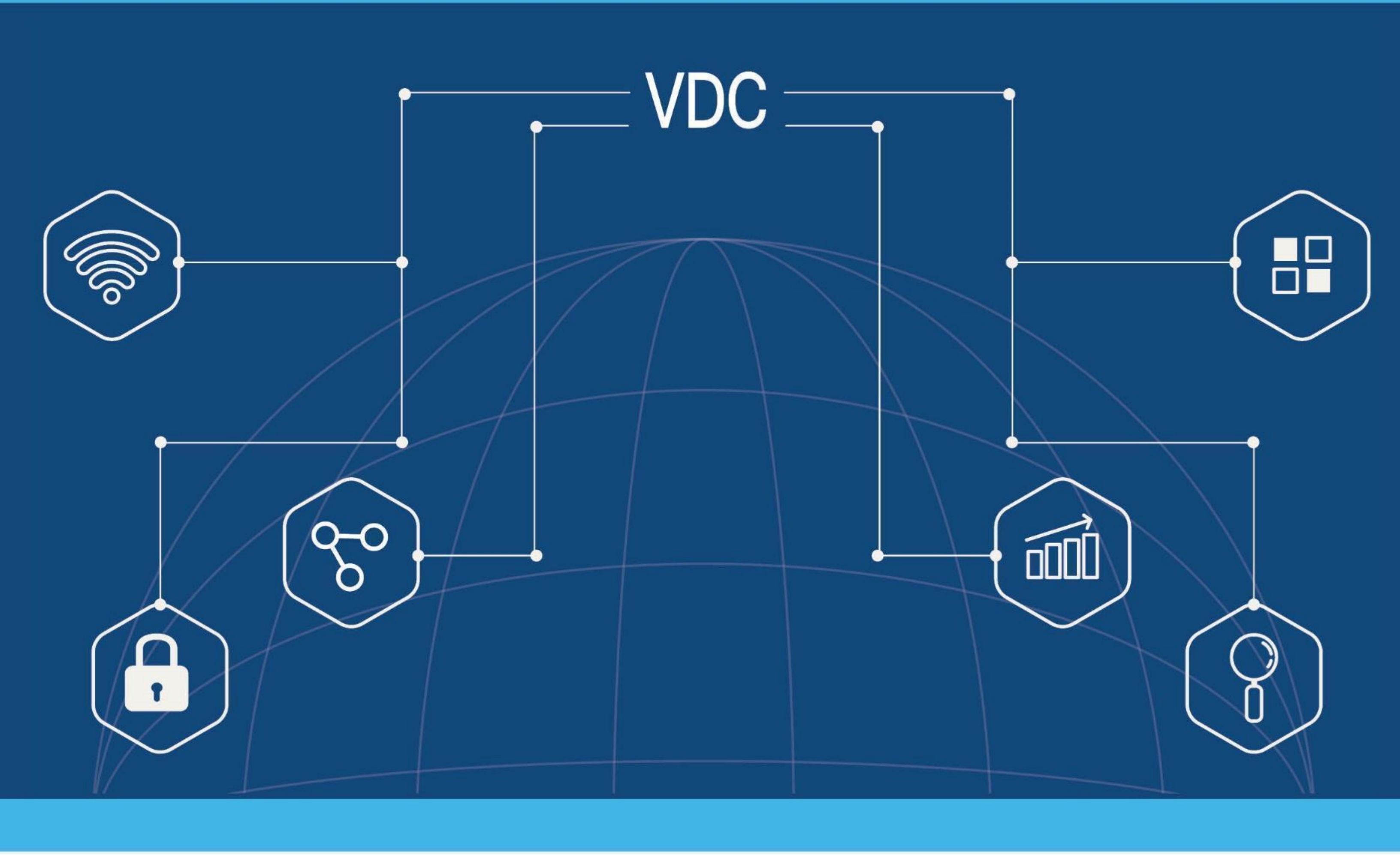
The State of IoT Software Development
A Benchmark for Teams and Leaders



Licensed to Distribute by:

Memfault

April 2024
by Steve Hoffenberg, Director; and Chris Rommel, Executive VP

VDC Research

Table of Contents

Table of Contents

Executive Summary & Key Findings

Internet of Things devices contain vast amounts of software code, such that software now accounts for the majority of project development costs. Combining cost constraints with time-to-market pressures is a potential recipe for poor software quality, which can reduce user productivity, risk cybersecurity vulnerabilities, and yield customer dissatisfaction. Fortunately, a variety of software development tools can help engineers improve code quality, reduce the time to find and fix bugs, and save many hours of developer time per year.

For this report, VDC Research conducted a global survey of more than 775 people directly involved in developing IoT products and/or embedded electronic systems, with questions about technical trends impacting software development. Respondents came from a variety of job roles and worked on products for a range of vertical markets. Their employers included OEMs, ODMs, independent product design and software development firms, and systems integrators. In this report, we analyze some of the findings from the survey and examine the implications for best practices and usage of software development tools and technologies.

Key findings from our research include:

- **Software drives differentiation, and project cost**
  - The median cost of project development was $556,000, and more than 60 of the projects cost at least $10 million to develop
  - Software accounts for nearly 60% of project development costs
  - The average project contained 548,000 lines of code (LoC), with ten of the projects exceeding 10 million LoC
  - Managers were more likely than the engineers or developers to believe their projects were ahead of schedule, and less likely to realize when projects were running late

- **Choosing the right tool saves teams time and money**
  - Organizations using remote device health and performance monitoring solutions were 3x as likely to finish ahead of schedule versus those collecting no data
  - 50% of organizations take more than a week to find the cause of reported software defects, while 20% take several months. More than 40% require more than a week to fix those defects once found
  - Only 8% of organizations release fixes within a day of finding software defects, yet 83% of respondents said their development team has adequate tools to efficiently fix defects when they are found in the field
  - Software defects reported by customers required anywhere from 75 additional person hours per year to fix for the least complex projects, and up to 3 person months for the most complex projects
  - Organizations using third-party tools to monitor device performance and health required one-third fewer person hours to remediate software bugs
  - Organizations using tools to monitor deployed devices spent half as much time remediating each software defect, allowing more time to focus on new, not old features
  - Engineers using third-party tools to collect device performance and health data saved 57% in overall project development costs versus those using in-house solutions

Introduction

The IoT has fundamentally altered product requirements and corporate business models. Product development organizations must innovate within this new engineering landscape while navigating tight cost and schedule constraints and growing expertise gaps. The addition of IoT connectivity is often a complicating factor for embedded electronics product development, due to expanding software stacks, connectivity hardware and services, communications protocols, cybersecurity risks, device management, and frequently evolving IoT cloud platforms. However, forward-looking product makers also realize the potential to leverage IoT connectivity and firmware-over-the-air (FOTA) capabilities to improve the product development process itself.

In this report, we explore the challenges facing today's product developers and look at the best practices and software technologies that engineering organizations can adopt to address them and focus on new areas of differentiation.

Survey Methodology & Respondent Demographics

VDC Research has extensive experience covering IoT product development, going back to the 1990s when the early Internet of Things was still referred to as Machine-to-Machine (M2M) technology. This document draws from our deep collective insights and knowledgebase, as well as the findings from a new global survey we conducted in 2024 specifically to provide the most recent and detailed insights from developers and engineers at organizations that are actively developing IoT products today.

For this report, VDC conducted an online survey of 783 people personally involved in developing IoT products and/or embedded electronic systems. This global survey offers insight into leading business and technical trends impacting product development organizations as well as the best practices implemented to address them. Respondents worked in a variety of job roles [See Exhibit 1], including:

- System architect
- Software developer
- Firmware engineer
- Hardware engineer
- Product/project manager
- Engineering manager/VP

Their employers included:

- OEMs
- ODMs
- Independent product design and software development firms
- Systems integrators and solutions providers

They came from companies large and small, from 36 different countries (approximately one-third were from the US). And the respondents developed products for a range of industries [See Exhibit 2], including:

- Aerospace and defense
- Automotive and transportation
- Communications and networking
- Consumer electronics
- Energy and utilities
- Industrial automation
- Medical devices and healthcare
- Retail automation and digital signage

Where appropriate in this document, we reference some of the many findings from this survey, including questions about development costs, project schedules, software coding and tools, and maintenance of software in deployed products.

Exhibit 1: Job Roles

(Percentage of Respondents)

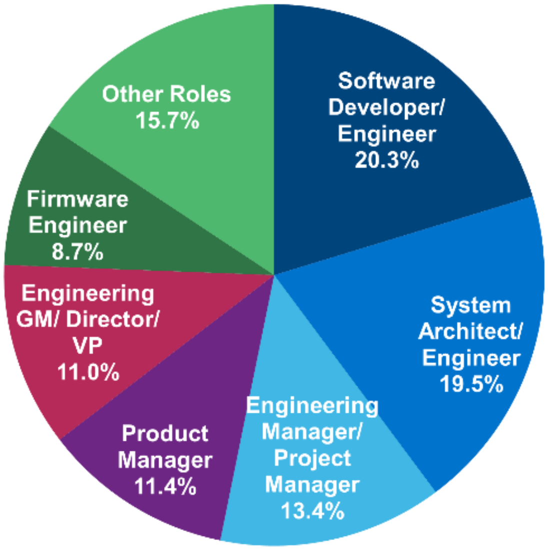
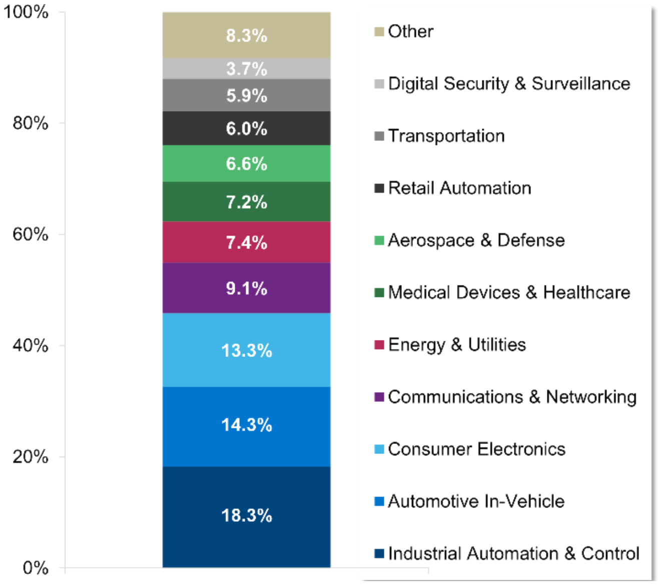


Exhibit 2: Primary Vertical Market

(Percentage of Respondents)



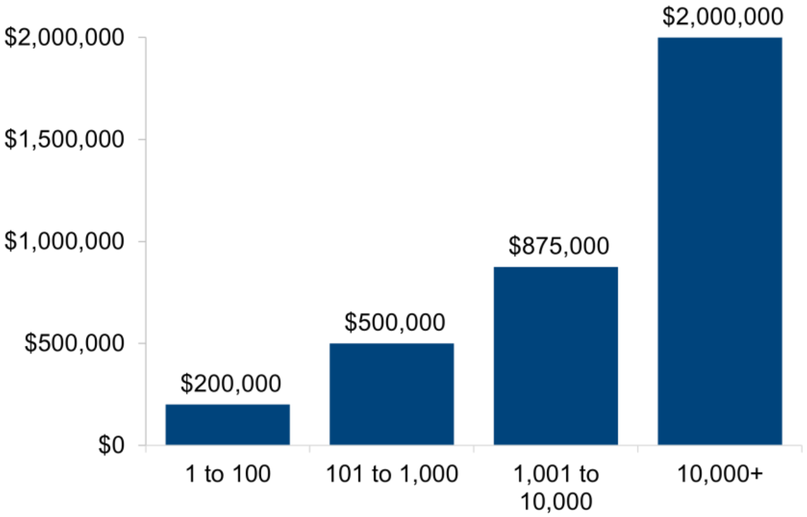
Pressures Facing IoT Product Development Organizations

The High Cost of Product Development

IoT products vary widely, from simple sensors (e.g. for measuring temperature or vibrations) to highly complex systems of systems (e.g. automobiles, military aircraft, and medical imaging systems). They can involve anywhere from a single engineer designing the hardware and writing the software, to hundreds of engineers and software developers working on highly segmented tasks with multiple layers of management overhead. As such, the costs of product development also vary widely. Our survey found that the median cost of developing an IoT product was approximately $556,000, with more than 60 of the projects exceeding $10 million each. In general, development cost was directly associated with the size of the company developing a product, i.e. larger organizations spent more on average, as shown below in Exhibit 3.

Exhibit 3: Development Cost of IoT Products, by Company Size (Total Number of Employees)

(Median of Reponses)

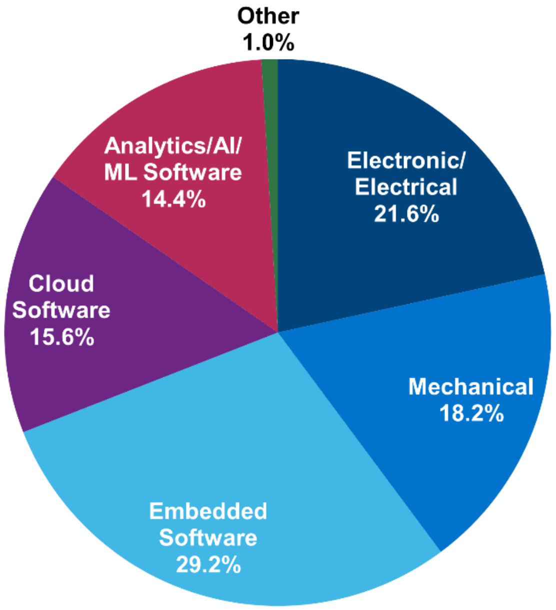


In recent years, the continual increase in development costs has been fueled by the rapid growth in the amount and complexity of software. Software might not yet be "eating the world" (as once postulated by Marc Andreesen), but it is eating up large chunks of product development costs. Among the projects referenced by our respondents to this survey, embedded software development accounted for the single largest share (29.2%) of product engineering costs, and if we add in cloud software and analytics software (including artificial intelligence/ machine learning), software development accounted for 59.2% of the total reported engineering costs [See Exhibit 4].

Software development accounts for nearly 60% of the total engineering costs

Exhibit 4: Percent of Product Engineering Costs by Task Type

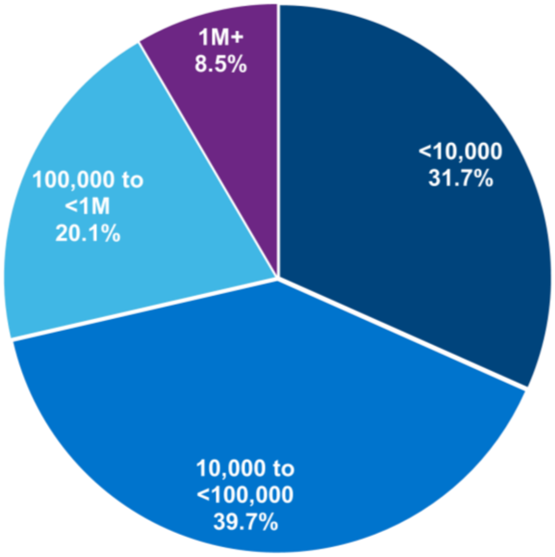
(Mean of Responses)



Further evidence of the importance of software—as well as its complexity—is demonstrated by the number of lines of code per project. As shown later in this report, more code can ultimately risk more bugs, as well as more time and resources to fix them. The average project had 548,000 lines of code (347,000 excluding mobile phones, which run even higher), and many now possess more than a million lines, including 10 projects with at least 10 million lines. However, even software projects with only thousands or tens of thousands of lines of code often exceed the ability of the developers to properly review and assess the quality of the code without assistance from automated tools.

Exhibit 5: Number of Lines of Code Per Project

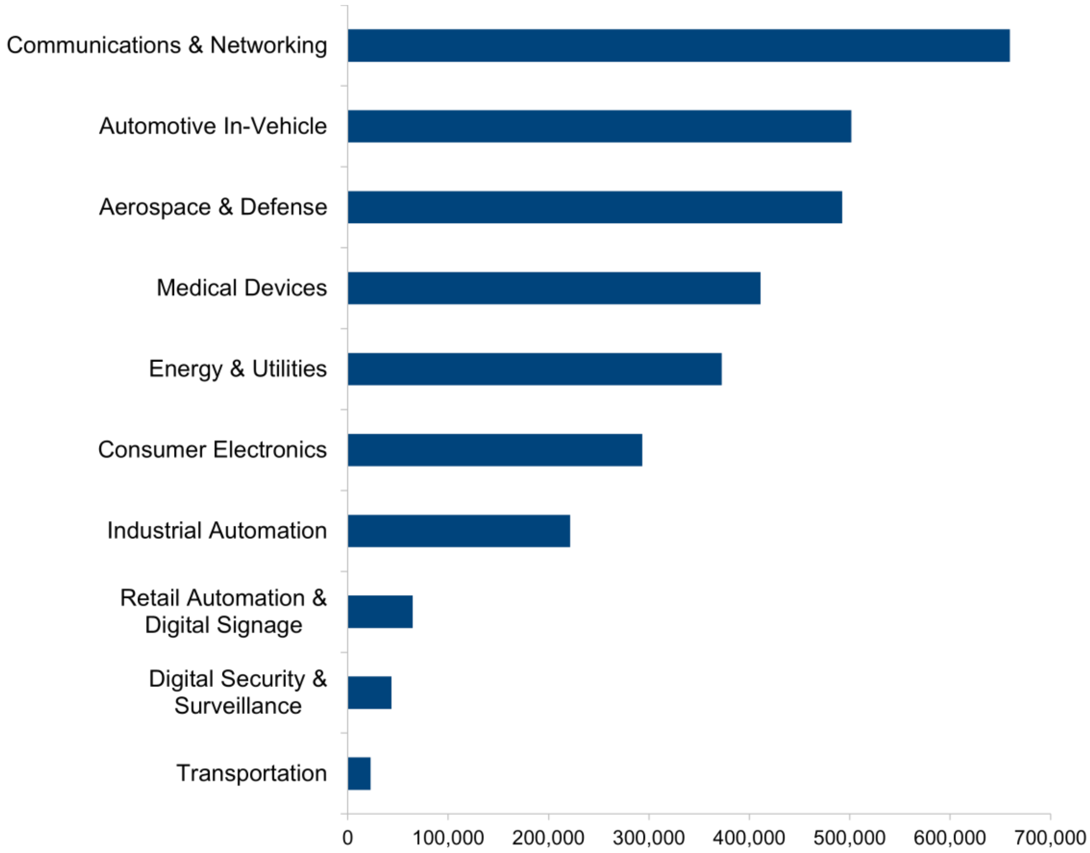
(Percentage of Respondents)



Projects for certain vertical markets tend to require more code than others, as shown in Exhibit 6, due to greater overall complexity of the systems, and often the need to meet more stringent functional safety, cybersecurity, or regulatory requirements. The Communications & Networking vertical had the largest number of lines of code on average, due to many large-scale cellular network infrastructure projects among our respondents. Moreover, in systems-of-systems design, many end products are a combination of dozens of projects and can now have hundreds of millions of lines of code. For example, it is over a decade since the Chevy Volt first made headlines as the first passenger vehicle with over 10 million lines of code. The Automotive industry's movement towards zonal architecture and software-defined vehicles has now resulted in many high-end vehicles having 200 to 300 million lines of code in total.

Exhibit 6: Number of Lines of Code Per Project, by Vertical Market

(Mean of Responses)



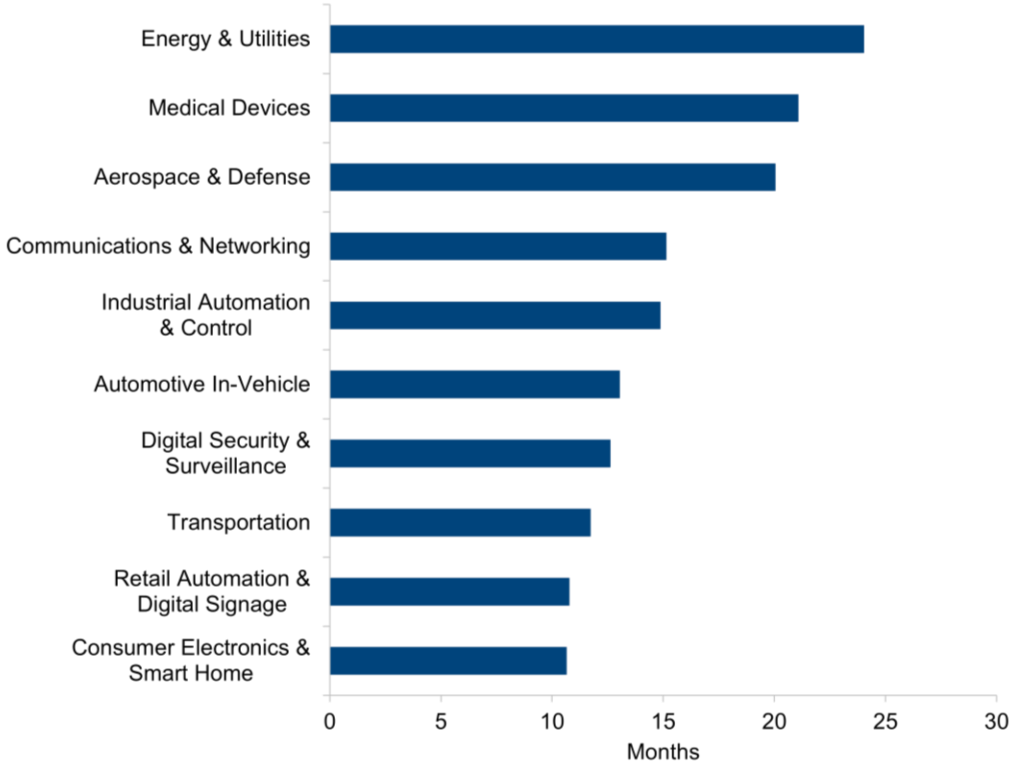
AI & Software Content Challenge Schedules, Necessitate New Solutions

As the pace of technology innovation continues to accelerate, product development organizations are under increasing pressure to get products to market faster. Doing so enables their organizations to generate revenue sooner, fend off competition, and capture or retain market share.

The average length of the product development cycle in our survey was 16.3 months, although it varied considerably by vertical market, with Energy & Utilities, Medical Devices, and Aerospace & Defense having the longest averages, at 24.0, 21.1, and 20.1 months, respectively [See Exhibit 7], due to the regulatory requirements in those industries.

Exhibit 7: Length of Product Development Cycles, by Vertical Market

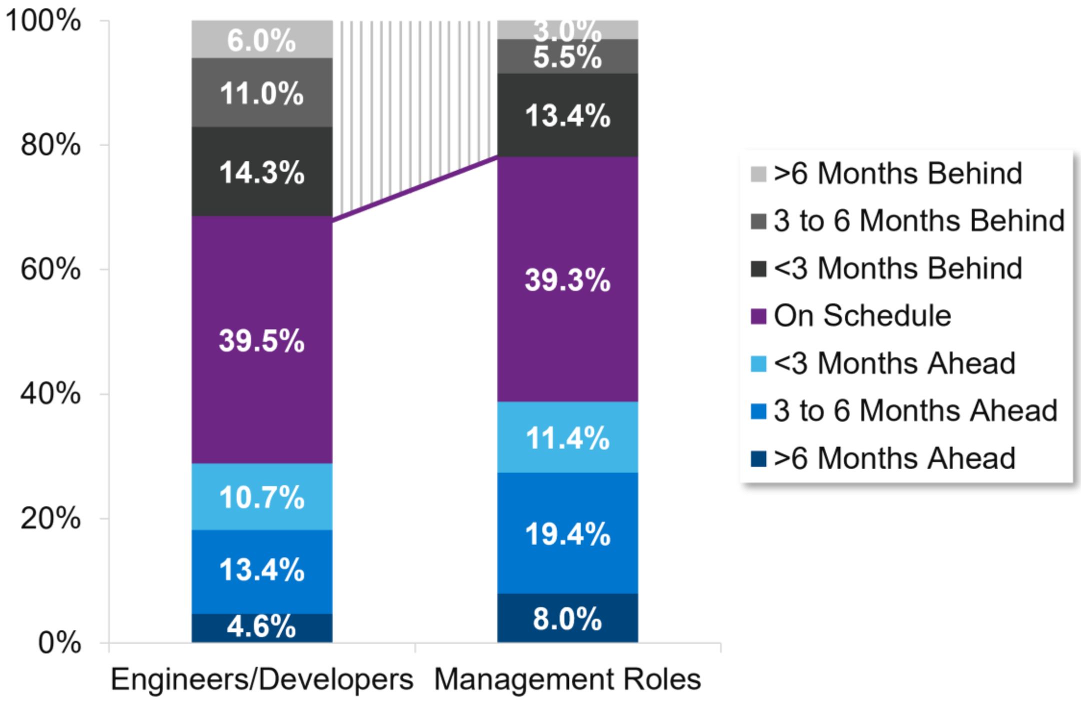
(Mean of Responses)



How well organizations are able to adhere to their planned development cycles is also an important metric, which can have compounding effects across an organization's agility and product portfolio. Among those respondents directly working in engineer or developer roles, 28.7% said their projects were ahead of schedule and 31.3% said their projects were behind schedule, while among managers, 38.8% said their projects were ahead of schedule and 21.9% said they were behind schedule, and the differences were greatest for projects at least three months ahead of or behind schedule [See Exhibit 8]. This suggests a disconnect in which managers may be overly optimistic with respect to schedules and less likely to realize when projects are running late or could benefit from the use of new tools.

Exhibit 8: Current Project Schedule Adherence, by Role Type

(Percentage of Respondents)



Failure to release products in a timely manner can have negative effects beyond the development process itself. These issues can impact planned coordination of manufacturing, distribution, and marketing, particularly if management is unaware of when engineering projects are delayed. For these reasons, many organizations are tempted to release products before the software is fully baked—that is, without adequate testing and debugging under the broad range of conditions in which they may be used in the field—in order to get the products manufactured and into distribution channels on a pre-planned timetable. Ready or not, here it comes.

The increasing volume of content and functionality now delivered post initial shipment only complicates the issues and magnifies the cost for organizations that lack adequate software monitoring and management solutions.

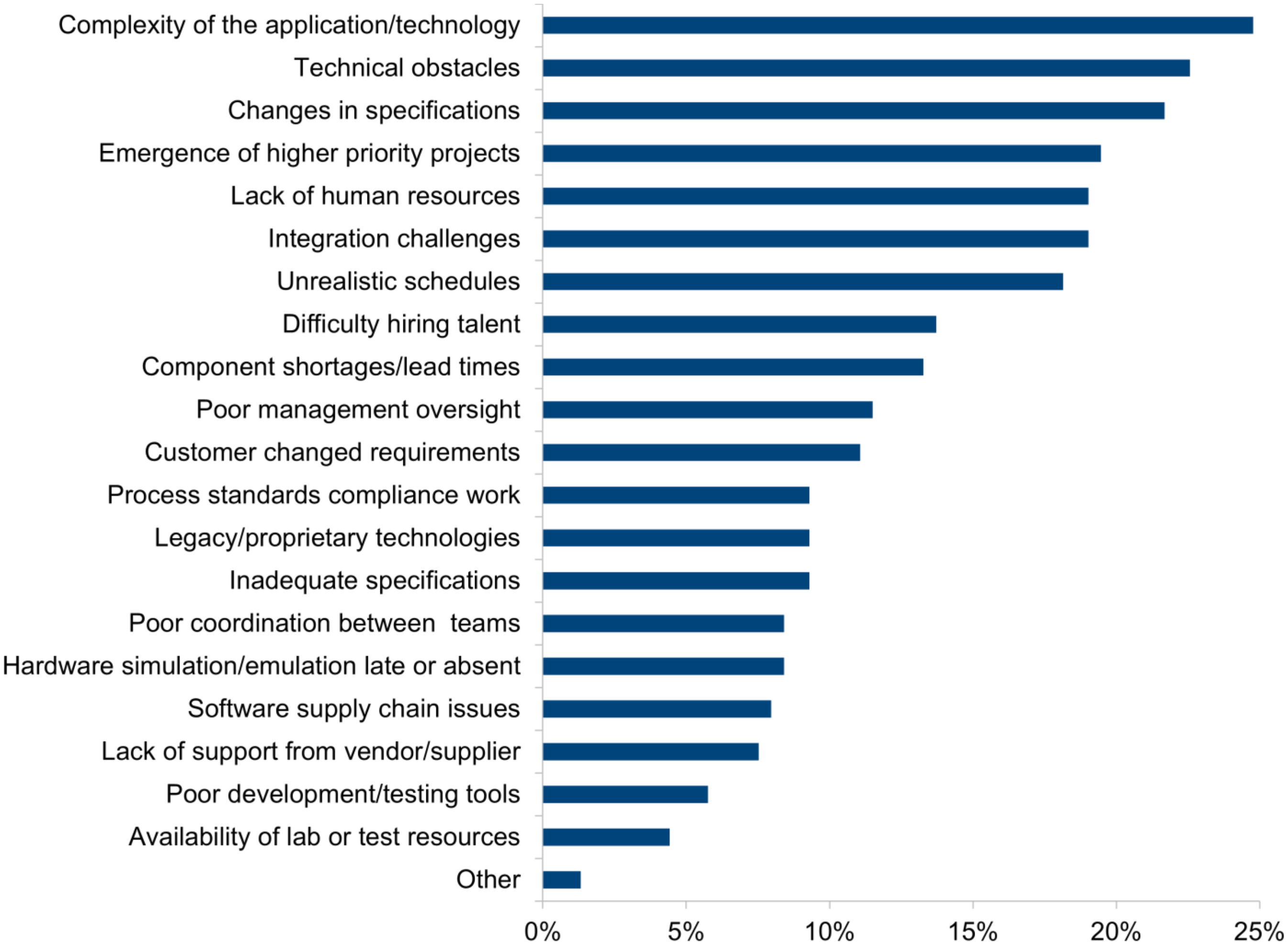
Development projects can be behind schedule for a variety of reasons, with the two most common being the complexity of the application/technology, and technical obstacles (as well as a long list of other possible causes shown in Exhibit 9 below). Such complexity is not only an organic function of new features but also a result of development organizations being pushed in areas outside their traditional siloes of expertise. Furthermore, these challenges are often paired with increasing expectations on the part of customers, salespeople, and product managers, who demand more features and capabilities out of products, sometimes in response to competing products on the market. The incursion of artificial intelligence and machine learning features into a larger share of embedded devices may exacerbate the complexity. In our survey, 43.3% of respondents said their current projects include artificial intelligence or machine learning, and 48.4% said they expect to include such capabilities in similar projects three years from now.

43% of projects currently include AI/ML

48% said they expect to include such capabilities in 3 years

Exhibit 9: Reasons for Project to be Behind Schedule

(Percentage of Respondents)



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

Challenges of Developing & Maintaining IoT Products

Code Quality

In addition to the cost and time pressures described above, product makers face a host of challenges. One such challenge is ensuring the quality of a product’s software code. Software quality consists of many factors, such as functionality, cybersecurity, adherence to regulatory and coding standards, speed of execution, severity of any bugs or defects, efficient memory usage, maintainability, etc.

Functionality—ensuring the product does what it is supposed to do—is arguably the most important characteristic of software code. Of course, there are many nuances to whether and how well software functions. At its most basic level, if a product doesn’t perform its primary purpose (for example, a temperature sensor doesn’t properly measure temperature), the product should not leave the factory. However, if a product performs its basic tasks adequately, it may still be released even if some of its less important or rarely used functions do not work properly under certain conditions. The development team may or may not be aware of all the conditions under which it fails. Software testing can catch many faults in code, but such testing generally cannot anticipate every possible scenario that a product might encounter when used in the field.

Cybersecurity—ensuring the product does not do what it is not supposed to do—is also an extremely important characteristic of software code. For many software developers, this approach is counterintuitive to how they were originally trained to write software, although the situation has been changing in recent years as cybersecurity lapses have become more frequent and prominent. Here again, software monitoring and testing tools can be of great assistance in uncovering both obvious and subtle programming errors, but they can never prove a negative (*e.g.* the code has no vulnerabilities) on complex software involving thousands or millions of lines of code.

Unfortunately, one third (33%) of respondents do not believe or are unsure whether their organization adequately tests the cybersecurity of its products. Furthermore, that portion rose to half (50%) among those whose projects were behind schedule. Too often cybersecurity testing can be cut short or skipped entirely when projects are running late.

33% of respondents do not believe their organization adequately tests the cybersecurity of its products

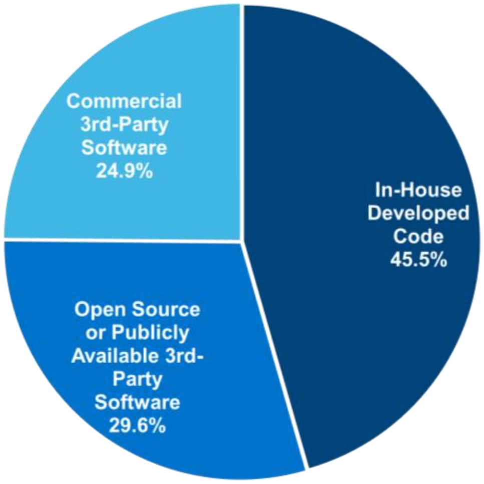
That portion rose to 50% among those whose projects were behind schedule

Origins of Software Code

Given the complexity of software in today’s embedded systems and the need to meet cost and time-to-market pressures, most product developers incorporate pre-existing software code originating from outside their organizations. Although in-house code accounts for the largest share, commercial third-party software and open source software collectively account for more than half of reported software content [See Exhibit 10]. While commercial off-the-shelf software (COTS) is usually well-vetted for code quality, it is not always entirely free of bugs or vulnerabilities. Open source or other publicly available software may or may not be well-vetted, and often constitutes software of unknown provenance (SOUP), the origins of which have not been fully documented.

Exhibit 10: Estimated Percentage of Software Code in Final Projects, by Code Origin

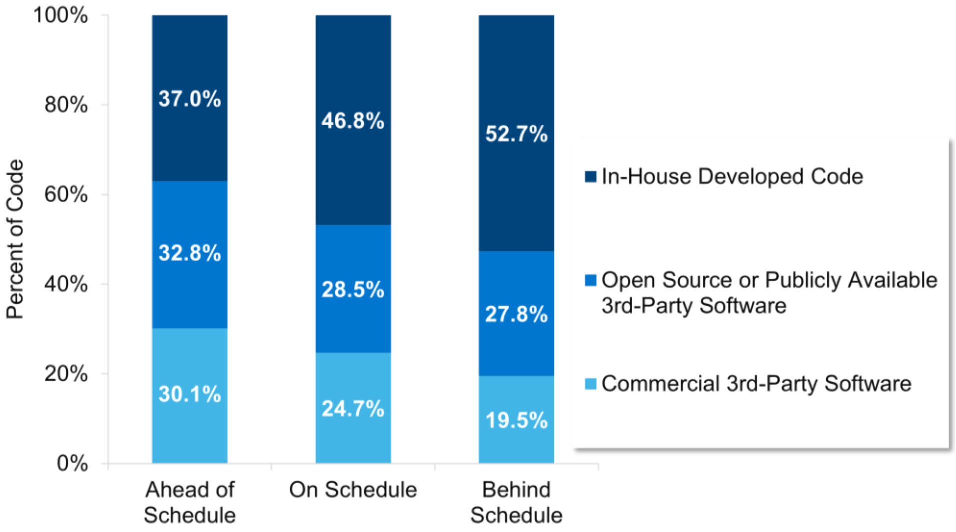
(Mean of Responses)



Despite the uncertainties regarding third-party software, in-house developed code may present the greatest challenge to development schedule adherence. On average, those respondents whose projects were behind schedule tended to have a greater share of in-house code in their projects [See Exhibit 11].

Exhibit 11: Estimated Percentage of Software Code in Final Projects, by Project Schedule Performance

(Mean of Responses)



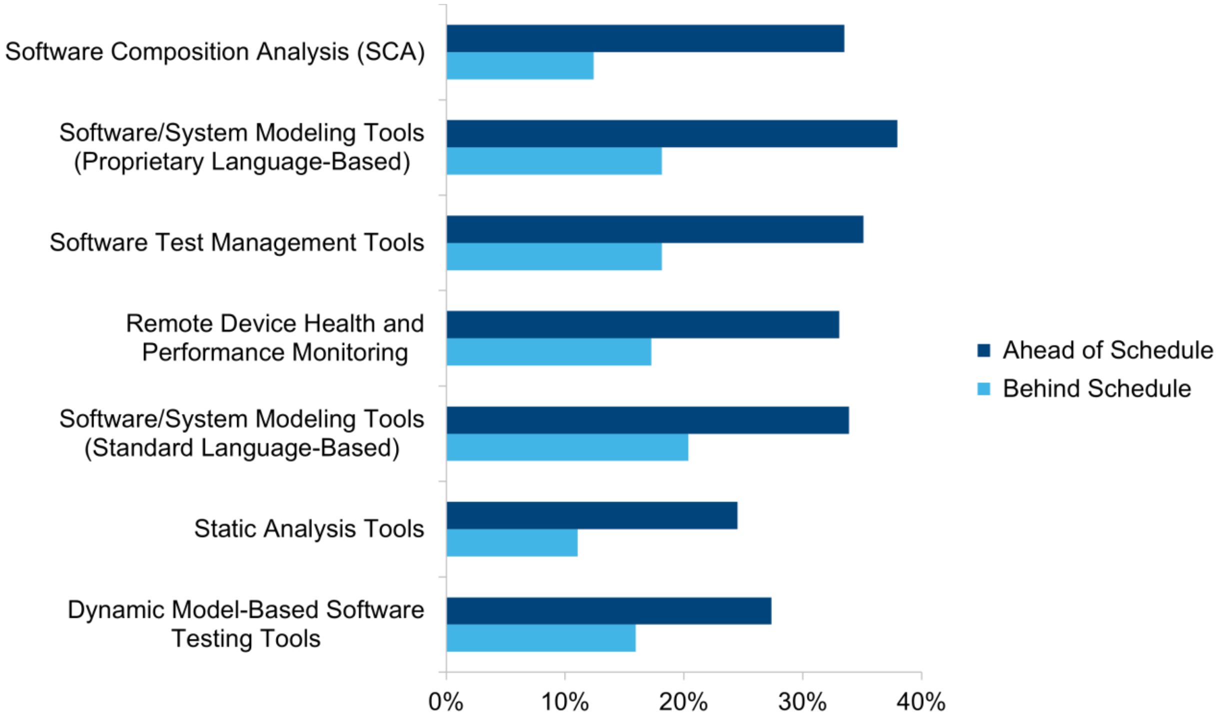
The Need for New Software Development Tools

Developers can select a range of development solutions to help them address next-generation IoT software engineering requirements. Dozens of these types of tools are available, ranging from basic code editors and compilers to sophisticated modeling and software composition analysis tools, often within an integrated development environment (IDE). Static analysis and dynamic/unit code testing tools are especially designed to find and help fix bugs and vulnerabilities in code during the development process.

Such tools can not only improve the quality of code, but also help accelerate the development process. Among our survey respondents, for example, several types of tools were particularly prevalent in usage on projects that were ahead of schedule vs those that were behind schedule [See Exhibit 12]. The greatest difference between the groups ahead of schedule and behind schedule was in the use of software composition analysis (SCA) tools. These SCA tools identify and analyze open source code, identify vulnerabilities and code quality issues, as well as uncover potential software licensing issues.

Exhibit 12: Software Development Tools Most Used in Projects Ahead of Schedule vs. Behind Schedule

(Percentage of Respondents)



With the growing number of connected and complex IoT devices, remote device health and performance monitoring tools will become increasingly important going forward. Since these tools are typically used in conjunction with devices deployed in the field, they are not often associated with initial software development. In many cases, however, it is prudent to release software updates to a small sample of deployed devices to evaluate its quality in real-world conditions prior to a general release. Integrating remote device health and performance monitoring tools into the product development process can pay dividends once products are deployed and provide valuable feedback that aids the development of subsequent projects. Combined with the ability to remotely update software and firmware in deployed products—often referred to as firmware-over-the-air updates, or FOTA—performance monitoring solutions provide developers greater confidence in their ability to find and fix issues that arise in the field.

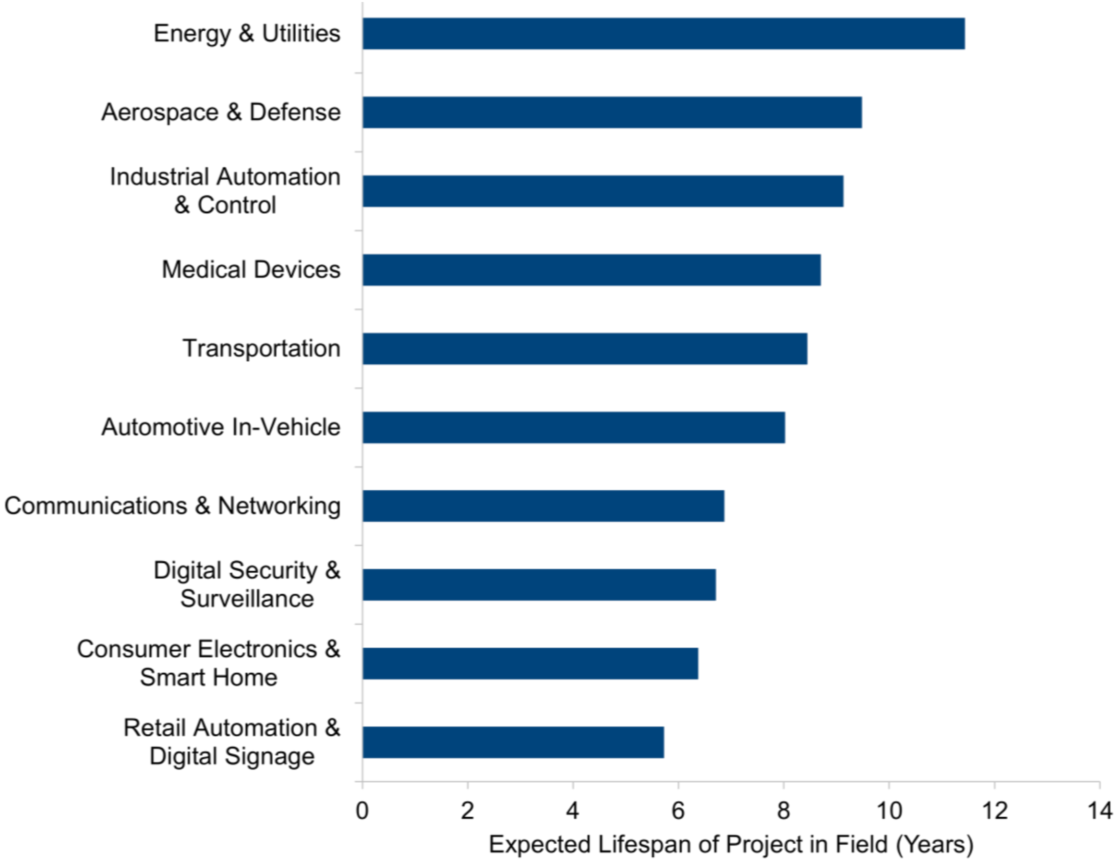
Maintaining Code for Deployed Products

Organizations using remote device health and performance monitoring solutions were 2x as likely to finish ahead than behind schedule

Once IoT products are released into the field, they typically remain there for years, creating a range of considerations and challenges for IoT software developers. The expected life of deployed products varies with the specific product, but it is often associated with the vertical markets in which the products will be used. In our survey, the overall average life expectancy of their projects in the field was 8.0 years, with the longest being 11.4 years in the Energy & Utilities sector, as shown in Exhibit 13.

Exhibit 13: Life Expectancy of Projects Once Deployed, by Vertical Market

(Mean of Responses)

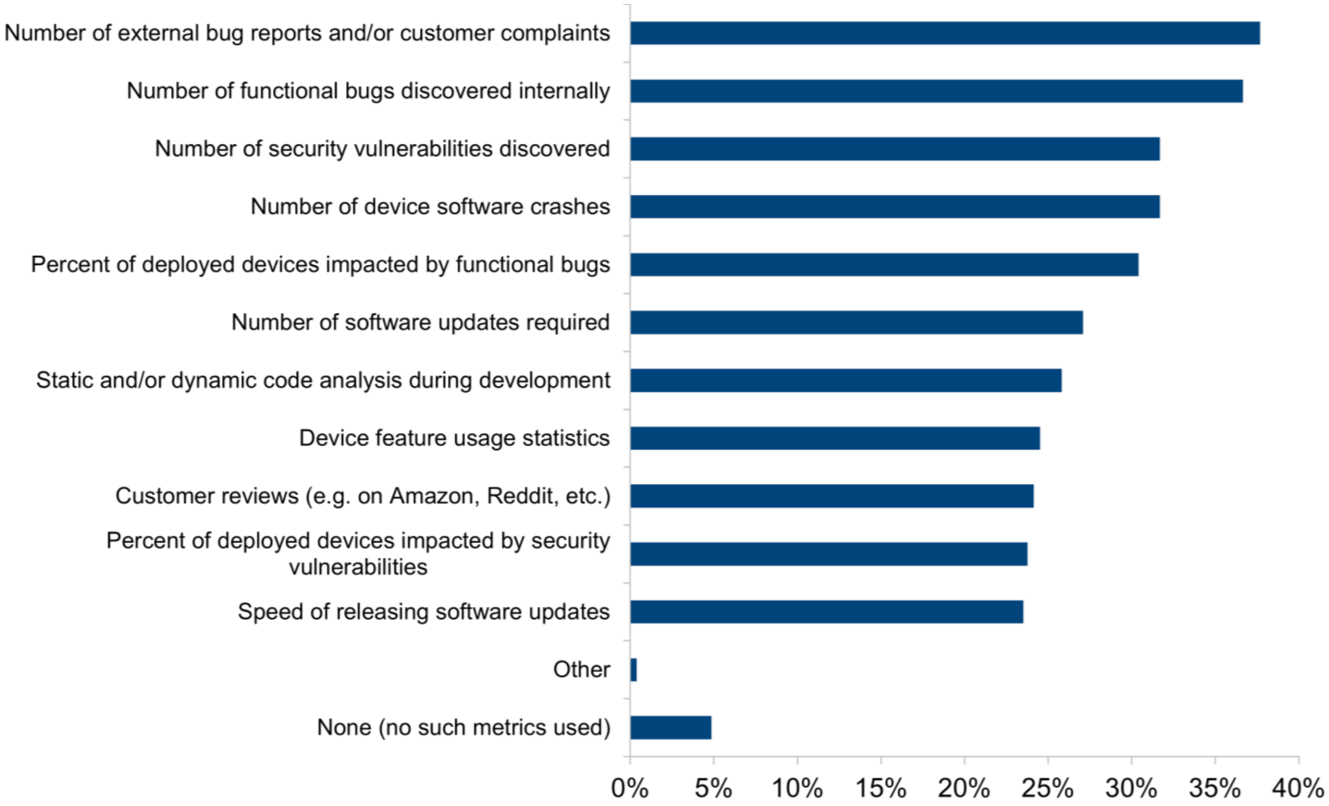


With the complexity of software in today's connected devices, nearly all of them will require software updates during their useful life to fix bugs, mitigate security vulnerabilities, and sometimes add new features. Yet only one third of the projects in our survey had remote firmware-over-the-air update, or FOTA, capability.

In order to produce such updates, developers need to assess the quality of software as it is being used in devices in the field, as well as understand how they are being used to best design new features. Exhibit 14 shows the metrics by which organizations most commonly measure the quality of their deployed software, with the top metric being the number of external bug reports and/or customer complaints. This is less than ideal for product makers, as it fosters customer dissatisfaction. Although an organization's internal testing often continues to find software bugs after devices are deployed (as shown in Exhibit 14), such testing is insufficient to catch and fix all the errors that might exist in the code as well as functionality issues that may arise from FOTA patches or unforeseen operating conditions. There are better ways, discussed in the following section, to measure the quality of code in deployed products and monitor for issues upon deployment.

Exhibit 14: Metrics Used by Organization to Measure the Quality of Deployed Software

(Percentage of Respondents)

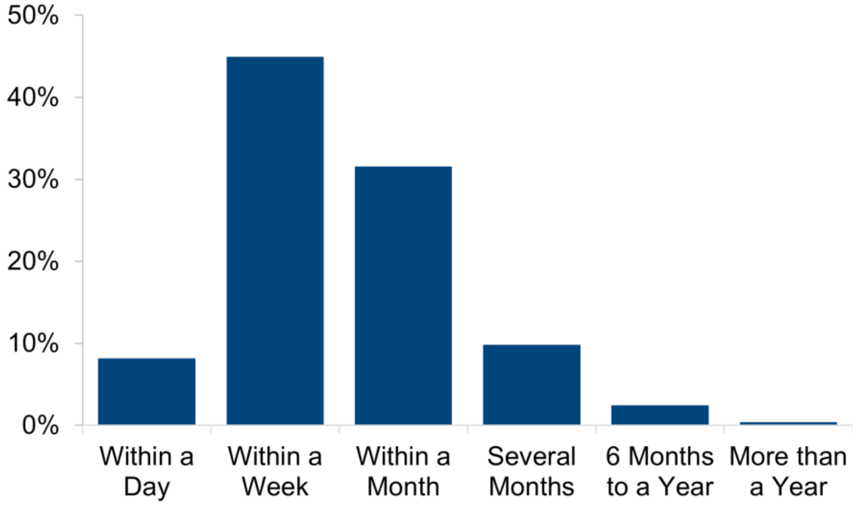


Note: Percentages sum to greater than 100% due to multiple permitted responses.

When products are deployed, bugs and other software defects often surface within a month, but VDC's survey data shows that fixes aren't typically deployed for another one to four weeks, leaving vulnerable or not fully functional devices in use. As shown in Exhibit 15, only 8% of survey respondents said their organizations typically release fixes within a day of finding software defects (which we believe should be the goal), with 13% taking several months or more. Yet in response to another question, 83% said they thought their development team has adequate tools to efficiently fix software defects when they are found in the field.

Exhibit 15: Time to Deploy Software Fixes Once Bugs are Detected in the Field

(Percentage of Respondents)



The quantity of software errors tends to increase as the complexity of the code increases. Among our survey respondents, as the number of lines of code in their projects went up, the number of bugs reported by customers also rose, as shown in Exhibit 16, from a median of five such bugs per year for projects with less than 100,000 lines of code, up to 16 bugs per year for projects with a million or more lines of code. And the number of engineering person-hours required to remediate each defect also rose with the number of lines of code in the project, as shown in Exhibit 17, from a median of about 15 hours for projects with less than 100,000 lines of code, up to 30 hours for projects with a million or more lines of code. This effect is likely due to the greater effort necessary to find bugs within a larger codebase, and to subsequently test all the results of code changes to ensure new bugs have not been introduced.

Combined, fixing the software defects reported by customers requires anywhere from 75 additional person hours per year of development for the least complex projects, up to 3 person months (480 person hours) for the most complex projects.

Fixing software defects requires anywhere from 75 additional person hours per year for the least complex projects, up to 3 person months for the most complex projects

Exhibit 16: Number of Software Bugs/Defects Reported by Customers in the Past Year per Deployed Product, by Number of Lines of Code in Product

(Median of Responses)

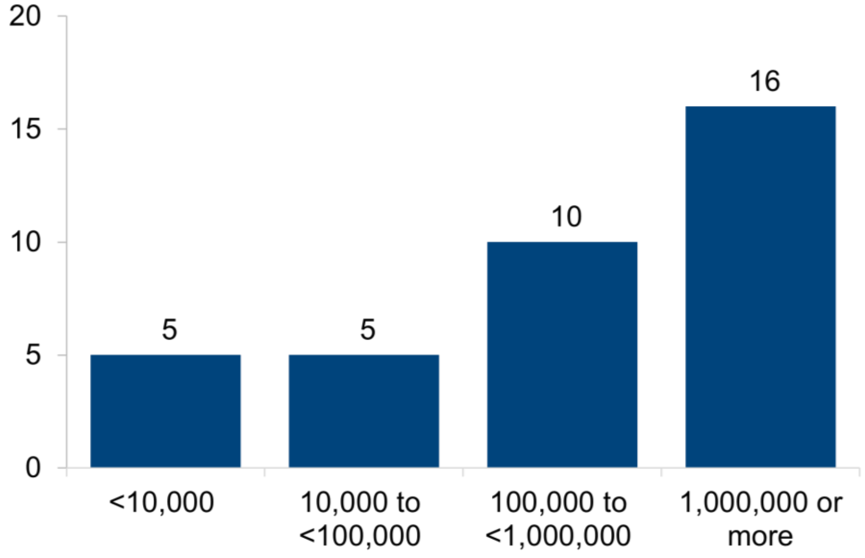
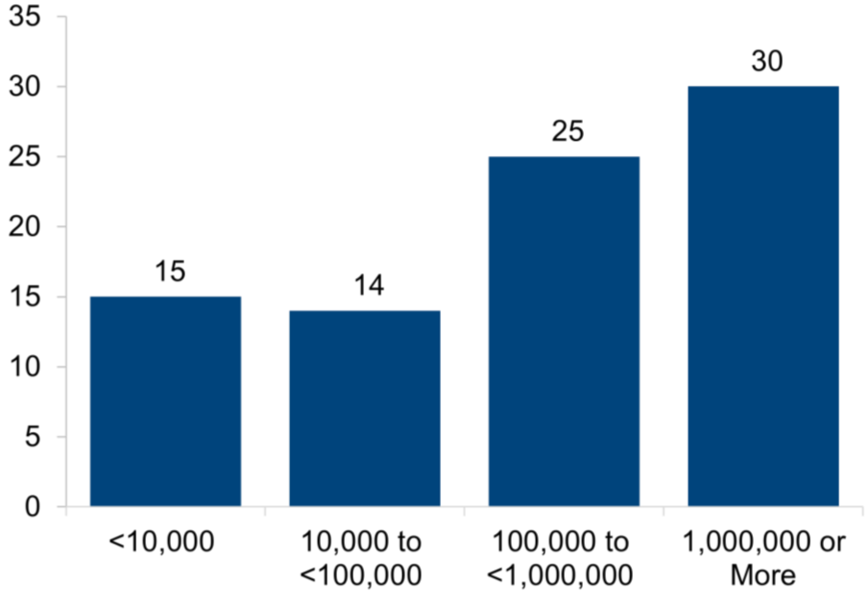


Exhibit 17: Number of Engineering Person-Hours to Remediate Each Software Defect Reported by Customers in the Past Year, by Number of Lines of Code in Product

(Median of Responses)



Benefits of Device Data Collection

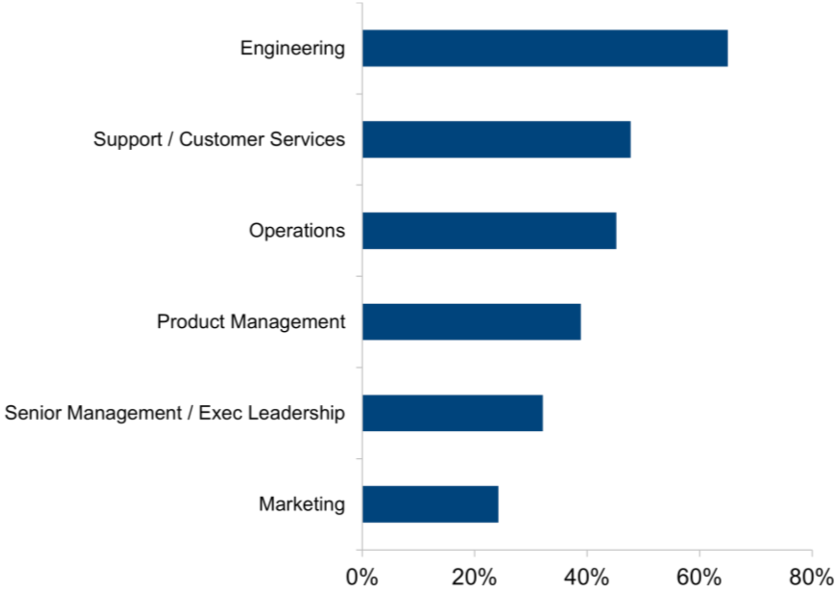
As can be seen from the above data, embedded software development still has room for improvement. The explosion of content and business value delivered through software has made such improvement a point of corporate focus and imperative. The ability to gather real-world usage data directly from deployed devices can help product makers find software defects in a timelier manner with fewer customer bug reports and complaints. Additionally, doing so can enable product makers to improve performance and iterate features more quickly by better understanding actual device usage patterns.

Collection of device data from the field also reduces the need for customers to register complaints or bug reports through distribution channel partners, such as dealers or systems integrators, where the relay of information often yields time delay, lack of sufficient technical details, and/or potential misunderstandings due to “game of telephone” types of alterations as information is relayed across multiple parties. Automated development tools are even more useful if they can detect software problems in devices in the field before customers even notice the problems.

In organizations that collect device performance and health data, it is not only used by software development teams, but also by product management, marketing, operations/manufacturing, customer support, and executive leadership teams [See Exhibit 18].

Exhibit 18: Teams in Organization That Use Device Performance and Health Data Collected from the Field

(Percentage of Respondents)

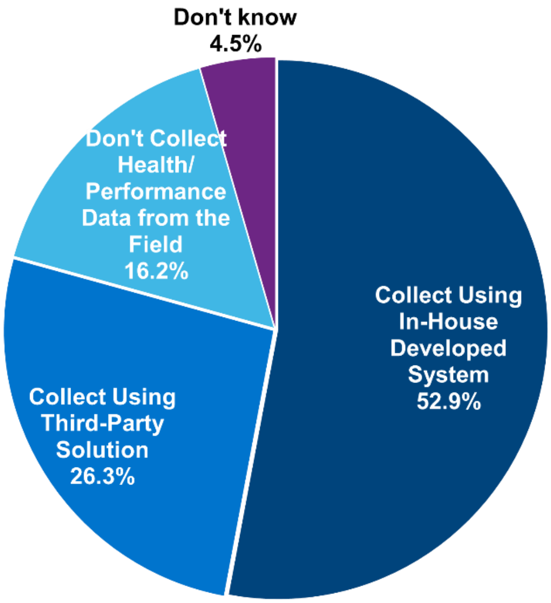


Note: Percentages sum to greater than 100% due to multiple permitted responses.

More than three quarters of our survey respondents say their organizations collect device performance and health data from deployed products, with the majority using in-house developed methods [See Exhibit 19].

Exhibit 19: Collection of Device Performance and Health Data from Products in the Field

(Percentage of Respondents)



Although in-house developed methods can be useful, there are many benefits to using third-party solutions, as described in the sections below.

Impact on Schedule Performance

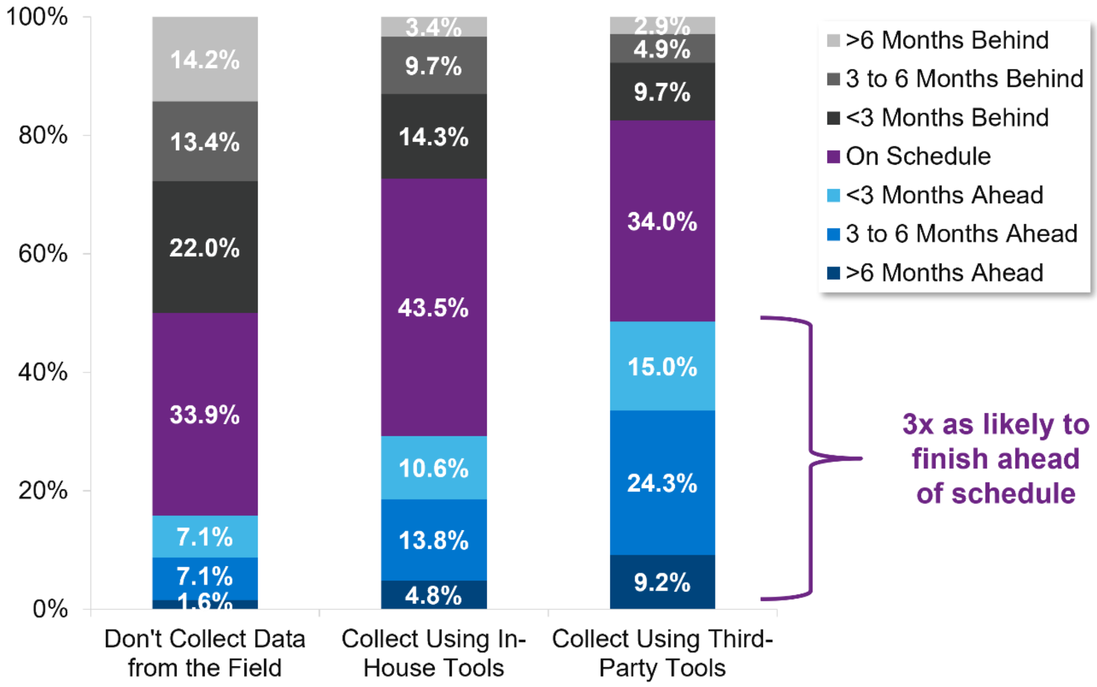
In an era of connected device proliferation, device performance and health monitoring solutions are emerging as an effective tool to reduce project uncertainty and cost. Not only did our survey find that projects in organizations using in-house developed tools to collect device health and performance data from the field were more likely to be behind schedule, but those using third-party solutions to collect such data were most likely to be *ahead of schedule* [See Exhibit 20].

Organizations using remote device health and performance monitoring solutions were:

3x as likely to finish ahead of schedule vs those collecting no data
1.7x as likely to finish ahead of schedule vs those using in-house tool

Exhibit 20: Project Schedule Performance,
by Collection of Device Performance and Health Data from Products in the Field

(Percentage of Respondents)



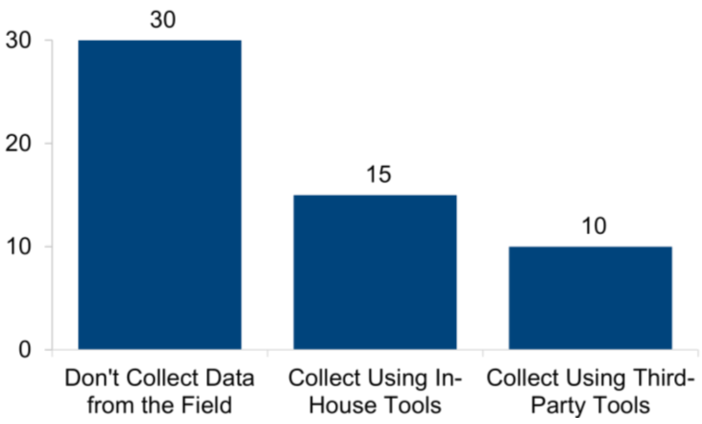
Save Time Fixing Bugs with the Right Tool Choices

Organizations using tools to monitor deployed devices spent half as much time remediating each software defect, allowing more time to focus on new, not old features

Furthermore, organizations using third-party tools to collect device health and performance data from the field required the fewest engineering hours to remediate each software defect, as shown in Exhibit 21.

Exhibit 21: Engineering Person-Hours to Remediate Each Software Defect, by Collection of Device Performance and Health Data from Products in the Field

(Median of Responses)



In short, bugs get fixed fastest when organizations use third-party tools for device data collection from the field. Considering the cost per hour of software development time, not to mention the opportunity costs of taking developers away from other tasks and projects, remediating such software defects more quickly has obvious and direct benefits to product development organizations.

Overall, the median cost of project development was higher for those using in-house tools to collect device performance and health data ($875,000), compared to those using third-party tools ($500,000). And for those using in-house tools, 27.1% of the development costs went to embedded software development, compared to 22.3% for those using third-party tools. In other words, engineers using third-party tools to collect device performance and health data saved 57% in overall project development costs versus those using in-house solutions.

Engineers using third-party tools to collect device performance and health data saved 57% in overall project development costs versus those using in-house solutions

Vertical Market Implications & Considerations

Failures in both functionality and cybersecurity can have devastating consequences in many products where human health and safety is directly at risk, such as in automobiles, medical devices, military equipment, and the like. Software quality is especially important in nearly all markets, not only where there may be severe functional impacts, but also in less obvious scenarios, where data is misused, or privacy is breached. Additional information about selected vertical markets is described below.

Automotive In-Vehicle Systems

Automobiles represent a unique and challenging vertical market, and the only one in which approximately 100 million units a year are shipped that have evident risk to human occupants. And perhaps no industry is going through greater degrees of simultaneous technological upheaval than the automotive industry, through:

- Increasing use of electronic sensors to monitor and/or control nearly every vehicle subsystem (including drive-by-wire electronics replacing many mechanical systems)
- Advanced driver assistance systems (ADAS) and autonomous driving developments
- Vehicle-to-vehicle and vehicle-to-infrastructure communications
- Digital dashboard display clusters
- Advanced infotainment centers with touchscreen displays, including the ability to pay for content and parking
- Cloud-based services for customers, dealerships, and manufacturers
- Electrification of powertrains, including their requisite batteries and vehicle charging systems

More and more, cars are transforming into software-defined vehicles, with many now containing over 100 million lines of code. Fortunately, most software development projects only involve specific vehicle subsystems that are more manageable in size. Unfortunately, developers must ensure that their subsystems interact correctly with all the other vehicle subsystems, which is complicated by the automotive industry's tiered supply chain. Collectively, vehicle makers must also do their best to thwart the threat of cyberattacks, through which organized groups of hackers may seek to steal cars, cause mayhem by shutting down the operations of large numbers of vehicles, or reap financial rewards by pilfering credit card information or other data that may be stored or accessed by the vehicle.

The industry's movement toward the utilization of more standard software frameworks and development tools (*e.g.* AUTOSAR, COVESA, MISRA) as well as the adherence to process standards such as ISO 26262 are also reinforcing the need and desire for development approaches that can offer both higher levels of automation and traceability across the development cycle.

In our survey, automotive projects had the highest number of reported software defects, an average of 29.8 per project, compared to 18.8 in the overall survey. Software defects will continue to be a challenge as more software features originate from the tiered supply chain.

Automotive projects had the highest number of reported software defects, an average of 30 per project, compared to 19 in the overall survey

Medical Devices

The medical device industry is one in which regulatory and functional safety requirements complicate the product and software development process, and increase the value of and need for change management, traceability, and collaboration across the development cycle. The IEC 62304 standard for medical device software, for example, not only places requirements on the software code, the standard also places requirements on the entire software development process, including risk assessment, cybersecurity, testing, and software maintenance plans. Medical devices are divided in classes, depending on risk of injury, with Class A (no risk of injury), Class B (risk of non-serious injury), or Class C (risk of serious injury or death). Class B and Class C devices require additional documentation and testing, including validation of the software code by specialized software tools or a certified test house. Any change of code may require re-validation of portions of the software, which can be both time-consuming and expensive.

Adding IoT connectivity introduces greater potential for high-risk cybersecurity vulnerabilities, not only related to the functionality of the devices, but also to privacy of medical data collected, especially where legal constraints such as HIPAA compliance is required. The addition of graphical user interfaces to medical devices further complicates the development process and may cause software development teams to select the use of larger and more fully featured operating systems (*e.g.* Linux instead of an RTOS). Both of these factors can add to the software development and testing burden.

In our survey, medical device projects were the most likely to be running *behind schedule,* at 42.9% of medical projects vs. 28.9% of projects overall.

Medical device projects were the most likely to be running *behind schedule*, at 43% of projects, vs. 29% *behind schedule* in the overall survey

Industrial Automation Systems

Factories keep the economy running, both literally and figuratively. Discrete manufacturing facilities make everything from semiconductor chips to kids' toys, while process manufacturing facilities pump out pharmaceuticals, gasoline, food ingredients, and many other chemical and biological products. The Internet of Things originated in industrial automation systems, back when it was being called machine-to-machine communications. In the survey conducted for this report, as well as in VDC's other census work on embedded engineering, industrial automation is the vertical market that accounts for the largest number of embedded software developers.

Factories can be especially challenging environments for software due to heterogeneous mixes of equipment from different vendors and different vintages, often using different communications protocols. And the failure of a single piece of equipment on a complex production line can bring the entire line to a halt, causing millions of dollars in lost productivity, as well as material scrap and possibly damaged equipment. Such facilities are also frequent targets of cyber attacks. Proper functioning and monitoring of device software is vital to keeping production lines up and running.

The variety of equipment that is in use varies from factory to factory, making it nearly impossible for device makers to fully test their software in the lab for every scenario it might encounter in the field. Monitoring when and how software defects occur or the conditions under which device software crashes is especially important for these systems. Despite this, nearly one-quarter of survey respondents working on industrial automation projects said their organizations were *not* collecting device performance and health information from products in the field. This may be contributing to costs and time needed to develop products.

Nearly 25% of industrial automation respondents said their organizations were *not* yet collecting device performance and health information from products in the field

Energy & Utilities

Energy and utilities systems are part of society's critical infrastructure. Power generation and distribution, fuel refining and distribution, and water purification and distribution, are all vital systems without which much of our lives would come to a screeching halt in a matter of days. That makes them prime targets for hackers, especially nation-state attackers. Many energy and utilities systems contain components that are decades old, and some are air-gapped such that they are not connected to any outside network. But as technology to monitor and control such systems continues to advance, Internet connectivity has become more of a requirement.

The electric power grid in particular is likely to go through considerable changes over the next decade, for two main reasons:

- Power generation shifting further to renewable sources such as solar and wind, and away from oil- and coal-powered plants
- The installed base of electric vehicles adding to total demand, as well as shifting demand across time of day as many vehicles charge at home overnight

Those factors will lead to many older power grids being updated and/or upgraded with newer systems.

Among survey respondents, embedded energy and utilities projects have the longest expected average lifespan in the field, at 11.4 years, and therefore they are most likely to need software updating to fix vulnerabilities or add functionality over their time installed. Energy and utilities projects also have the highest portion of their software code coming from in-house development, at 56.7% of total code. This may be due to the specialized needs of the market, but it also implies that the code may benefit greatly from remote performance and device health data collections.

Energy and utilities projects have the highest portion of their software code coming from in-house development, at 57% of total code

Consumer Electronics & Smart Home

These days, nearly every consumer product that is powered by electricity, from toasters to thermostats, includes one or more microprocessors, which also means they contain embedded software. Increasingly, those products also include Internet connectivity (with cloud software), as well as graphical user interfaces (with more software).

Although most consumers are less concerned about cybersecurity compared to industrial or commercial users, cybersecurity breaches can enable hackers to infiltrate entire home networks, snoop on video doorbells, disrupt heating and lighting systems, co-opt devices to operate as part of a DDOS network, among other nefarious deeds.

Consumers often review and rate these products based on the functionality of the software. Software bugs are among the frequent complaints, and they can cause both user inconvenience and brand damage. Collection of device health and performance data certainly can be valuable to help software teams find and fix bugs, which is especially important as consumer electronics products typically have the shortest development cycles, an average of 10.7 months, compared to 16.3 in the survey overall.

Interestingly, our survey also found that consumer electronics projects were the most likely to have product management teams use such data (51.8% compared to 38.9% for the survey overall), for example, to spot usage trends and identify opportunities for product improvements.

Consumer electronics projects are the most likely to have product management teams use device health and performance data, to spot usage trends and identity opportunities for product improvements

Summary & Conclusion

Software already accounts for more than half of development costs for most embedded products. Time-to-market pressures continue to ramp up further, feature and function requirements balloon, security vulnerabilities threaten to derail products in use, and software bugs discovered in the field can lead to customer dissatisfaction and potential loss of future product sales.

The complexity of today's software makes it infeasible to completely test the software under all the conditions it might encounter in deployed devices. A wide variety of software development tools are available to help engineers and developers find and fix bugs faster, improve overall software quality and performance, and provide valuable feedback into the product development process. One of the best ways to achieve all of these aims is to remotely monitor the performance and health of devices in the field. About 69% of survey respondents said they believed their organization adequately monitors the status of deployed devices for software bugs and security vulnerabilities. Yet, for the average product deployed, customers reported 19 software defects in the past year.

Third-party tools that actively monitor device performance and health can help organizations efficiently discover and address any software issues. About half the survey respondents said their organizations use in-house developed tools to monitor device performance and health in the field, and about one-quarter said they use third-party supplied tools. Organizations using third-party tools required one-third fewer person hours to remediate bugs. In addition, those organizations that had already been using the third-party tools were more likely to bring their products to market ahead of schedule, which we believe is due to direct feedback enabled over product iterations.

Organizations using third-party tools to monitor device performance and health in the field required one-third fewer person hours to remediate software bugs

Our research also highlighted the common disconnect between managers and engineers regarding the development processes, performance, and best practices. For example, 74% of managers believed that their organization adequately tested the cybersecurity of their products prior to deployment, but only 65% of engineers and developers said so. Yet in 73% of the organizations, managers and senior executives were responsible for deciding which software testing tools to purchase. Managers were also more likely to consider their organizations' projects to be running ahead of schedule, even though the engineers and developers did not always agree with them. Proper software tools can provide objective metrics, so that both groups have ready access to the same set of data and can make informed business and development decisions.

The challenges presented by software complexity will only be exacerbated in the coming years. As shown in this report, establishing best practices for software development can help reduce initial software development and subsequent maintenance costs, improve product quality, and bring software and bug fixes to market more quickly.

As IoT devices continue to be relied upon for more and more tasks—everything from critical infrastructure to entertainment—product makers have a responsibility to ensure those devices are functioning properly, for their own business efficiency as well as for the sake of their customers. This is especially important when devices can cause serious harm to society if something goes wrong.

About the Authors

Steve Hoffenberg is a leading industry analyst and market research professional for Internet of Things technology. He has more than two decades of experience in market research and product management for technology products and services. Prior to joining VDC, he spent 10 years as Director of Consumer Imaging and Consumer Electronics Research at the firm Lyra Research, where he led industry advisory services providing extensive market research on consumer technology trends, user adoption, market sizing, marketing strategy, and competitive analysis for major consumer electronics manufacturers. Previously, he worked in product management for electronic design companies that developed and licensed embedded digital imaging and audio products. Steve holds an M.S. degree from the Rochester Institute of Technology and a B.A. degree from the University of Vermont. He is also a Certified Information Systems Security Professional (CISSP). Email Steve at **shoffenberg@vdcresearch.com**.

Chris Rommel leads VDC's syndicated research programs 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 industrial and device markets, such as security, the IoT, and engineering 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. Email Chris at **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 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. For more information, contact us at **info@vdcresearch.com.**

About Memfault

Memfault is a leading embedded device observability platform that empowers teams to build robust devices with software at scale. Memfault gives embedded engineering teams an off-the-shelf observability solution with built-in over-the-air update management. It is designed for constrained devices and provides device performance monitoring, debugging, and other-the-air capabilities. To learn more, visit **memfault.com**.