Qt Group

WHITEPAPER

RESEARCH

The State of Medical Software Development

What medical software leaders reveal about development priorities, tooling gaps, and investment decisions



Introduction

Between April and June 2026, Gatepoint Research surveyed 50 leaders in software development, engineering, R&D and product management at medical device, diagnostics, healthcare technology and life sciences organizations. The study asked what they prioritize in development, how they test, which capabilities their tools lack, and what would release budget for better ones.

Qt Group commissioned Gatepoint Research to better understand the priorities and pressures shaping medical software development. We are sharing the findings because they offer a useful view of where teams are aligned, where current tools fall short, and which outcomes are most likely to justify further investment.

In the responses, the answers contain a mismatch. The capabilities this group weighs most heavily when choosing a development tool are also the capabilities they most often report as missing or insufficient in the tools they run today. Long-term support and support for medical safety and regulatory standards lead both lists.

A capability the tools do not provide is rarely a capability the work can do without. It reappears as testing, reporting and coordination carried out by people, somewhere else in the development process.

This report sets out where the mismatch comes from, what it costs while it holds, and how to make the case for closing it.

Contents

1. The Job that Never Ends 4
1.1 What Matters Most, And What Gets In The Way 5

2. What Teams Value, and Still Lack 6
2.1 What Decides a Tool Choice, and What Current Tools Still Lack 7

3. The Difference Is Paid In Hours 8
3.1 Where The Work Can Show Up 9

4. Why A Three Out Of Five Never Gets Replaced 10
4.1 How Teams Rate Their Tools, And What Would Make Them Invest 11

5. Where Do You Stand? 12

6. The Case Worth Making 13

7. Where Qt Group Fits 14

8. Survey Methodology 15
8.1 Notes 16

1. The Job That Never Ends

Ask this group what matters most in their development work and one answer dominates. Meeting all safety, regulatory and compliance requirements is the most important thing in medical software development today for **78%** of this group, well ahead of every other priority tested. Their biggest development challenge is long product lifecycles and hardware constraints on embedded devices, at **54%.**

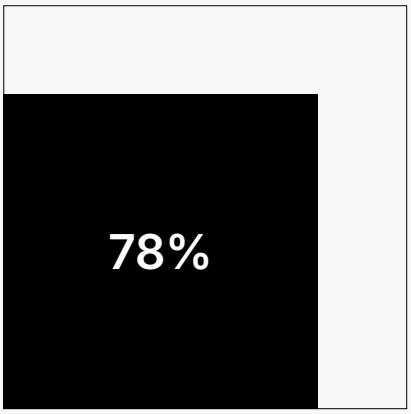
Who said it, matters. Respondents came mainly from engineering, R&D and product functions, so these are the priorities of the people writing and shipping the software.

Read together, the two results describe a decision with a long tail. A medical product's software carries safety and regulatory obligation for as long as the product is in clinical service, and the hardware underneath it is usually chosen before development begins. As a disclaimer worth mentioning, the survey did not measure how long medical products stay in service, so this report puts no figure on it.

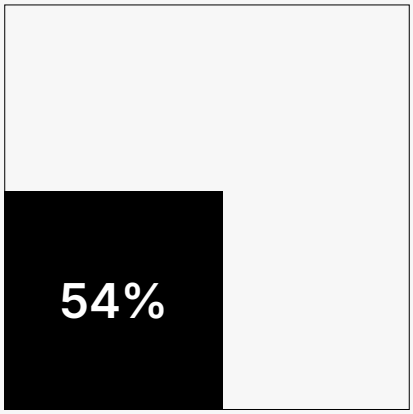
A tool decision for that job turns on three things: how long the tooling itself stays supported, whether changes can be verified repeatedly across the product's service life, and how challenging the platform choice is to reverse once software and documentation have accumulated above it.

A development tool in this market has to be judged against the service life of the products built on it, not only against the first release.

When the job runs that long, what do you look for in the tools underneath it?



78% prioritize meeting all safety, regulatory and compliance requirements

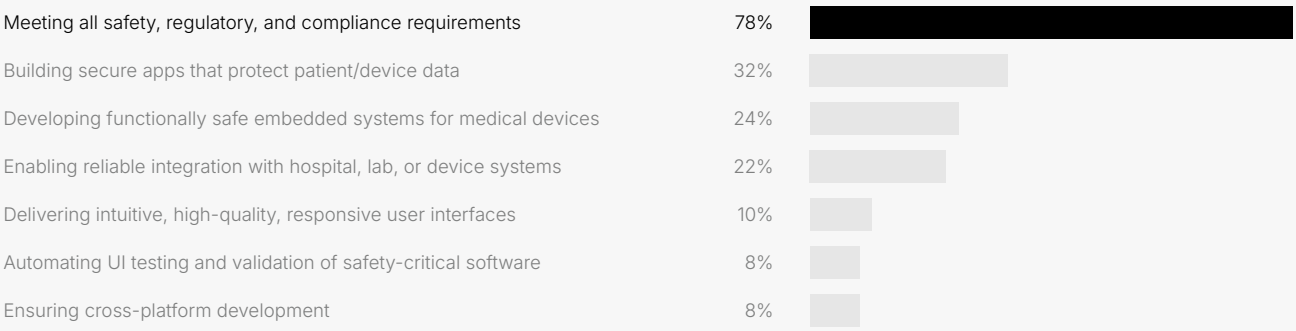


54% finds long product lifecycles and hardware constraints on embedded devices most challenging

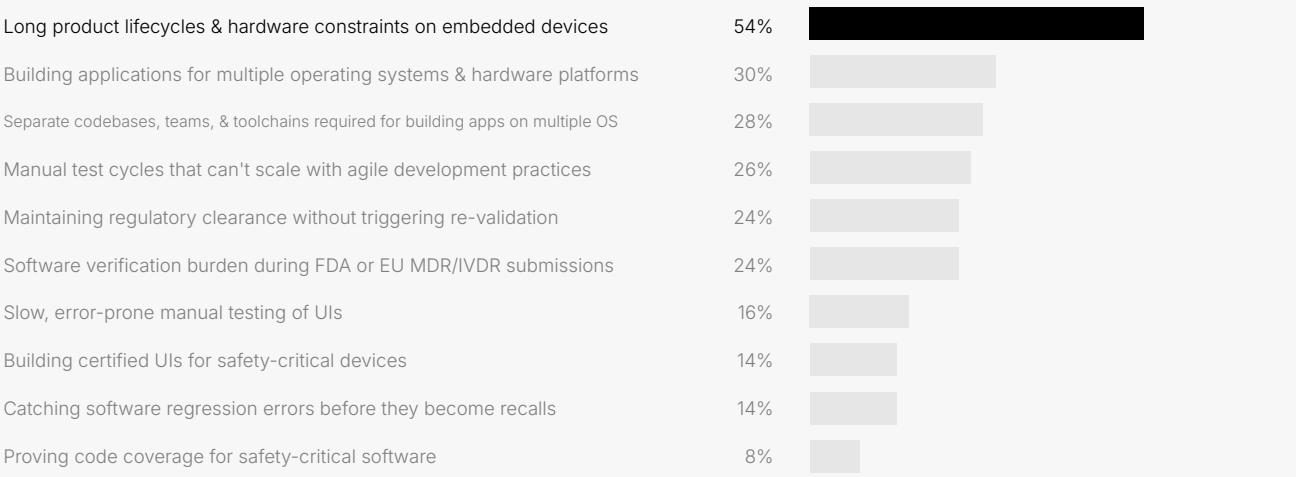
THE DATA

1.1 What Matters Most, And What Gets In The Way

Question: What is most important in your medical software development today?



Question: What are your biggest challenges with software development today?



Key takeaway: Compliance leads the priority list. Long lifecycles and hardware constraints lead the challenge list. Three of the challenges name manual testing or the verification burden at submission.

Chart Note: Both questions allowed multiple selections, so the columns do not total 100%.

2. What Teams Value, and Still Lack

The survey asked about capabilities twice: once as the factors that decide a tool choice, once as gaps in the tools teams already run. The same items sit at the top of both lists. Long-term support and support for medical safety and regulatory standards lead the selection criteria, and they also lead the capabilities respondents describe as missing or insufficient.

Neither is an optional feature. Long-term support shapes whether a development environment can be maintained through the years a regulated product stays in clinical service. Support for safety and regulatory standards shapes how much of the required verification and documentation work the tools help carry. A shortfall in either has to be covered another way.

The same two capabilities lead both lists, in reverse order. Teams recognize what the long product lifecycle requires, yet many still report those capabilities as insufficient in practice.

The consequence is operational. Work that the development environment does not support has to be completed elsewhere in the process.

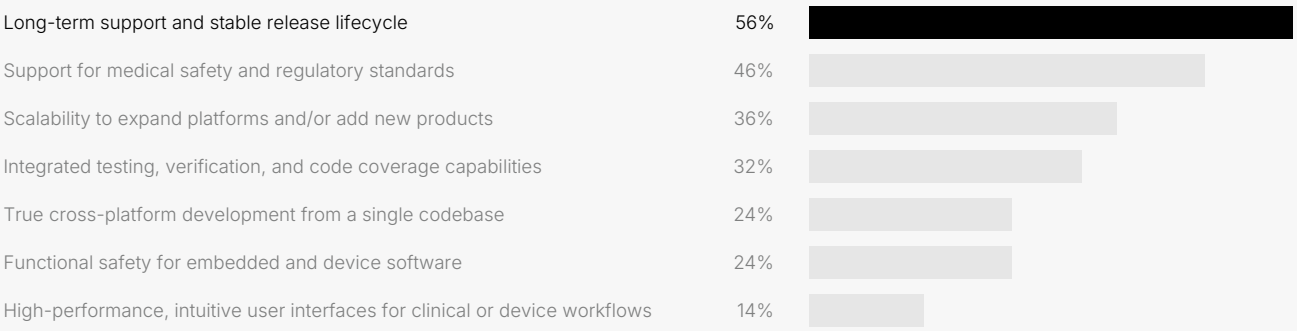
Cross-platform development appears more often as a reported gap than as a selection criterion. One possible explanation is that the requirement becomes more visible as a product family expands beyond its original targets.



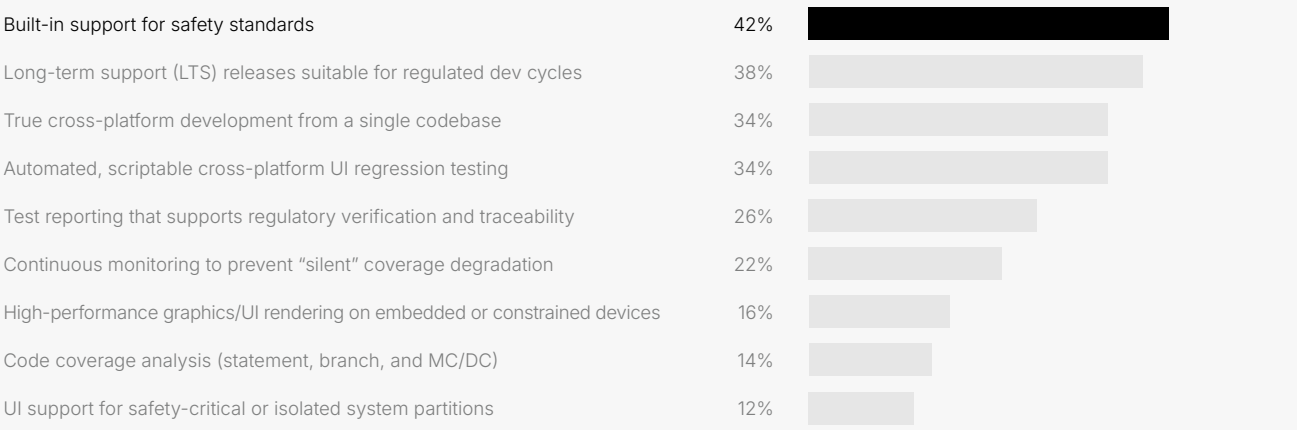
THE DATA

2.1 What Decides a Tool Choice, and What Current Tools Still Lack

Question: What factors are most important to consider when evaluating software development tools?



Question: What capabilities are missing or insufficient in your software development tools?



Key takeaway: The capabilities teams value most at selection remain among the ones most often reported as insufficient.

Chart Note: The questions used different option sets and both allowed multiple selections. The comparison shows a pattern across the respondent group, not paired answers from the same individuals.

3. The Difference Is Paid In Hours

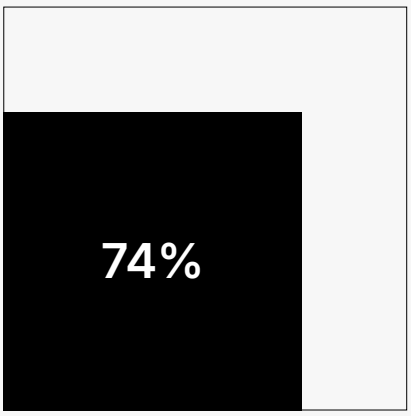
When a toolchain does not support part of the required work, teams complete it through existing roles and release processes. **The survey points to this pressure in two areas:** teams report gaps in automated UI regression testing and verification reporting, while also naming manual test cycles and submission-related verification among their development challenges.

Two of the reported gaps reappear on the challenges question. Automated, scriptable cross-platform UI regression testing is missing or insufficient for 34%, and 26% name manual test cycles that cannot scale with agile development practices among their biggest challenges. Verification reporting and the verification burden at submission pair the same way.

Because this effort appears across existing roles and schedules, it can stay outside the visible tooling budget. It is also spread across engineering, testing, quality and regulatory work, so no single function tends to see the whole of it.

This work and effort are real, they accumulate, and they are already being paid for. But it's distributed across working hours rather than visible as a tooling cost. That makes it harder to quantify and easier to leave out of the case for improving the development environment.

A second product built on the same development foundation can inherit the same manual steps unless the underlying capability gap is addressed.



When asked how challenging it is to find defects hidden in untested code paths, **74%** rate it a 3 or 4 out of 5.

None rate it 5. Moderate, persistent difficulty can be easier to absorb than to elevate into a tooling decision.

3.1 Where The Work Can Show Up

Capability reported missing or insufficient	Work that may remain	When it tends to recur
Automated cross-platform interface regression testing	The interface gets tested manually on each supported target	Release candidates, per target
Test reporting for regulatory verification and traceability	Verification records located and assembled from separate systems	Submissions and significant changes
True cross-platform development from a single codebase	Parallel implementation and parallel testing per target	New targets and new hardware classes

Estimate your own recurring effort: Take the people and hours involved in each manual activity, how often it recurs, and how many products or target configurations need a separate cycle. Count a product or target only where it creates separate work. The result is a planning estimate for your own discussion.

Table note: The first column is survey data. The other two describe work that can remain when a capability is missing.

4. Why A Three Out Of Five Never Gets Replaced

Compliance governs the work. Speed and economics decide the purchase.

Asked to rate satisfaction with their current development tools from one to five, 48% of this group chose the midpoint. More rated their tools above the midpoint than below it. Tools that are adequate instead of directly failing do not force a replacement decision, and the effort their limitations create stays spread across teams and release cycles, where it is harder to put a number on than a license line.

The investment question shows what would change that. The three outcomes most likely to spur investment in a new development tool are:

1. Reduced time to market
2. Streamlined development processes and reduced development time
3. Increased ROI

Every safety and regulatory outcome tested ranks below those three.

That sits oddly beside the result showing that 78% named safety, regulatory, and compliance requirements as the most important thing in their work. The survey asked about two different things though: **what the work has to satisfy, and what would justify allocating budget to it.** Meeting an existing obligation may be essential, but it is not always perceived as an additional business gain.

A stronger tooling proposal connects the regulatory need to measurable development outcomes: time to market, release predictability, and reuse and recurring effort.

THE DATA

4.1 How Teams Rate Their Tools, And What Would Make Them Invest

Question: How satisfied are you with your current software development tools?
(1 = not at all satisfied, 5 = extremely satisfied)

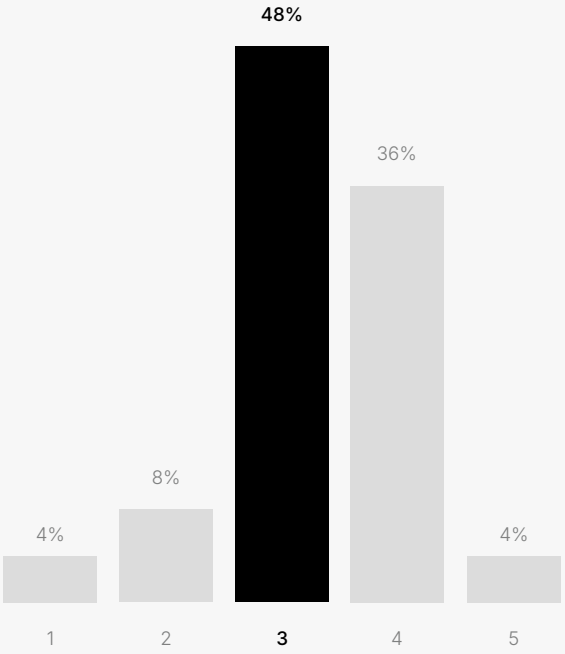
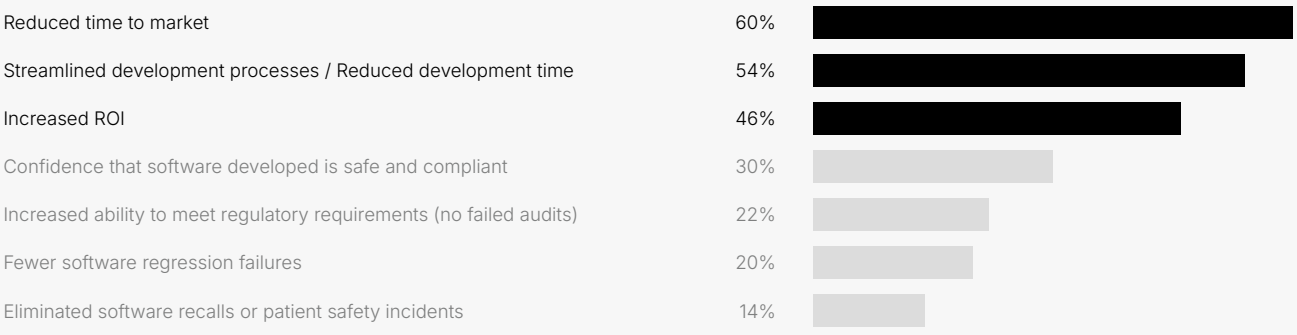


Chart Note: Satisfaction was a single-choice rating, so those five columns total 100%. The investment question allowed multiple selections.

Question: Which of these outcomes would spur investment in a new software development tool?



Key takeaway: Current tools are rarely rated as failures, while speed, efficiency, and return are the main reasons to consider alternatives.

5. Where Do You Stand?

The findings so far are 50 other teams' answers. These four questions help identify where the same pressures may exist in your own product portfolio.

1. **How much of the next product can reuse software and verified components already in service?**

   Estimate what can carry over without major redesign or re-verification. The answer reveals whether reuse is an architectural capability or simply an ambition for the next project.

   Survey benchmark: Scalability across platforms and new products ranks among the leading tool-selection factors.

2. **How many codebases, toolchains and test paths support your current target set?**

   The count shows how many parallel paths may need to be maintained when the software changes. It also helps identify where platform expansion has created duplicated development or testing work.

   Survey benchmark: More than a quarter report separate codebases, teams and toolchains for multiple operating systems.

3. **Which testing, reporting and traceability activities are repeated manually, and who measures that effort?**

   This identifies recurring work that may be spread across engineering, testing, quality and regulatory functions. Measuring it makes the effort visible enough to evaluate in a tooling proposal.

   Survey benchmark: Manual test cycles and insufficient verification reporting both appear prominently in the findings.

4. **When does support end for the framework and tooling in the oldest product still in service?**

   Knowing the support end date reveals when maintenance options may begin to narrow. Teams can then assess whether to maintain the current stack, negotiate extended support or plan a controlled move before the decision becomes urgent.

   Survey benchmark: Manual test cycles and insufficient verification reporting both appear prominently in the findings.

Together, the answers provide the starting point for a stronger tooling case.

6. The Case Worth Making

Safety and regulatory obligations are fixed. What varies is how much repeated testing, reporting and maintenance work is required to meet them, and how much of that work the development environment can support.

The survey reveals a clear tension. Teams value long-term support and support for safety and regulatory standards, yet many still report those capabilities as insufficient. At the same time, the outcomes most likely to secure investment are faster delivery, streamlined development and stronger return.

This changes how the case for further investment should be made. The regulatory need establishes why the foundation matters. The investment case should show how a stronger foundation can reduce recurring effort, improve release predictability and increase reuse across a product family.

The next product creates a practical decision point. Before its code, tests and dependencies accumulate, teams can evaluate whether the current foundation is suitable for the product lifecycle ahead, and whether it should also support the products that may follow.

So choose the foundation for the product lifecycle ahead, not only for the release in front of you.

7. Where Qt Group Fits

Qt Group commissioned the survey, while Gatepoint Research conducted it independently. The report interpretation is Qt Group's, based on the findings presented on the preceding pages.

From our perspective, the research reinforces three practical needs in medical software development: the ability to reuse work across suitable targets, make software quality activities more repeatable, and maintain a defined support path over a long product lifecycle. Qt Group supports these areas through the Qt application framework and its software quality tools

Builds on a shared application foundation

Qt supports application development across embedded, desktop and mobile targets, helping teams share code and design assets where the product architecture allows. Hardware, platform and product requirements still determine how much target-specific work remains.

Customer reference: Clarius builds handheld ultrasound across iOS, Android and its own hardware with about 1% of the code written for a specific platform, and reports around 25% shorter time to market.

Read the story here ↗

Makes software quality activities more repeatable

Axivion supports architecture verification and static analysis, Coco measures code coverage, and Squish automates GUI testing. Used within the development workflow, the tools provide repeatable results that support manufacturer-owned development and verification processes.

Customer reference: Oncosoft adopted Axivion, Squish and Coco alongside the Qt framework, and reports around 20% fewer interface defects.

Read the story here ↗

Maintains a defined commercial support path

A commercial relationship provides access to maintained releases, a route for maintenance and security issues, and supplier documentation that the manufacturer can use within its own processes.

Qt Commercial vs. Community Edition:
Explore the difference ↗

Discuss your product roadmap, tooling gaps, and support horizon with us

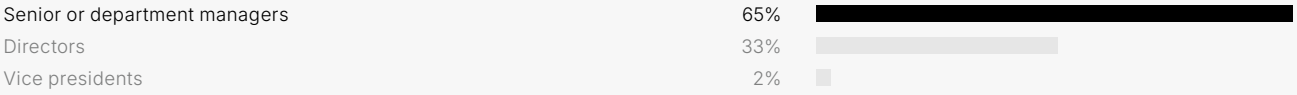
Learn more about Qt Group in MedTech ↗

Responsibility note: The manufacturer remains responsible for intended use, product requirements, risk management, architecture, validation, change control, regulatory strategy, submission and post-market obligations. Qt Group provides comprehensive tools and support within those manufacturer-owned processes.

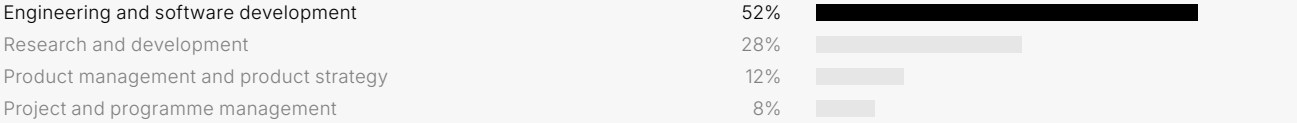
8. Survey Methodology

The survey, *Approaches to Medical Software Development Complexity Today,* was conducted by Gatepoint Research between April and May 2026. Select leaders were invited to fill it out via email, and 50 participated. The interpretation of the results in this report is Qt Group's.

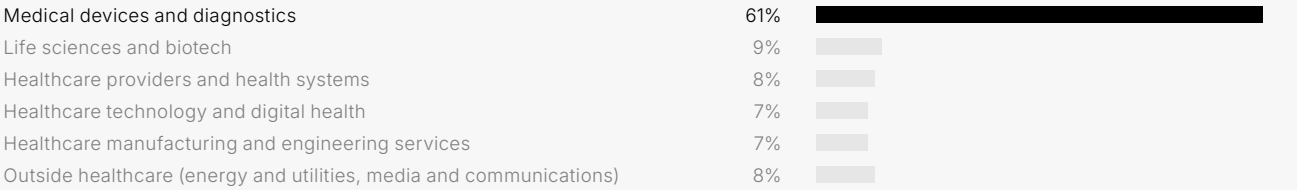
Seniority



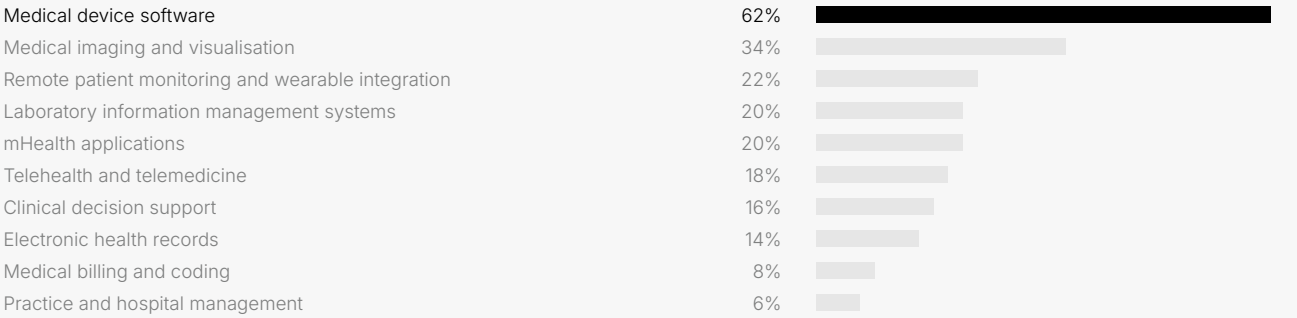
Functional area



Industry



What type of medical software is your organization developing today?



Company size: Annual revenues from under $250 million to $1.5 billion and above.

8. Survey Methodology: Reading Notes

Question types

Most questions allowed multiple selections, so those percentages do not total 100%. Two questions were single-choice ratings on a one to five scale, covering tool satisfaction and the difficulty of finding defects in untested code paths; those total 100%.

Sample size

At 50 respondents, two percentage points represents one person. Treat rankings within a few points of each other as effectively level.

What the survey did not measure

Working hours, financial cost, product service life, revalidation effort and return on investment. Where this report discusses recurring effort, it describes an operational pattern and offers an estimation method.

Securing patient and device data appears as a development priority and was not explored further. This report draws no conclusion about security readiness.

About Qt Group

Qt Group (Nasdaq Helsinki: QTCOM) is a global software company, trusted by industry leaders and over 1.5 million developers worldwide to create applications and smart devices that users love. We help our customers increase productivity through the entire product development journey: from UI design to software development, optimizing embedded systems, and quality management. Our customers are in more than 70 different industries in over 180 countries. Qt Group employs some 1100 people, and its net sales in 2025 were 216.3 MEUR. To learn more, visit **www.qt.io.**