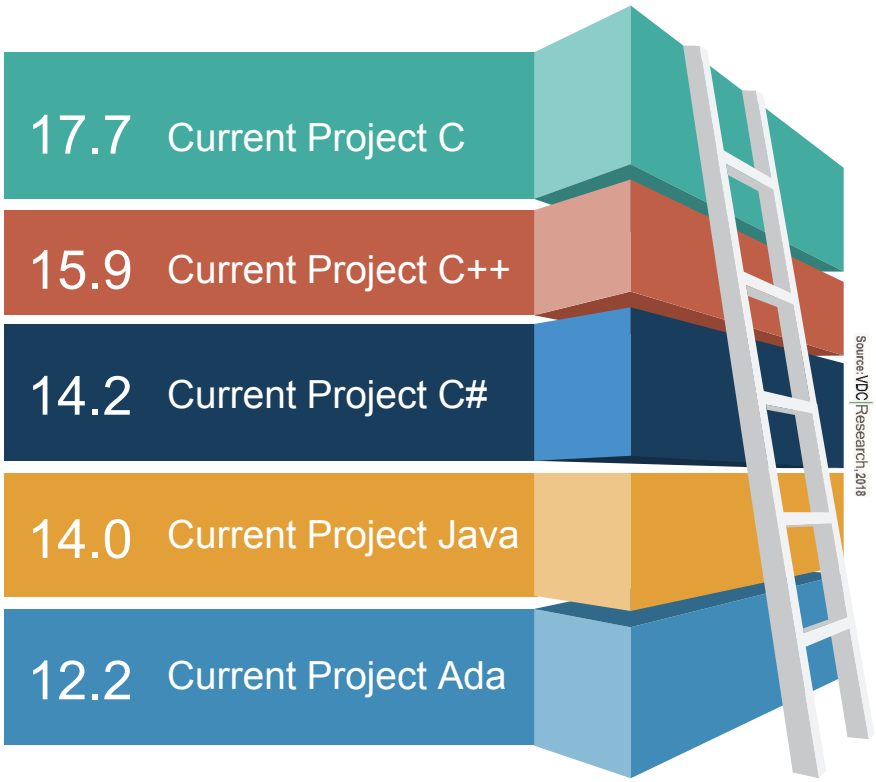
For any organization, controlling labor costs is one of the most important ways to manage costs. This is especially true for embedded systems engineering as highly trained and experienced developers can garner significant wages. To that end, software development language skills and focus have a substantial impact on market wages. For example, the median fully-loaded labor cost for engineers using Ada was ten percent less than those using C and twenty percent less than those using Java ($85,000 versus $95,000 and $105,000, respectively).

At first glance, Ada's relative standing in this regard was surprising. For many years, safety-critical markets and the technologies tailored to them were synonymous with high-cost labor markets. This specialization often led to a significant cost premium. In many cases, this dynamic is still true as seen through comparing labor cost data between consumer electronics and aerospace & defense markets. However, the comparative premiums for engineers with safety-critical backgrounds is not as high as it once was. In some cases, outsized demand for languages that are very popular at the moment, such as C++, are driving down market wages for developers specialized in less common technologies. Additionally, the increased use of languages, such as Java, which are more highly used in IT environments, opened labor markets and depressed salaries overall.

2. Average Ada Projects are Shorter in Duration

Established ecosystems of third-party Ada solutions and libraries help engineers get a jump start on development and focus internal investment and speed time to market. In fact, our research shows that average project in which Ada is used is 30% shorter than the average C project. In fact, Ada projects compared favorably in this regard versus all of the languages highlighted in this analysis (See Exhibit 8). Further reflecting this dynamic, Ada users consistently report a lower percentage of their project costs devoted to labor. In other words, the availability of more commercial solutions reduces the in-house OpEx risk associated with engineers – and thus the potential cost multipliers when projects require more time than expected.

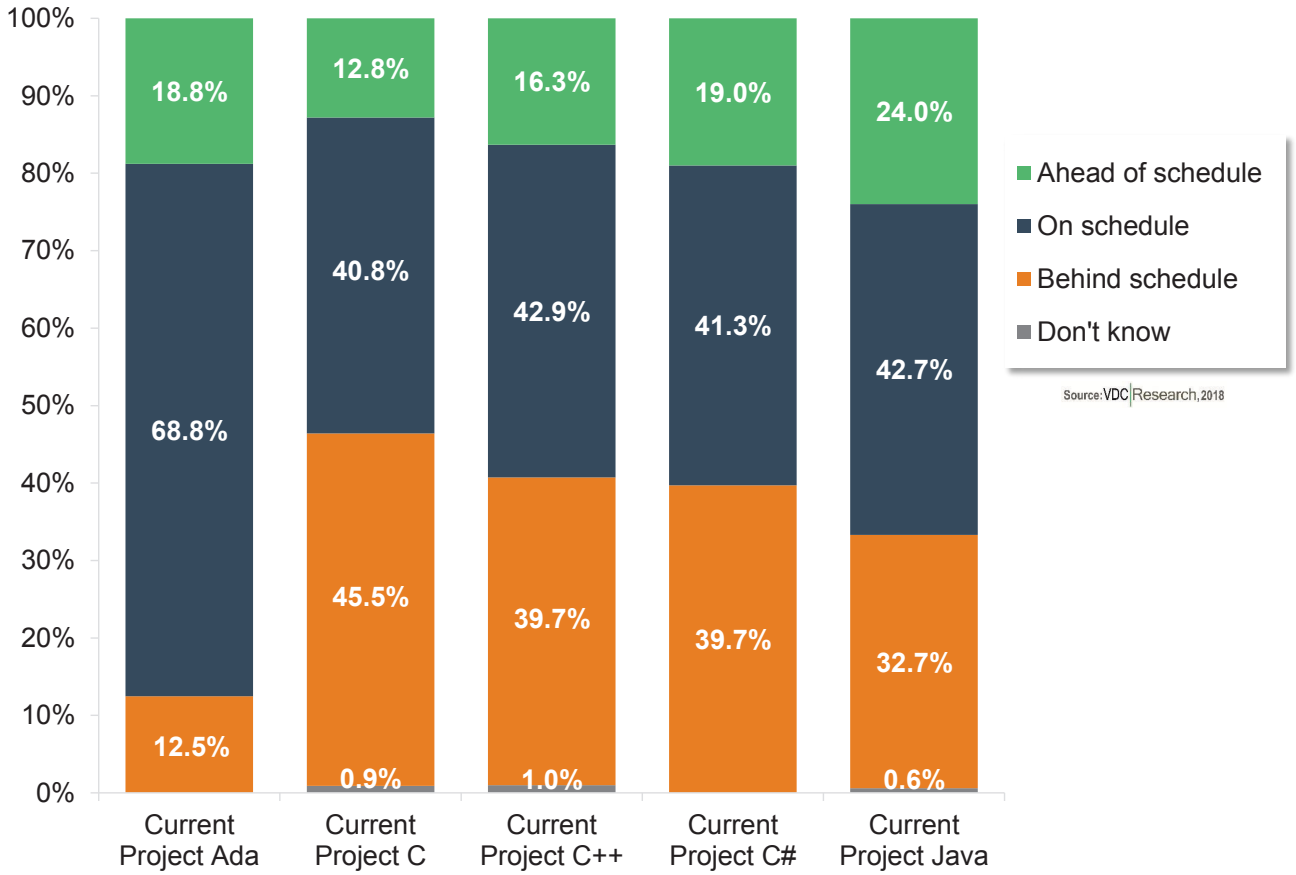
Exhibit 8: Total Project Length in Calendar Months (Time from Initial Specification to Shipment), Segmented by Software Development Language Use (Mean of Responses)



3. Ada Projects are Reported as More Likely to Be on Schedule

Maintaining schedule adherence remains one of the most consistently challenging and frustrating aspects of engineering project management. Although there are a number of factors that can contribute to delays (even if just focusing on the identifiable ones), OEMs often go to great lengths to control schedules through process and technology change. Once again, software development language choice demonstrated a significant correlation with project outcomes. For example, engineers using C were more than three times as likely to report a project behind schedule as compared to those using Ada. In other words, projects using Ada as a software development language not only tend to be shorter but also hold less risk of delay.

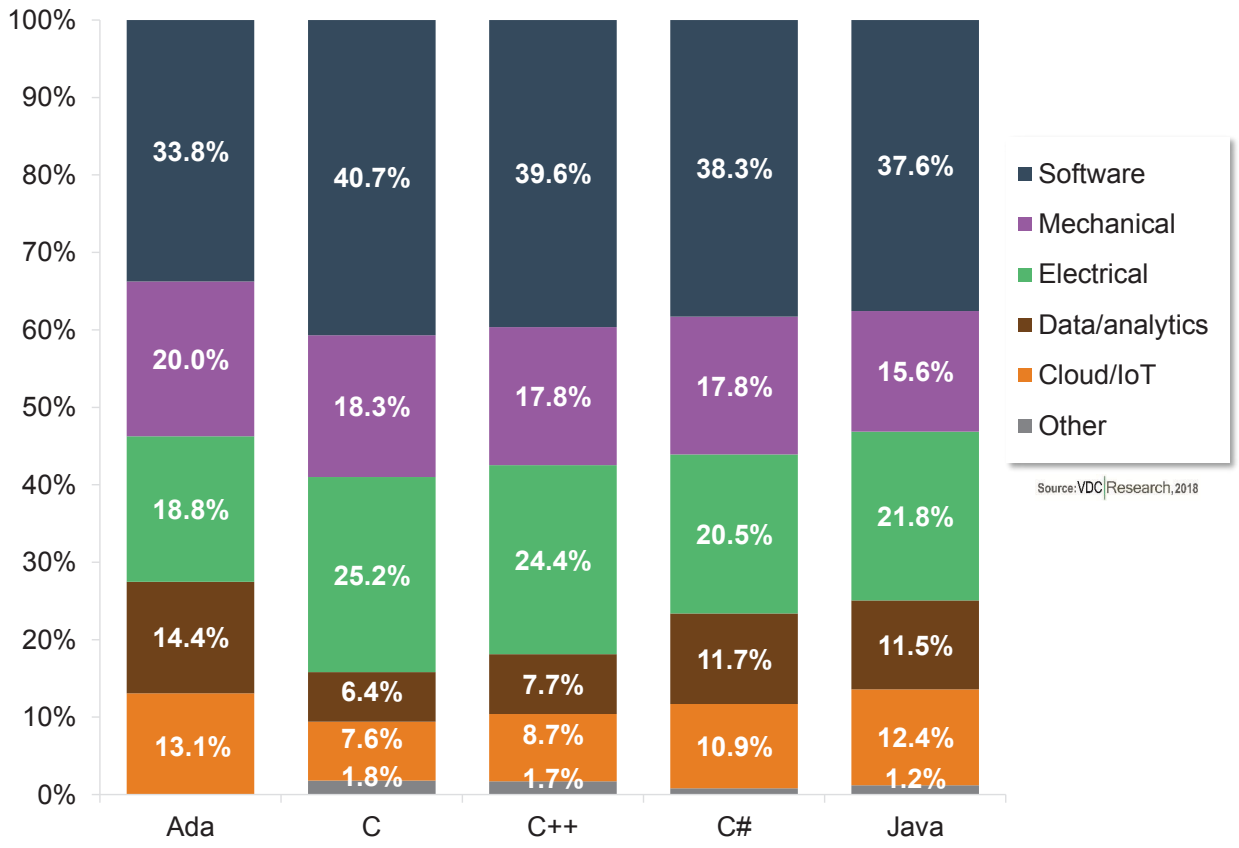
Exhibit 9: Adherence to Current Project Schedule, Segmented by Software Development Language Use (Percentage of Responses)



4. Focused Resource Application

With software the single largest cost center for engineering organizations, it is critical that development teams minimize any unneeded additions in this area. This level of focus can allow organizations to direct labor resources to new areas of differentiation. In fact, Ada users report spending a smaller percentage of their project team's total development cost on software development than do those using other top languages. Furthermore, Ada users report a higher proportion on other potentially differentiating areas like Cloud/IoT and analytics functionality.

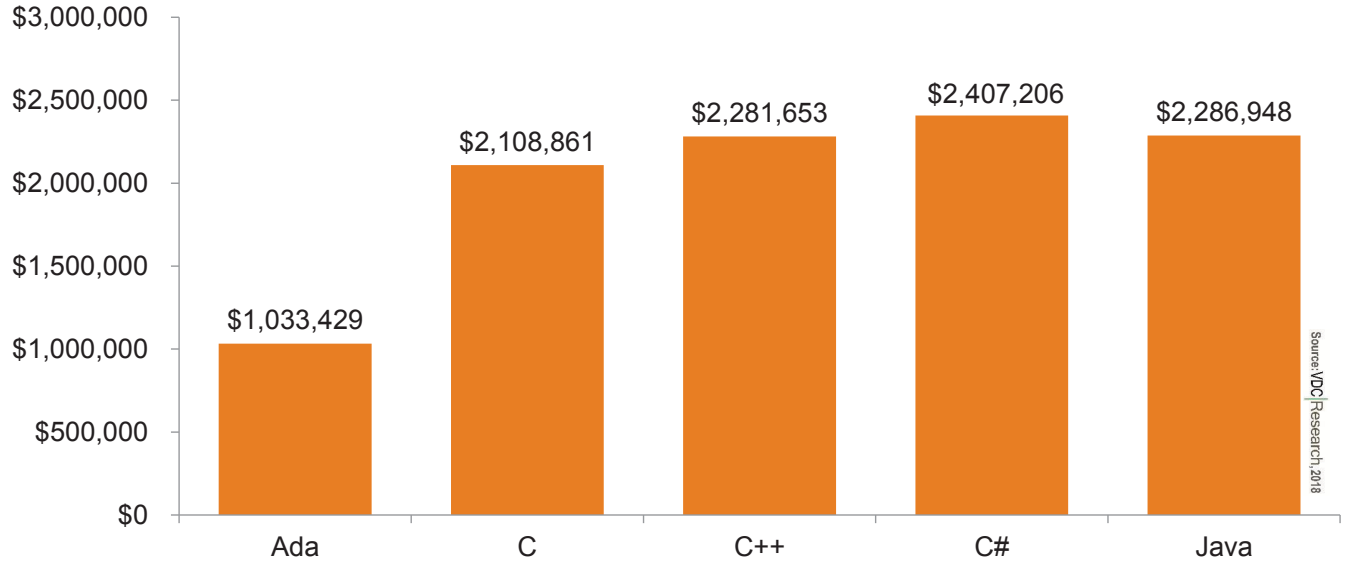
Exhibit 10: Estimated Distribution of Development Costs on the Project, Segmented by Software Development Language Use (Mean of Responses)



5. Software Development Cost Savings

Beyond any single reported statistic related to software efficiency, Ada users cited significantly less expensive projects when reporting their total development costs. In fact, Ada users reported a mean total cost development that was approximately half of that associated with other leading languages. The results similarly hold true in most scenarios in our TCO calculator, which is constructed to normalize outputs base on industry peer averages.

Exhibit 11: Estimated Total Cost of Development for the Project, Segmented by Software Development Language Use (Mean of Responses)



"HOW" ADA CAN HELP

Aside from the quantifiable metrics mentioned above, it is important to understand how Ada can benefit a development organization and lead to those types of results. When evaluating Ada's suitability, we recommend considering four main factors:

1. Reliability

Despite the focus and attention on IoT, reliability remains a critical success factor for embedded systems. In many ways, the deeper interconnection with business results that IoT yields puts reliability at an even higher premium. One of the clearest ways reliability manifests itself is through deterministic and real-time requirements. As shown in Exhibit 2, this remains a leading consideration for many engineers. Ada already has a long history of use in safety-critical markets and has several features beneficial to this arena. For example, the technology can help ease certification documentation through the availability of qualified code generators. Ada's track record in this sector can make it a good choice or compliment to embedded and IoT development projects – nearly half of which still have real-time requirements. Additionally, the use of solutions like SPARK can help further reduce time and cost associated with verification via the inherent facilitation of formal verification and related processes that ensures data and information flows are correct. As such, unlike some other language choices, it is inherently free from run-time errors and all variables are initiated before use, which also helps reduce security exploit vulnerabilities.

2. Reusability

The aforementioned time-to-market pressures necessitate an implicit focus on efficiency – and reuse. Organizations cannot afford to invest valuable man-hours in a technology if they cannot leverage a significant portion of that investment in future designs. Ada was designed with reusability in mind and its semantics reduces implementation dependencies. The fact that it is an ISO standard also helps reduce the introduction of any ambiguity, thus further facilitating future reuse.

3. Flexibility and Scalability

In today's market, organizations need to maximize agility and their ability to adapt to future requirements. Their technology selections must then be rooted in this same principle to maximize speed to deployment and ROI. Flexibility and scalability can be even more valuable in markets requiring certification. Certification requirements typically lead to additional time and costs, so engineering organizations need to identify ways to simplify development and reduce future changes' impact on the time and cost of certification (and recertification). To that end, Ada's hierarchical libraries ease the ways in which a module can be extended without modifying or recompiling the original.

4. Ease of Use

As with any technology, the adoption, developer buy-in, and, ultimately, the benefits will be greatly affected by ease of use. As discussed above, Ada's strict programming rules versus other languages can prevent coding errors that would otherwise lead to additional time during debug/verification. This feature also thus improves ease of use by requiring less consideration and planning needed around specific bits and memory size, which would otherwise necessitate the implementation of various coding mechanisms in C. Obviously, a big factor in ease of use is also catering to the skill sets and experiences related to incumbent or more commonly used technologies. To this end, the Ada community developed extensive libraries and literature targeted at traditional C and C++ users to learn Ada (not to mention traditional Ada's inclusion in many introductory CS curriculums). In fact, Ada is rather unique in that it has interfaces with other languages already built in. This focus on improving experiences for a range of developers has become even more valuable in today's environment, given the increasing frequency of polyglot development in current projects.

CONCLUSION

Never before has there been so much need and opportunity to redefine embedded system engineering practices. The Internet of Things has catalyzed widespread changes in system feature sets and business process requirements. Unfortunately, too often these requirement changes have not been paired with sufficient reevaluation of development technologies and practices. Often the need to satisfy specific functionality requirements leads an engineering organization to focus primarily on Bill of Material components. Other times, tight time-to-market windows drive organizations to default to the status quo and opt for reuse of incumbent solutions.

Development technologies have a profound impact on one of the largest and most variable costs associated with embedded system engineering-labor. At a time when on-time system deployment can not only impact customer satisfaction but access to services revenue streams, engineering team efficiency is at premium. Our research showed that programming language choices can have significant influence in this area, leading to shorter projects, better schedules and, ultimately lower development costs. While a variety of factors can influence and dictate language choice, our research showed that Ada's evolution has made it an increasingly compelling option for engineering organizations, providing both a technically and financially sound solution.

ABOUT THE AUTHOR

Chris Rommel

Chris Rommel is responsible for syndicated research and consulting engagements focused on development and deployment solutions for intelligent systems. He has helped a wide variety of clients respond to and capitalize on the leading trends impacting next-generation device markets, such as security, the Internet of Things, and M2M connectivity, as well as the growing need for system-level lifecycle management solutions. Chris has also led a range of proprietary consulting projects, including competitive analyses, strategic marketing initiative support, ecosystem development strategies, and vertical market opportunity assessments. Chris holds a B.A. in Business Economics and a B.A. in Public and Private Sector Organization from Brown University.

Contact Chris:
crommel@vdcresearch.com

ABOUT VDC RESEARCH

Founded in 1971, VDC Research 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 Natick, 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.

VDC Research
Insights for the Connected World