VDC Research
Insights for the Connected World

2016 | IoT & Embedded Technology

VDC | White Paper

Using Java to Control IoT Development Costs

Exclusive License to Distribute:

Oracle

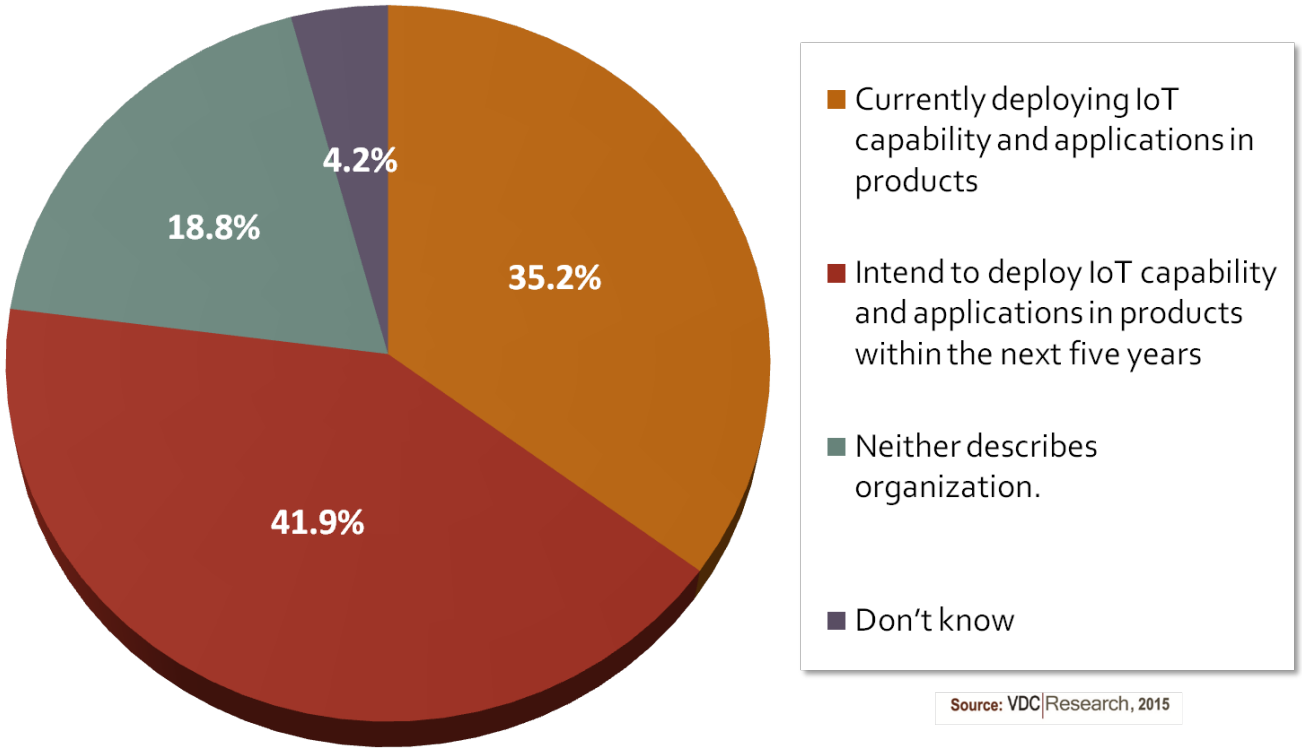
By Chris Rommel, Executive Vice President

Introduction

The Internet of Things (IoT) is now much more than hype. Expanding connectivity between devices and enterprises has recast traditional product development, operational, and business model strategies. While machine-to-machine connectivity and remote system monitoring are not new, the IoT is providing OEMs and enterprises a range of new or enhanced capabilities, such as the ability to provision content and functionality, deliver concierge or location-aware services, and predict system maintenance requirements based on operational data patterns. New revenue opportunities and partnerships can be derived from these enhanced services as richer real-time data insights and contextual intelligence rapidly change engineering roadmaps and solution selection.

Exhibit 1: Adoption of IoT Capability and Applications

(Percent of Respondents)

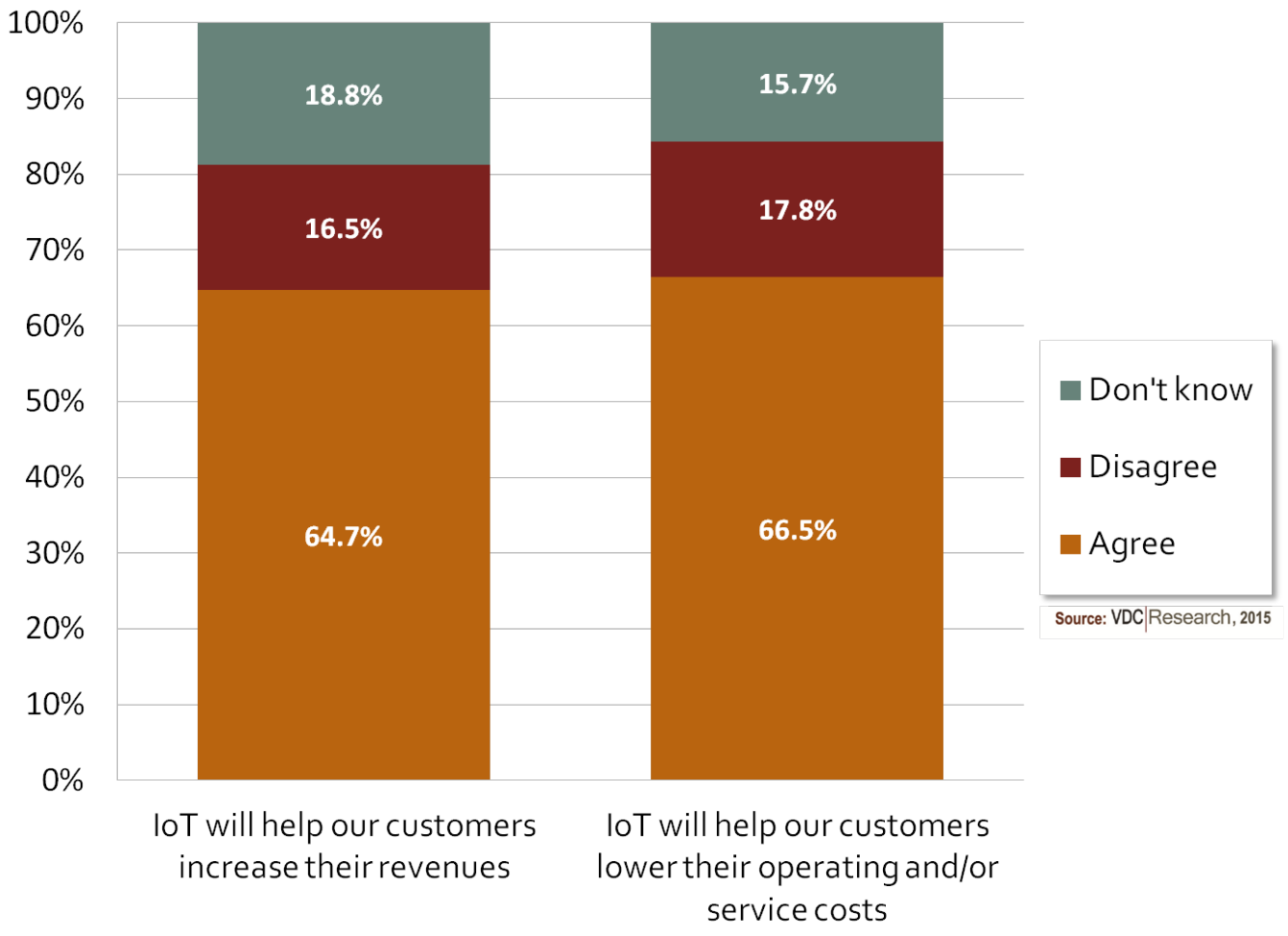


Source: VDC Research, 2015

Already more than 35% of engineers are deploying IoT capabilities with their current products under development. Moreover, nearly two-thirds of engineers believe that their customers will be able to leverage the IoT to increase revenues and/or lower operating costs. In many, if not all, cases, IoT products will require integration with enterprise and cloud-based services, require partner and customer modifications, and support provisioning of new capabilities after the device has been deployed – resulting in a more complex and extended product lifecycle. These market shifts and new functionality requirements, however, can cause engineering organizations to bear a significant integration burden and large potential development cost increase if they are not able to quickly adapt and identify more efficient solutions and methodologies.

Exhibit 2: More Engineers Believe in Value of IoT

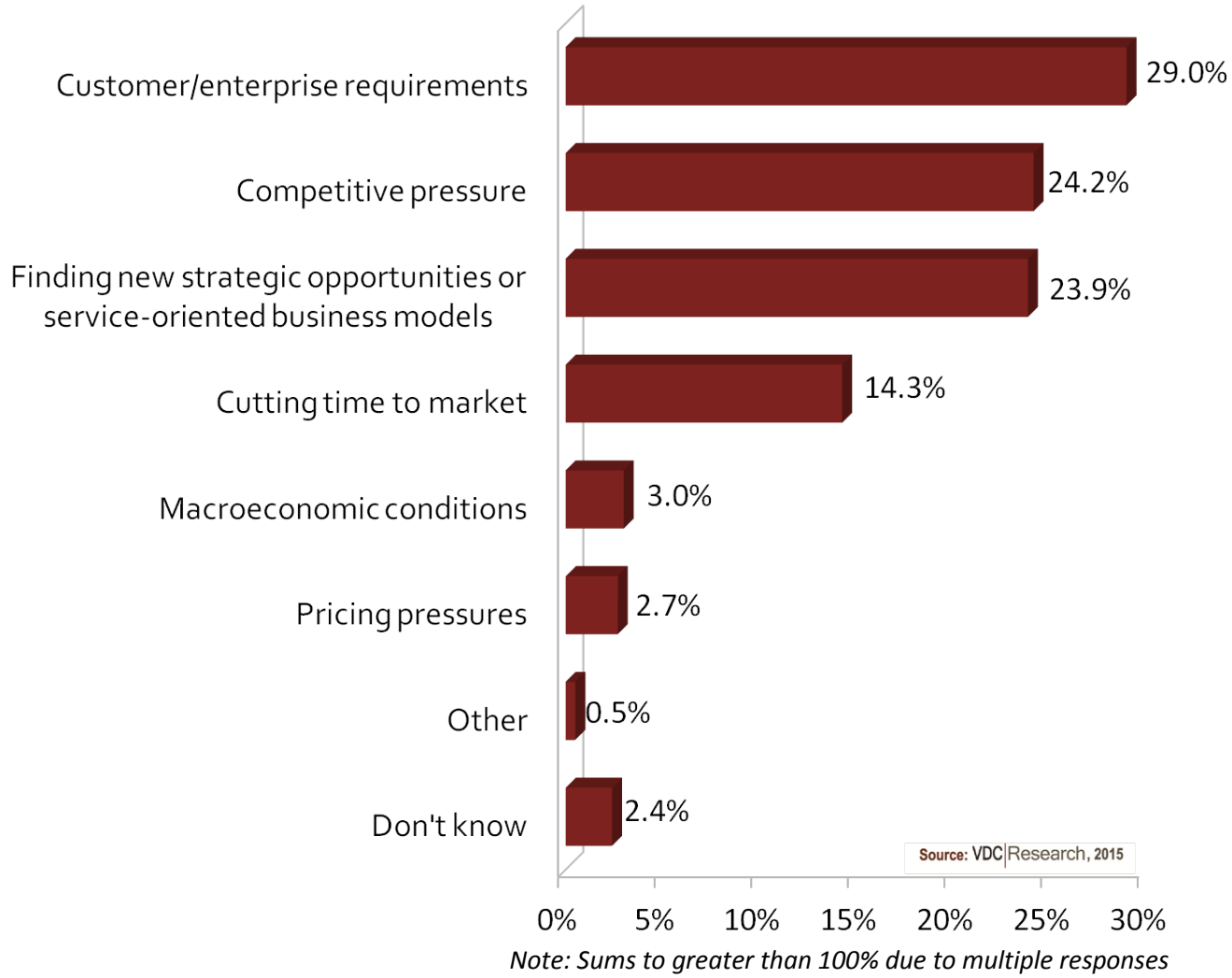
(Percent of Respondents)



IoT Forcing Reevaluation of Development Economics

While new IoT revenue opportunities are enticing and strategic catalysts for OEMs on their own merit, customer and competitive pressures are forcing OEM's development choices. These IoT development imperatives are moving many organizations into uncharted territory where they lack the resources and/or expertise to efficiently bring solutions to market. In addition to traditional software development challenges, engineers must also contend with new functionality and connectivity requirements that are critical to competing in the IoT domain. Additional considerations, such as security, identity, and privacy, which were previously much less of a concern, now must be addressed in a connected context. In fact, surveyed engineers recognized development resources and cost as the largest challenges for IoT solution development. These added challenges facing development organizations serve only to amplify issues already impacting product development schedule and quality.

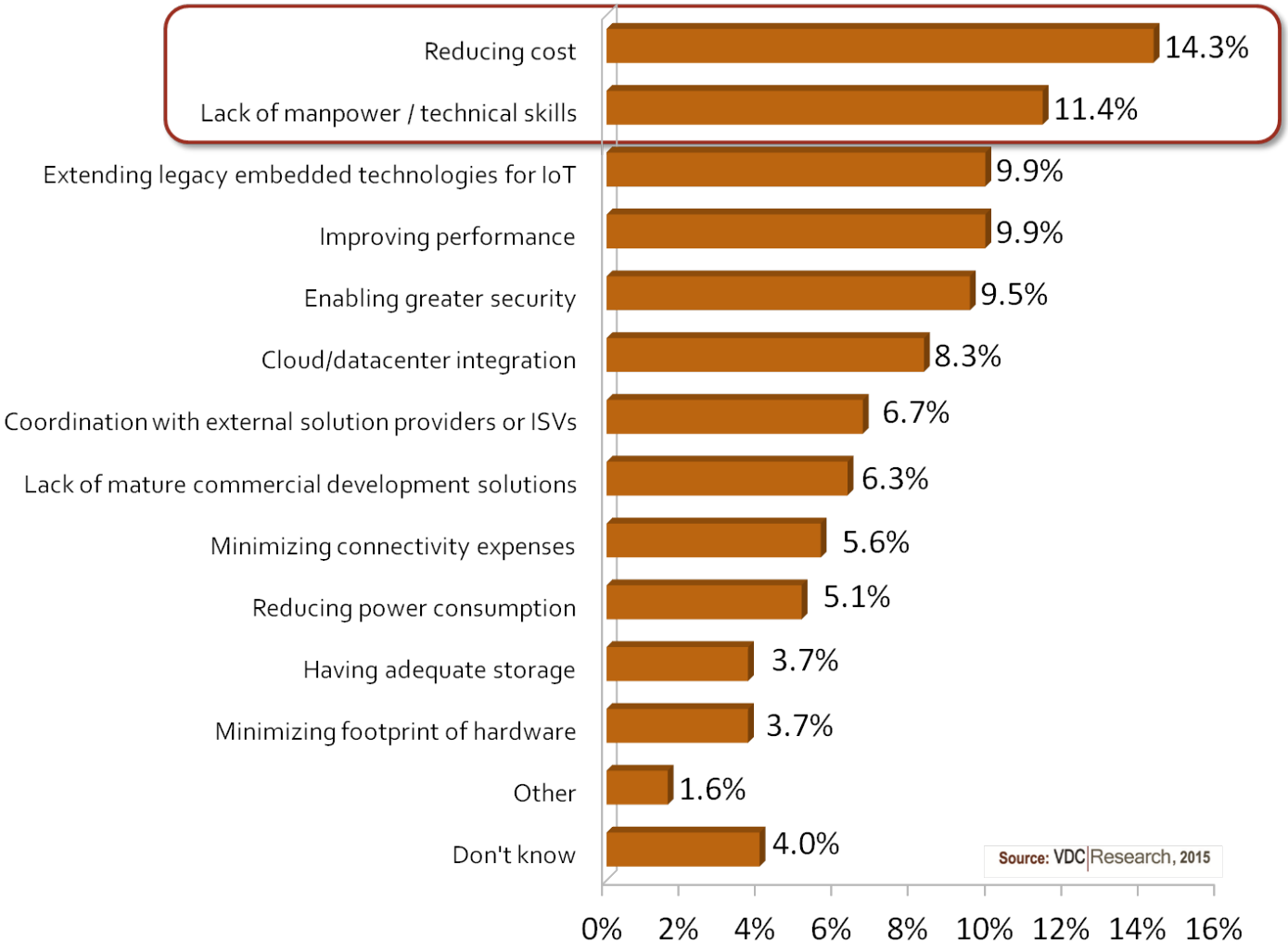
Exhibit 3: Primary Factor Driving Organizations towards Utilizing and/or Providing IoT Solutions
(Percent of Respondents)



Note: Sums to greater than 100% due to multiple responses

Exhibit 4: Biggest Overall Challenge in Developing IoT Solutions

(Percent of Respondents)



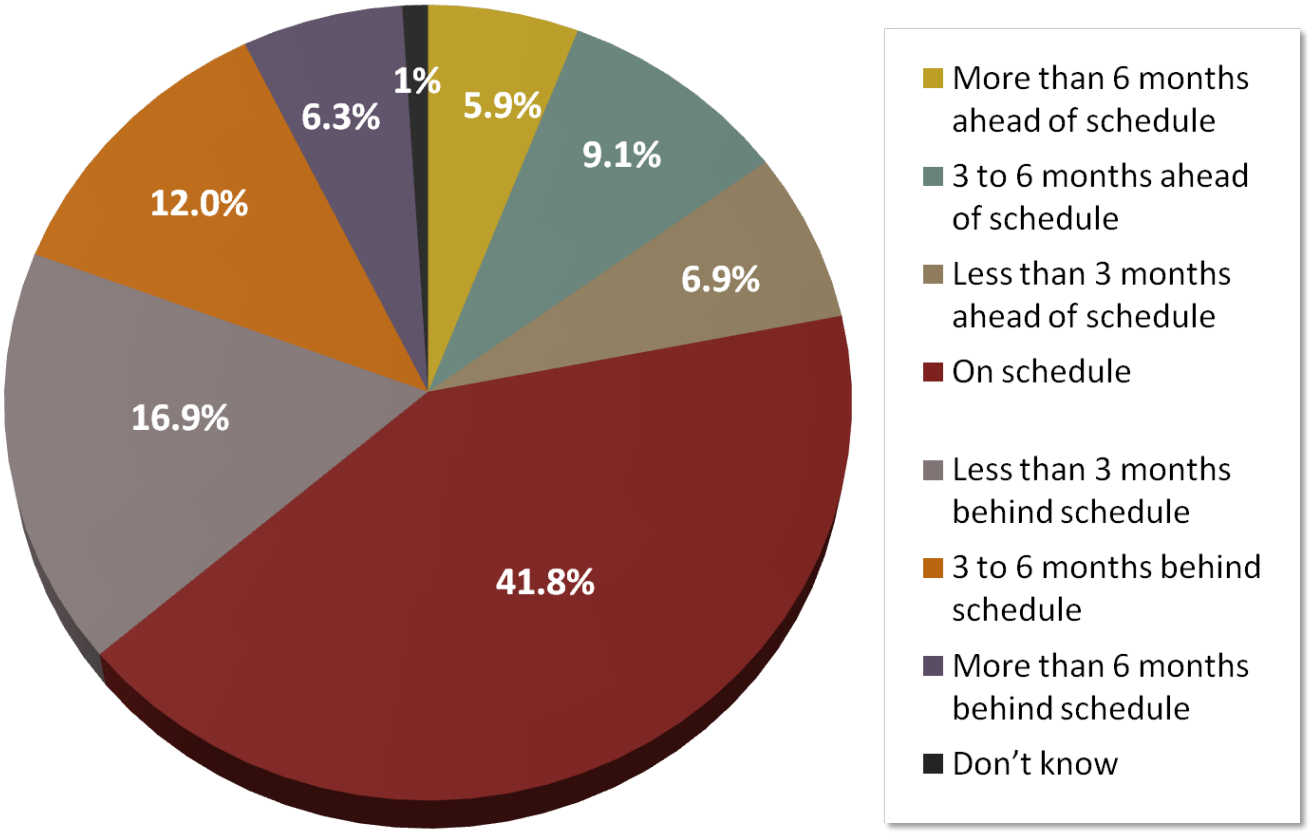
Missed or delayed schedules are not new challenges to product development organizations. Now, however, IoT-driven complexity exacerbates traditional time-to-market challenges as engineering organizations face new development and capacity challenges. For example, not only must developers contend with increased software content creation requirements, but they must also mitigate new connectivity-driven risks like security, over-the-air updates, and solution governance.

In our most recent survey, over one-third of engineers reported that their projects were behind schedule. While a sizeable figure, performance against this metric has improved over recent years. In fact, 71% of respondents indicated improvement in schedule adherence versus past projects. Part of this improvement is from the adoption of new development technologies and methodologies. The top-level improvement, however, masks some of the underlying project management and execution problems plaguing engineering organizations. For example, 18% of respondents reported projects more than three months behind

schedule. Given an average development project length of only 12 months, those delays translate into schedule misses between 25% and 50% or more and, consequently, potentially profit-dooming labor cost overruns. And it is not getting easier; the magnitude and complexity of IoT-driven requirements will necessitate additional organizational flexibility to maintain and/or improve upon current schedule adherence levels.

Exhibit 5: Adherence to Current Project Schedule

(Percent of Respondents)



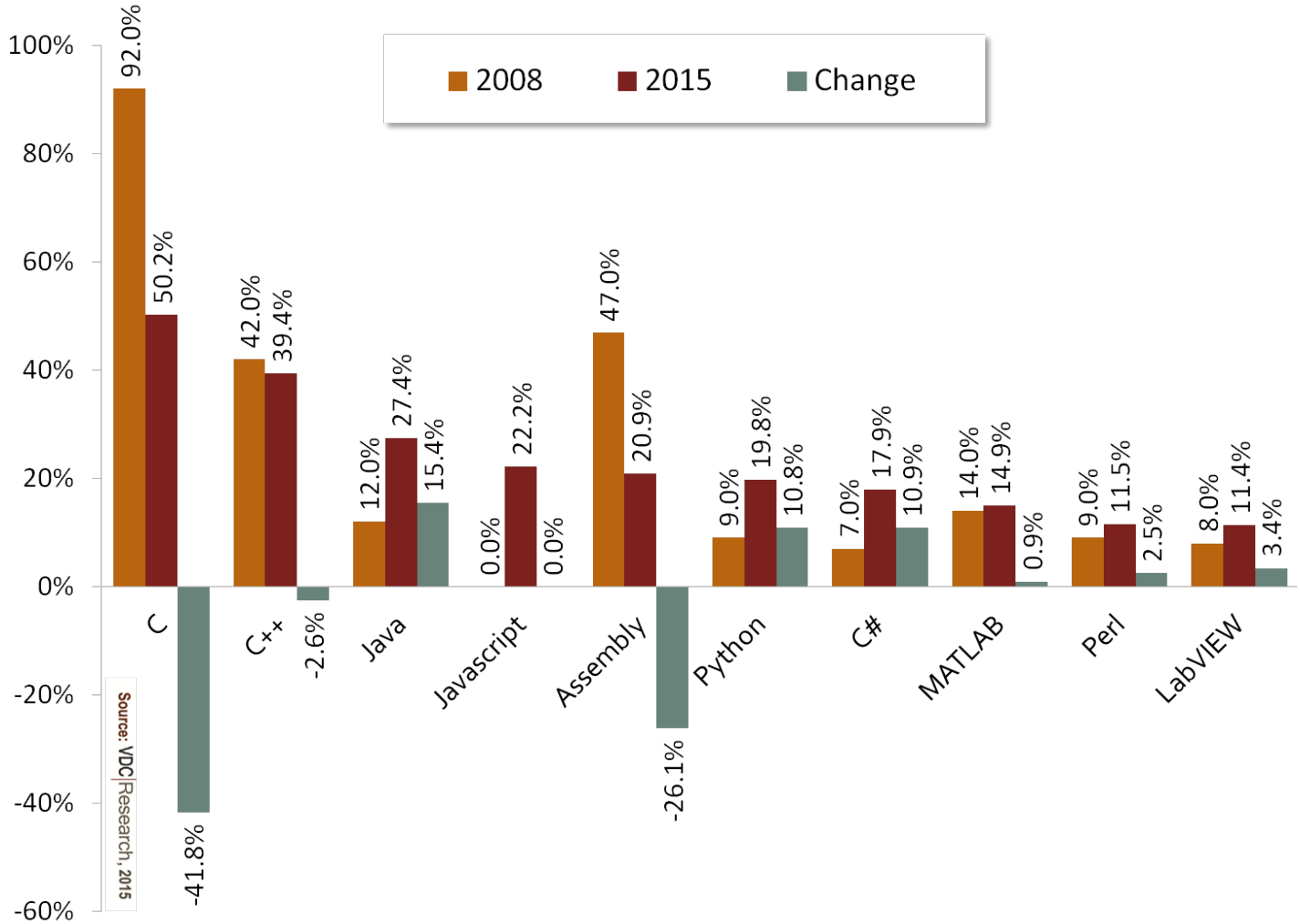
Source: VDC Research, 2015

IoT Reinforcing Adoption of New Software Development Technologies and Languages

The breadth and speed of innovation required to remain competitive in the IoT is forcing many engineering organizations to reevaluate existing technologies and plan further wide-ranging changes in the future. In addition, engineering organizations must consider how their new products will support and satisfy IoT business goals that now include: increased operational data collection, delivery of new connected services, and interfacing with existing operational and IT infrastructure. In many cases, the evolution of IoT system functionality requirements necessitates new technology throughout the device software stack from operating systems, to security, to connectivity middleware, and beyond.

Exhibit 6: Languages Using to Develop Software on the Current Project

(Percent of Respondents)



Note: Sums to greater than 100% due to multiple responses. Not all response options shown.

An important way these market forces manifest themselves is through changes in software programming language/platform choice. Until recently, embedded software development was predominantly conducted in C and other lower-level languages. Today, however, embedded engineering is no longer synonymous with small footprint, fixed-function, single purpose devices. Today's devices need to perform multiple functions and rely on multiple semiconductors and/or cores for the necessary processing power. The requirements for high reliability and determinism that perpetuated the use of C are still present, but the communication, analytics, and security-driven IoT requirements drive adoption of object-oriented and higher-level languages such as C++ and Java.

From a complete platform perspective, the IoT has accelerated the rate of innovation and emergence of higher-level feature set requirements in embedded systems. The benefits of highly specialized components and hardware have now been surpassed by the flexibility offered by software and need for faster time to market. OEMs are now searching for new ways to abstract growing hardware complexities in favor of flexible platforms that offer broader access to off-the-shelf connectivity and middleware components, and help to speed development. Furthermore, time-to-market pressures have driven many organizations to seek more reuse of pre-existing and newly developed software assets. Developing reusable assets across hardware platforms reinforces the value proposition of object-oriented languages with extensive libraries such as Java. In many ways, this evolution has paralleled that undertaken by the general IT ecosystem over the past two decades, which also faced increasingly dynamic architectures, higher level of data exchange, and strong security/identity. It is no surprise that the dominant language used across IT is Java.

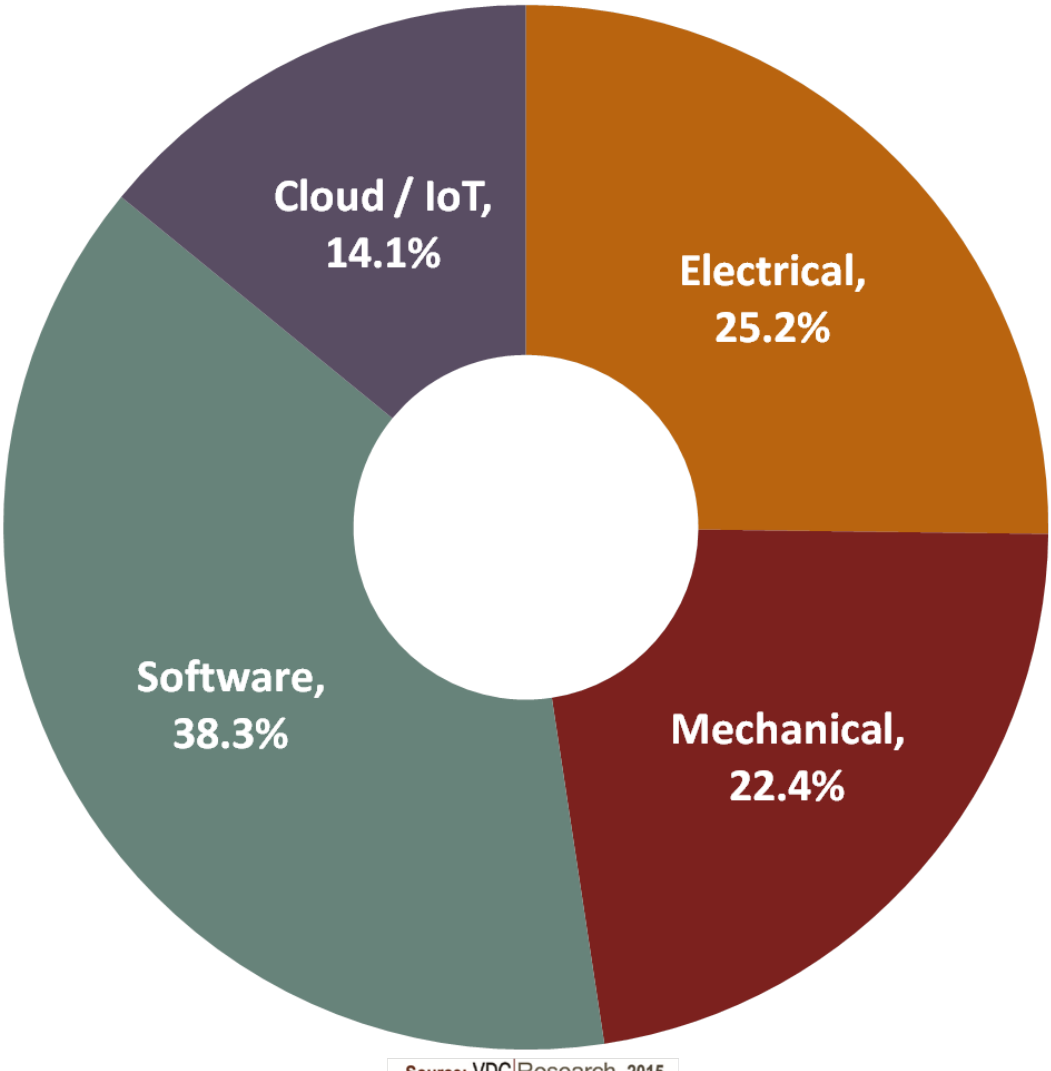
The traditional barriers and negative perceptions about using Java in embedded deigns are also wearing down. The rapidly growing processing and memory resources available in lower cost and/or smaller footprint hardware has enabled OEMs to adopt robust software stacks capable of addressing the sophisticated functionality goals within the IoT. The evolution of other complementary technologies for the embedded space, like hypervisors and heterogeneous processor architectures, provide OEMs with new ways to enable real-time components to coexist with Java-based components. This co-existence offers the best of both worlds; one can simplify and optimize a real-time control loop in C code, while retaining the rich feature set for communication, security, and displays that are better addressed by Java. Enhancements to Java's embedded portfolio in recent years and an expanding ecosystem of semiconductor vendors embracing Java now present a more complete solution that scales down to small ARM-based devices up to x86-based datacenters. These technology and ecosystem advancements, combined with the already growing amount of Java deployment in the embedded market, should further reinforce the platform's growth trajectory in the future.

Reevaluating Total Cost of Development in the IoT

The IoT-fueled challenges facing many organizations have sparked a need to reevaluate the organization's resources and incumbent technologies to ensure that they are selecting the options that can not only help them differentiate, but also unearth additional development efficiencies and cost savings. The cost and value centers for many organizations are already changing. Engineers expect lines of software code to grow at nearly 18% for their next projects, far eclipsing the organic rates of growth in the embedded engineering community. As the lines of code increase, so will the complexity, thus the IoT will amplify many of the existing challenges already facing development organizations. With these changes underway, it has become critical that OEMs tap into new partner ecosystems and resource pools to identify new talent and lower cost resources if available.

Exhibit 7: Estimated Distribution of Development Costs on Current Project

(Average of Responses)



Source: VDC Research, 2015

Calculating the Total Cost of Development

VDC's TCO Calculator

In July 2015, VDC completed data collection for its 2015 IoT & Embedded Engineering Survey, with more than 800 respondents from a broad range of industries. Those results were used to build a total cost of ownership and development calculator.

The Total Cost of Ownership Calculator uses project statistics such as: devices per project, Bill of Material costs, distribution of development costs by engineering discipline on current project, reported total cost of development, 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 account for results for projects using the same processor architecture, same language, and same vertical market.

As discussed above, the IoT and embedded market is incredibly complex and heterogeneous. There are a large number of factors that drive selection of specific hardware and software components, with influence coming from current requirements, as well as capital and IP-development investments made during past projects.

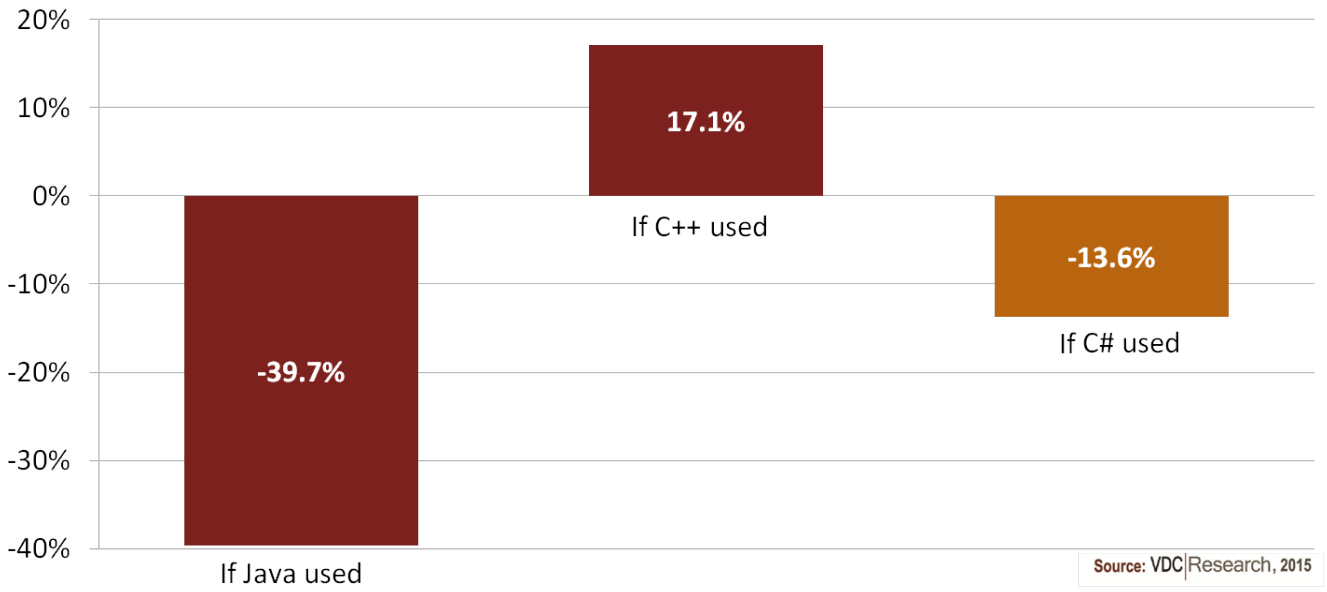
Despite these organization- and project-specific requirements, certain trends and conclusions can be drawn when comparing similar projects. Our survey of over 800 engineers has enabled us to identify some of these common tendencies and patterns in order to construct a tool to help organizations assess the total cost of ownership of their product under development. There are a number of different expenses that impact the total cost of ownership, such as one-time payments for the Bill of Material and initial development costs, as well as those costs that extend beyond product deployment, such as ongoing operational costs, software updates, and defect repair rates. In an effort to identify the variables most impacting development expenditures, our research showed that software defect rates and costs remained fairly consistent between similar projects. The cost of development varies widely, however, with project length, team size, and average developer cost having the greatest impact on total cost. Within this labor cost assessment, the growing significance of software to end-product functionality and differentiation places a great importance on organizations' ability to identify technologies that could lead to greater engineering efficiencies. One such technology that displayed potential software development cost savings versus alternatives for certain projects was Java.

Java Can Save up to 40% on Software Development Costs for an ARM Project

Consider a project expected to produce 1,000,000 units using an ARM based architecture in which C was reported as a primary language for software development. According to VDC's Total Cost of Ownership Calculator, the use of Java could have contributed to software development cost savings of approximately 40% as compared to a C-based design. Those same projects also tend to produce overall engineering costs over 30% less than C.

Exhibit 8: Potential Software Development Costs Increase per Device

(Percent Change in Costs)



Estimated development costs were consistently higher for C than Java for many project types – even for safety-critical industries like medical. For example, in a medical device project using an ARM Cortex A-based processor, an engineering organization would have saved 17.6% on software development costs when using Java. Furthermore, our research showed that potential software development cost savings from Java extended beyond ARM-based designs. When comparing x86 projects, other languages like C++ also demonstrated cost savings versus C – and for some industries, like Industrial Automation and Control, the cost savings were consistent regardless of architecture.

Exhibit 9: Potential Software Development Costs Increase Per Device - Medical/ARM Cortex A-based project

(Percent Change in Costs)

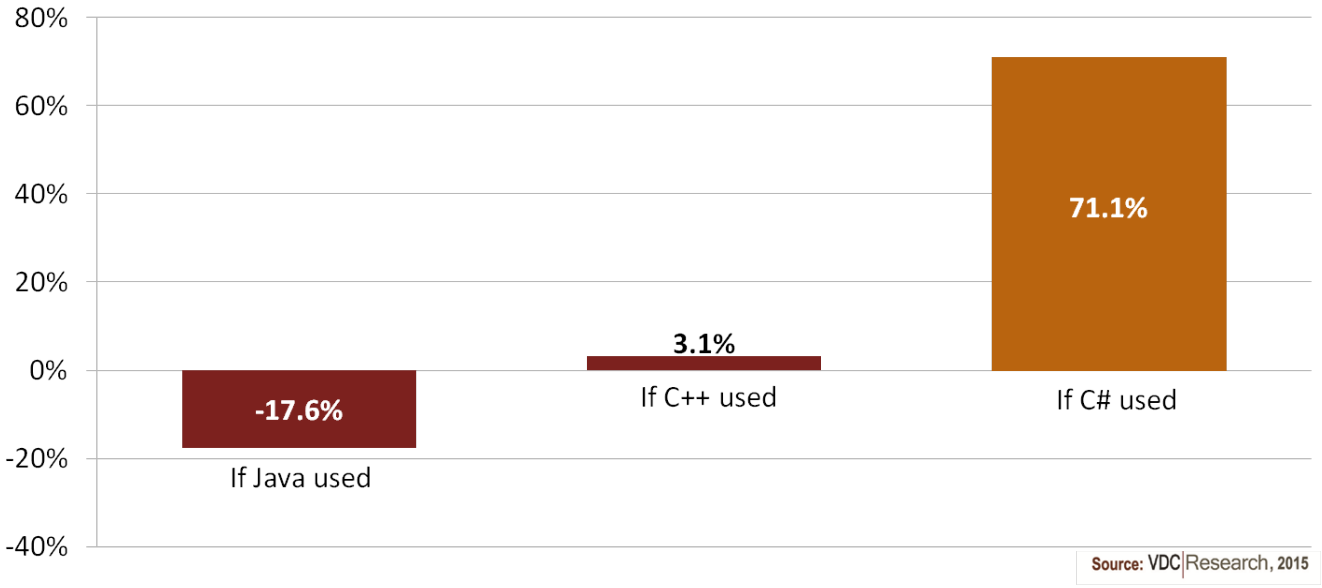


Exhibit 10: Potential Software Development Costs Increase Per Device - Industrial/ARM-based project

(Percent Change in Costs)

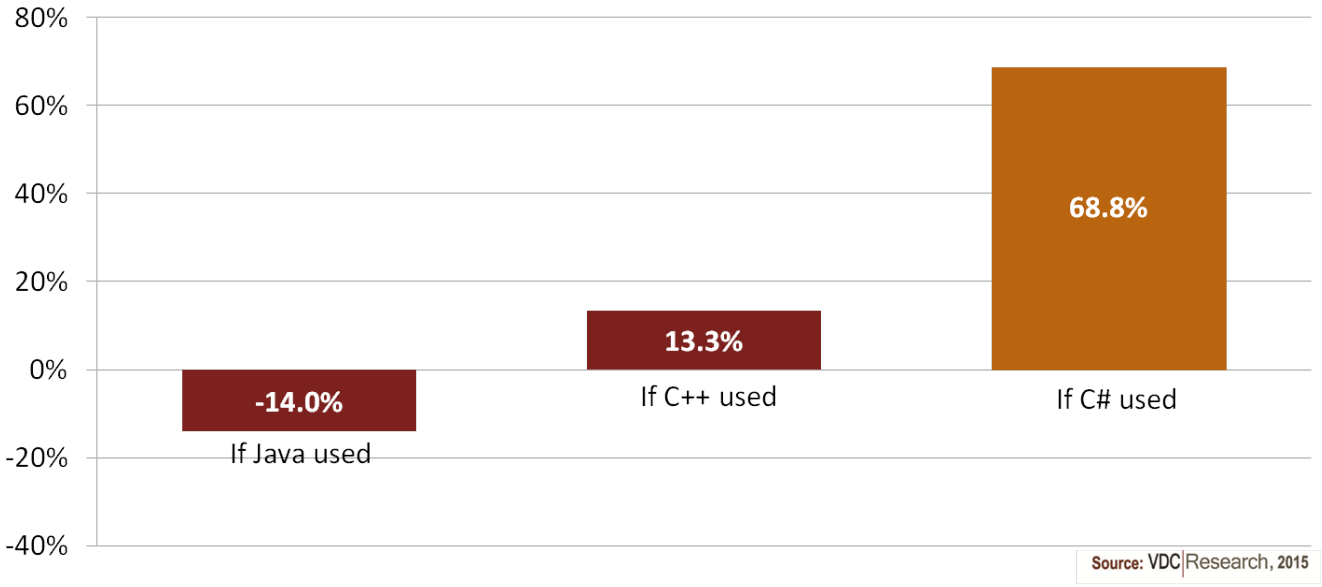
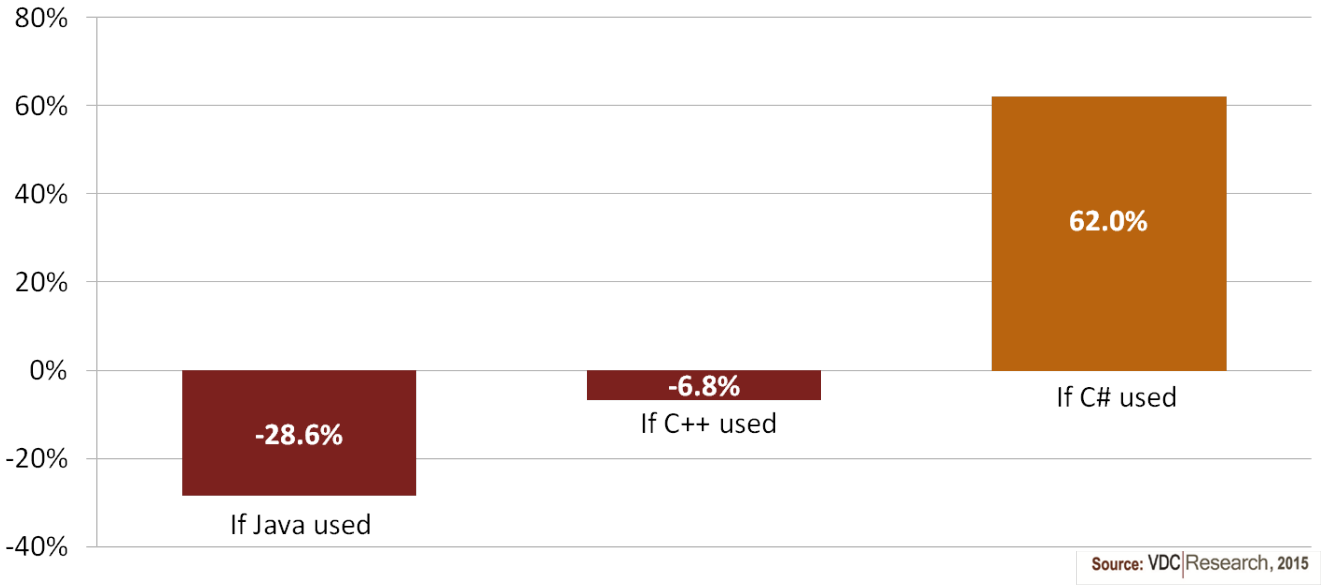


Exhibit 11: Potential Software Development Costs Increase Per Device - Industrial/x86-based project

(Percent Change in Costs)



So How Does Java Save Development Costs?

Although every development project is different, our research highlights four main reasons Java often saves engineering organizations money, as compared to other software development languages traditionally used for embedded engineering.

1. **Lower Cost Development Resources**

The increased level of software content creation places a great deal of stress on in-house development resources. As a result, organizations are looking for new ways to pragmatically expand their capacity. In the past, organizations sought lower labor markets to outsource software development. Today, however, some of those localized cost advantages have started to erode, while, at the same time, the growing importance of software to end product differentiation has given OEMs less reason to outsource it. As such, it is becoming increasingly important for engineering organizations to look for ways to identify lower-cost local resources. The selection of programming language is one such way engineering managers can opt in to a lower cost labor pool.

Today, there is a much broader community and resource pool with Java experience, as compared to C, given Java's wider use in general IT software development. The entire traditional embedded ecosystem contains just over one million engineers, whereas there were over 9.3 million Java developers worldwide in 2014. This broader IT resource pool translates into cost savings for engineering organizations. The median fully-loaded cost of an engineer working on a project where Java is used is 10 percent less than that for the industry overall ($95,000 versus $105,000). More narrowly used languages, such as C, can be valuable for certain optimized software design, but the specialized skill sets of the smaller community of C developers come at a premium.

2. **Average Java Projects Are Shorter in Duration than Those for C and C++**

Exhibit 12: Total Project Length in Calendar Months (Actual Time from Initial Specification to Shipment), Segmented by Software Development Language Use

(Mean, Excluding Responses from Outlying Deciles)

Java	C	C++	C#
11.8	17.0	16.1	12.2

Source: VDC Research, 2015

Project length impacts cost of development in much the same way that the number and cost of engineers can impact development cost. Despite the fact that Java is often used in more complex software development projects, engineers using the language reported project durations 30% shorter than those using C. In addition to its place as an indicator of additional cost, project length also impacts the top-line for organizations (a factor not directly accounted for in the TCO calculator). As more organizational revenue is tied to post-deployment content and services, project length can more directly impact revenue opportunities and risk missing market windows. As a result, the ability to choose platforms that enable flexibility in resource selection, access to more complementary, third-party software, and that offer better track records of predictable development schedules will increasingly be placed at a premium.

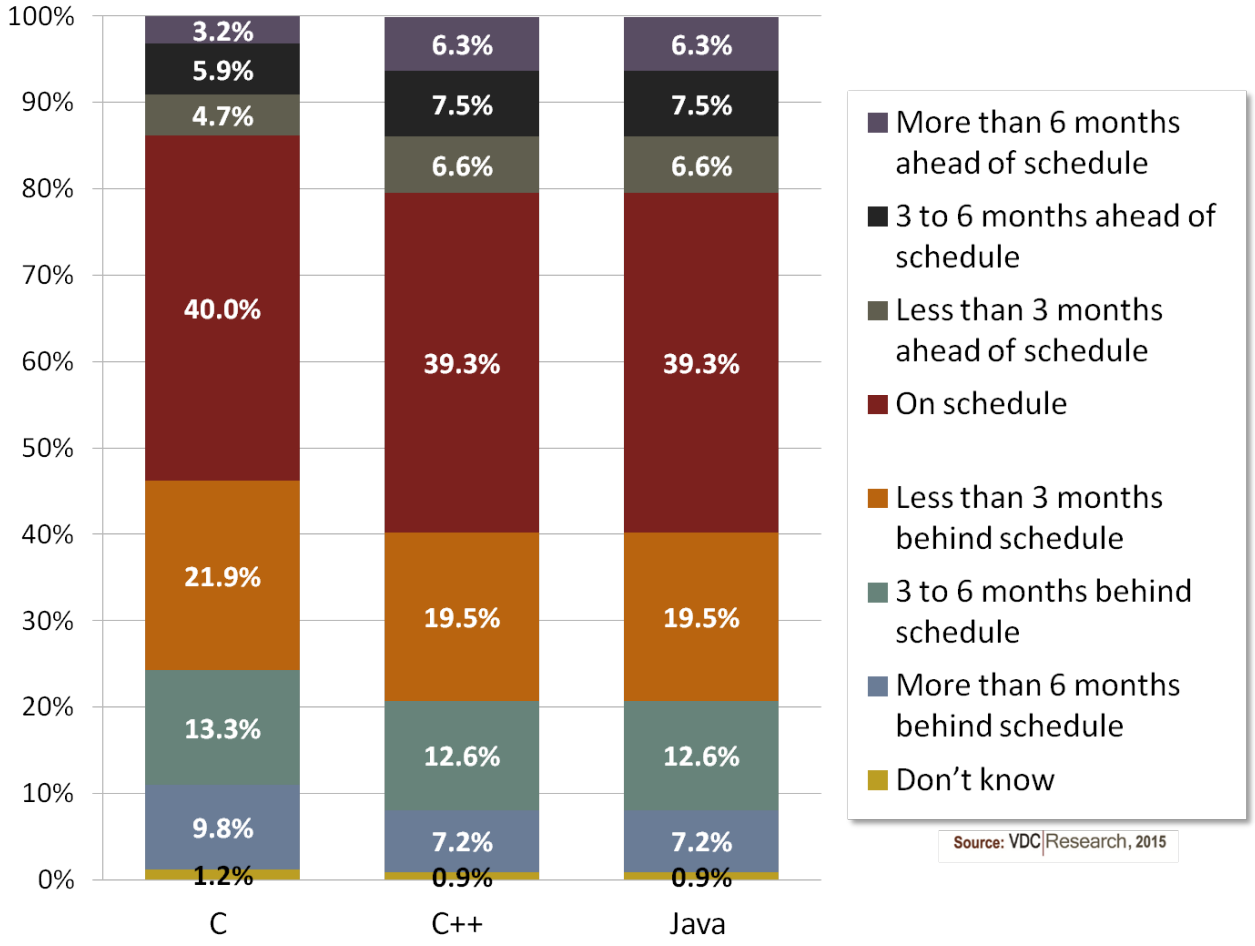
3. Java Projects Are More Likely to Be on Schedule

As discussed above, meeting a schedule remains a consistent issue for engineering teams. Delays lead to cost over-runs, missed market windows, and resource bottlenecks that impact future projects. Beyond simply improving efficiency and speeding time to market, schedule adherence can be improved through more accurate project planning.

There are far too many variables that can impact a development cycle to reasonably expect elimination of all delays. However, reducing their occurrence and/or magnitude and otherwise improving schedule predictability and confidence is a key way to improve end-product profitability. To that end, 69.2% of Java developers reported projects that were on or ahead of schedule, as compared to 53.8% of C developers. “Late” Java projects were also behind schedule less than those using C. Of late C projects, 22% were more than six months behind schedule, while only 3% of late Java projects were that far behind. Additionally, 83% of engineers using Java reported that their project schedule adherence improved versus their last project. In other words, projects not only tend to be shorter, as described above, but also hold less risk of delay.

Exhibit 13: Adherence to Current Project Schedule

(Percent of Respondents)



4. **Less Time Spent on Non-Value-Added Software Development**

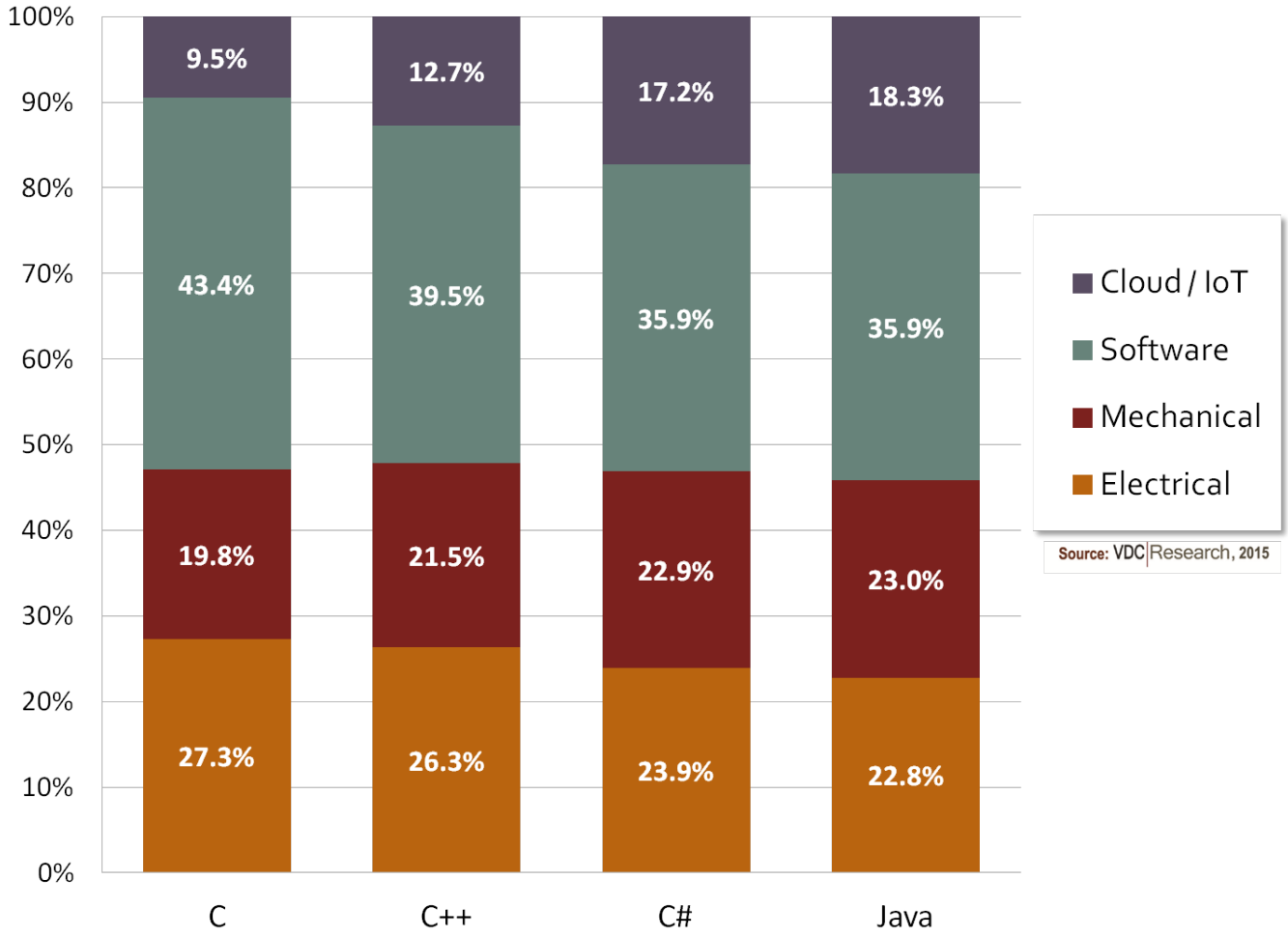
Perhaps the single largest driver of development cost savings, Java users report that their organizations spend a smaller percentage of their project team's total development cost on software. With software becoming the single largest cost center for engineering organizations, the ability to identify technologies that help limit additional development labor can have a significant impact on the overall project's costs. For example, the extensive middleware-like libraries offered by Java are one way that developers can minimize their time spent coding basic building blocks for communications, data management, and security. Built-in Java libraries covering these areas save time and are standards that can be leveraged between teams and partners. This time saved can allow organizations to focus a greater proportion of their labor resources on new areas of differentiation like Cloud/IoT functionality.

Leveraging these proven and tested functions is a focus area for Java benefits. A huge portion of many projects in IoT is the work to integrate with third-party, open source datacenter/Cloud services, and external code from customers and vendors. Integration with enterprise architectures, Cloud services, and third-party

enterprise workflow integration are also common requirements of IoT service development. These external – and internal – components are more likely to be written in Java as it is the ubiquitous choice for most Enterprise applications.

Exhibit 14: Estimated Distribution of Development Costs on Current Project

(Average of Responses)

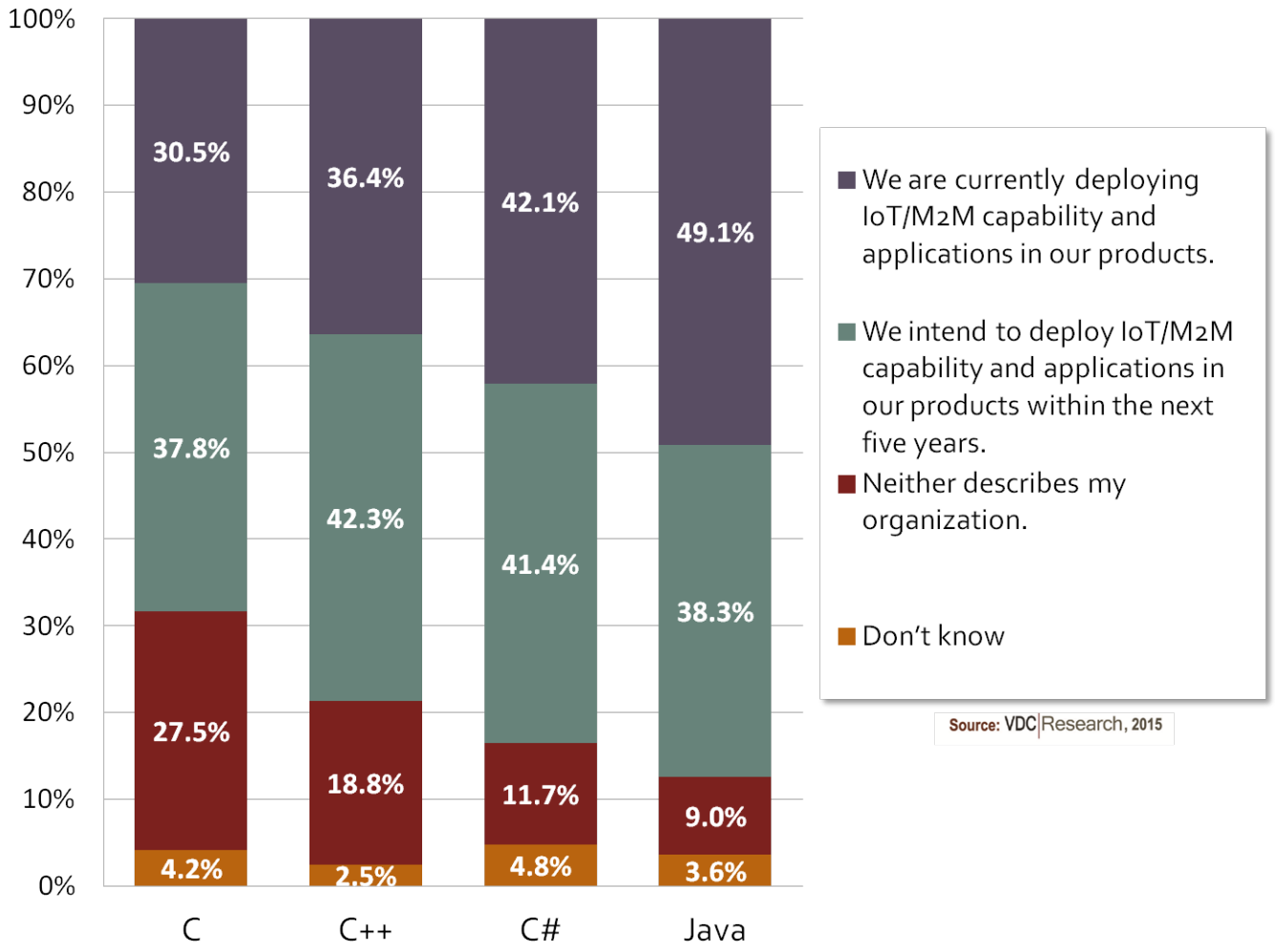


Java is not, however, always the right fit. Individual projects' requirements vary; and for certain applications with real-time latency requirements and/or small footprints, such as Automotive ECUs or sensors, C can make more sense than Java given processor characteristics and more limited system resources. Conversely, Java is often chosen for use within projects that have more complex and sophisticated functionality specifications, which can lead to an impression that Java causes higher BOM costs or more extensive software development requirements. In practice – as always – it is important to choose the right tool for the job.

As mentioned above, Java developers are more likely to be focusing on IoT development. In fact, 87% of Java developers said their organizations are either deploying IoT applications today or have plans to do so in the coming years. Our research also shows that added complexity does not have to translate to longer development timelines with greater risks of failure or delay. The shorter project durations and lower rates of schedule delays shown by these same Java developers with more complex projects indicate that Java's modern language features and third-party code and libraries can be a resource to capacity-constrained engineering organizations.

Exhibit 15: Adoption of IoT Capability and Applications

(Percent of Respondents)



Conclusion

The IoT is redefining product development and operational strategies. Engineering organizations are solving new problems and delivering more sophisticated, connected products – but often with a much greater cost of development. Furthermore, as more organizational revenue is tied to post-deployment content and services, the ongoing product development cycle is more directly impacting revenue opportunities and can even risk missed market windows. The ability to choose development technologies and platforms that enable flexibility in resource selection, access to more complementary, third-party software and that offer better track records of development schedule predictability is being placed at an increasing premium.

The diverse nature of IoT and embedded product development will continue to require project-by-project evaluation of product components and development solutions. For many types of projects, however, Java has become an increasingly relevant and compelling option. Java provides a platform that can help accelerate development through code reuse, eases the integration of new IoT and Cloud services with existing enterprise infrastructure, and offers access to a large partner and developer ecosystem that can offer labor cost savings.

VDC Research

About the Author

Contact Chris:
crommel@vdcresearch.com

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.

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.

For more information, contact us at info@vdcresearch.com.