MEDACUITY

KINOVA

UNLOCKING THE FULL POTENTIAL OF ROBOTICS DEVELOPMENT:

THE MEDACUITY & KINOVA PARTNERSHIP APPROACH TO REQUIREMENTS DEFINITION

A MedAcuity & Kinova White Paper

Executive Summary

Success in robotics development hinges on a robust Requirements Definition phase. MedAcuity and Kinova have partnered to deliver a unified approach, aligning software and hardware requirements from the outset. This collaboration minimizes rework, accelerates timelines, and ensures safety, scalability, and performance. This white paper explores the pivotal role of Requirements Definition, illustrates its impact through practical examples, and highlights the financial and strategic benefits of engaging our expertise early.

Introduction

Integrating hardware and software seamlessly is critical for safe, reliable systems in the dynamic fields of robotics and medical device development. Yet, poorly defined requirements often derail projects. The Project Management Institute notes that 37% of project failures stem from inaccurate requirements, while studies suggest up to 70% of projects falter due to inadequate requirements gathering. These gaps trigger delays, compliance risks, and cost overruns—challenges our partnership addresses head-on. By uniting MedAcuity's software expertise with Kinova's robotics prowess, we provide a cohesive Requirements Definition process, ensuring alignment across teams and delivering efficient, compliant solutions.

The Critical Role of Unified Requirements in Robotics Projects

In robotics development, success starts with well-defined requirements. Requirements serve as the design inputs for the entire project – they translate stakeholder needs (surgeons' needs, patients' safety, market objectives) into technical specifications that engineers across hardware, software, and systems can implement. If requirements are vague, conflicting, or incomplete, each engineering discipline might make different assumptions, leading to integration nightmares later. Regulatory frameworks like the FDA's design control process (21 CFR 820.30) require that requirements be clear, complete, and aligned with user needs, underscoring that this is a regulatory necessity, not just best practice.

Unified requirements ensure all disciplines—mechanical, electrical, software, clinical, and regulatory—work from the same playbook. In surgical robotics, for example, performance specs for motion, precision, UI workflows, and usability must all be aligned. If one team defines a requirement without cross-functional input, it can lead to late-stage redesigns or missed performance targets. A unified approach brings everyone together early to define what the system must do—and must not do—grounded in both engineering feasibility and user expectations.

From a project management standpoint, comprehensive requirements reduce scope creep, a major cause of budget and timeline overruns. Clear, measurable requirements improve communication, accountability, and predictability—especially vital in large, complex projects where risks multiply. Studies show that large projects are far more likely to fail, and a shared vision from the outset is key to avoiding that fate.

Ultimately, fixing issues during requirements is far cheaper than during development or after release. A few extra weeks upfront can prevent months of rework, costly redesigns, or recalls. Unified requirements offer high ROI by aligning teams, containing scope, and minimizing late-stage surprises, setting the stage for smoother, more successful development.

Company Strengths & Capabilities

MedAcuity: Leaders in System and Software Requirements Definition

MedAcuity specializes in software for safety-critical applications, excelling in defining requirements that ensure interoperability, cybersecurity, and regulatory compliance. Our rigorous upfront planning builds scalable, high-performance systems, rooted in extensive experience with medical devices, medical robotics, and non-medical robotic technologies.

"Supporting robotics firms—from cardiothoracic to neurological surgery—we've seen how solid requirements drive success. Tight timelines and honest mistakes often lead to mid-project fixes, inflating costs and delays. Getting it right early, with expert guidance, is invaluable."

— Shawn Vanseth, Robotics Practice Lead, MedAcuity

Kinova: Experts in Robotics and Hardware Requirements Definition

Kinova fast-tracks medical robotics with off-the-shelf and custom solutions, including medical-grade arms and actuators. Our hardware design and integration expertise ensures innovative, compliant systems that meet stringent safety standards.

"From startups to giants in laparoscopic and diagnostic robotics, we've learned that skipping requirements after a proof-of-concept risks setbacks. Our proven tools and certification savvy make partnering with us a smart launchpad for success."

— Francois Boucher, VP Business Development, Kinova

The Power of Partnership: A Unified Requirements Definition Approach

Misalignment between software and hardware teams often sparks integration issues and delays. Our partnership counters this with a collaborative process that ensures:

- Seamless system integration.
- Early risk detection and resolution.
- Clear team communication.
- Smooth transition to design.

This unified approach—visualized in Figure 1—lays a foundation for compliance and efficiency, integrating standards like IEC 60601-1 and ISO 14971 into a cohesive requirements spec from day one.

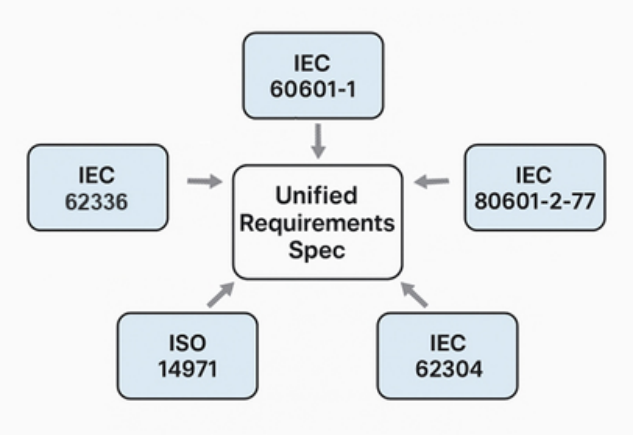


Figure 1: Unified Requirements Specification

Tackling Multi-Disciplinary Challenges with a Unified Team

Building a medical robot is the ultimate team sport. It brings together mechanical engineering (for structure and actuation), electrical engineering (for circuits and power), software engineering (for control, logic, UI), clinical experts (for understanding how the device will be used on patients), quality/regulatory specialists, and often industrial designers and human-factors experts. Each of these disciplines speaks a slightly different language and has areas of focus. One of the historical challenges in complex product development is ensuring these diverse teams remain synchronized throughout the project. This is where a unified requirements framework becomes a powerful collaboration tool. When MedAcuity and Kinova partner on a project, they form an integrated product team with the client.

Equally important is the broader medical robotics technology ecosystem—an invaluable asset for companies looking to accelerate development. Technology leaders like Kinova (robotic platforms), MedAcuity (software development), real-time communication, and real-time operating systems partner work synergistically to consult, collaborate, and deliver best-in-class solutions tailored to customer needs. This ecosystem model enables companies to tap into specialized expertise while maintaining a unified, high-performing development team.

Architectural Complexity of Medical Robotic Systems

Robotic systems are among the most complex electromechanical systems to design because they integrate a web of components – sensors, actuators, controllers, computing units, user interfaces, power management, and safety mechanisms – each with its requirements and constraints. Defining a system architecture for a medical robot is like orchestrating a symphony: each section (subsystem) must play its part harmoniously with the others. The requirements definition phase is the musical score that the orchestra follows. A clear set of requirements helps partition the system into subsystems and modules with well-defined responsibilities and interfaces, which is the essence of architecture. Conversely, a murky requirements set will lead to an incoherent architecture as engineers struggle to guess the intended behavior.

Let's consider a typical medical robotic system architecture at a high level. Taking the example of a surgical robotic platform (similar in concept to Intuitive's da Vinci system or other robotic-assisted surgical devices), we can identify significant subsystems:

- Surgeon Console (User Interface): This may include displays, control handles, or haptic input devices, foot pedals, etc., where the surgeon inputs commands and receives feedback. It runs user-interface software, possibly video processing for endoscopic cameras, etc.
- Robotic Manipulator Arms (Patient Side): One or more robotic arms with joints (each joint containing motors/actuators, encoders or other position sensors, maybe force sensors) and interchangeable surgical instruments (end effectors) perform the actual procedure on the patient.
- Core Controller/Compute Unit: This is the "brain" of the system that translates surgeon's inputs into coordinated motion of the robot arms. It often includes real-time control computers, safety processors, and communication interfaces between subsystems.

- Modern designs might distribute computing across the system, but logically, this is where control algorithms, kinematics, and safety checks run.
- Vision System: Often a 3D endoscopic camera system in surgical robots, with its own control and processing, feeding video to the console.
- Power Management and Safety Circuitry: Robotics requires significant power for motors, so there will be a power supply unit, battery backup, or UPS (for medical device uninterrupted power requirements), emergency stop circuits that can cut power to motors, etc.
- Ancillary Subsystems: e.g., networking components if the system has multiple units (console connected to robot via a network), communication protocols (possibly CAN bus in arms, or Ethernet using a real-time protocol, etc.), and sensor subsystems for any auxiliary sensing (like tracking systems or patient monitors that interface with the robot).

These subsystems must work together in real time and under strict safety conditions. A system architecture diagram for such a robot might show the surgeon console (with its PC and input devices) connected via a high-speed communications link to the robotic base, which contains multiple motor controllers (one per joint), all supervised by a master control system; a video subsystem sending feed to the console; and safety interlocks (like an E-stop button that instantly disables motion). Additionally, because it's a medical system, there may be a supervisory safety board that monitors the health of the primary controller (a typical pattern is a dual-CPU architecture – one primary, one safety watchdog that can intervene if the primary malfunctions).

Now, how do requirements influence this architecture? Fundamentally, architecture is the bridge between requirements and implementation. Early in development, as soon as high-level requirements are defined, system architects (from MedAcuity and Kinova in the partnership model) collaborate to propose an architecture that can satisfy those requirements. For example, if a requirement states, "The system shall support at least four robotic arms operating simultaneously with coordinated control," that immediately drives an architectural decision: the control system must handle at least four kinematic chains in parallel – implying a specific processing throughput, a multi-axis motion planning capability, and communication bandwidth to support it. If another requirement says, "The surgeon shall receive force feedback cues when predefined force thresholds are encountered by the instrument," this might necessitate adding force sensors at the instrument tips and a haptic feedback device at the console, altering the architecture to include those components and the signal processing they entail. A safety requirement like "An emergency stop shall bring the system to a safe state within 0.5 seconds" affects architecture by potentially requiring hardware interrupts or a parallel safety channel to cut power quickly, because handling that purely in software on the main CPU might not be reliable enough. Thus, requirements drive the inclusion or exclusion of components and determine how they interact.

In a unified requirements approach, architects ensure that each significant requirement is accounted for in the architecture. Typically, they will create a mapping (sometimes called allocation) of

requirements to subsystems. For instance, requirements about user interaction map to the Console subsystem; requirements about motion precision and speed map to the Manipulator and Control subsystems; safety requirements might map to a dedicated Safety subsystem, etc. This exercise often uncovers if any requirement is architecturally significant, meaning it broadly impacts the system structure. Those significant ones (like the number of arms example, or a requirement for modular upgradeability, etc.) must be decided early because they shape the whole project.

Complex robotic systems also often employ middleware and frameworks. For example, the Robot Operating System (ROS) is commonly used in research and increasingly in products (with ROS 2 offering a more industry-ready, real-time capable framework). ROS 2 provides a distributed architecture where different parts of the robot (sensors, controllers, algorithms) are decoupled into "nodes" that communicate over a data bus (medium.com). Suppose a team decides to leverage ROS 2. In that case, the requirements need to specify things like: communication latency requirements between nodes (e.g., "joint position updates shall be published at 500 Hz with <5 ms latency"), reliability ("lost packets shall not exceed 1 in 10^6 messages" if using DDS reliability), and so on. MedAcuity's software architects might contribute by selecting a communication framework (DDS, ZeroMQ, etc.) that meets these requirements. Kinova's hardware expertise ensures that the network linking the console and robot is robust (perhaps using EtherCAT or Time-Sensitive Networking(TSN) for determinism). In effect, the architecture selection of whether to use ROS or a custom real-time OS is guided by performance, scalability, and safety requirements.

We note that sensors and actuators also define part of the architecture: a requirement for high-resolution positioning may mean using optical encoders on motors (hence requiring an architecture supporting encoder feedback into control loops at a high rate). A requirement for portability might push toward a battery-powered design, meaning the architecture must include battery management and power distribution.

Manufacturers also face decisions on redundancy and fail-safe design as part of architecture. For life-critical functions, redundancy is often employed (dual sensors, dual processors). These decisions should tie back to requirements: e.g., "The probability of uncommanded motion due to a control system fault shall be <10^-6 per hour" – a requirement like that (perhaps coming from a hazard analysis) would in practice force an architecture with a redundant controller or at least a hardware watchdog that can override or stop the motion if the primary control deviates. An example architecture in some surgical robots is a two-channel system: a primary control computer issues motor commands, while a secondary safety PLC monitors joint positions and speeds. If the primary commands something that violates predefined safety limits (perhaps stored in redundant memory), the secondary system can intervene (cut power or limit the motion). Defining those limits and the need for a secondary channel would all stem from requirements derived from risk analysis. Without writing those down as requirements, an engineering team might not uniformly understand that such a safety architecture is needed. It could end up with a single-channel design that fails a later safety assessment, requiring painful re-engineering.

A unified approach championed by the MedAcuity & Kinova team means that system architects, hardware engineers, and software engineers collaborate during the requirements phase to sketch the architecture and ensure it can meet the requirements. This often involves rapid prototyping or analysis. For example, suppose there is a requirement for very high precision motion (say, sub-millimeter accuracy in instrument placement). In that case, the team might do calculations or simulations to ensure the combination of mechanical arm stiffness, sensor resolution, and control loop design can achieve that. If not, they either adjust the requirement (with stakeholder agreement) to a realistic level or decide on an architectural change (maybe adding active calibration or higher-grade components). It's far better to review that tradeoff in month 2 of the project than to discover the shortfall in month 20.

Examples: The Impact of Requirements Definition

Let's explore two contrasting examples that illustrate how the presence or absence of a comprehensive Requirements Definition phase can shape a robotics project's trajectory. These scenarios draw on common challenges in medical robotics development to highlight the value of early planning and collaboration.

Example 1: Maximum Outcomes Through A Deliberate Requirements Strategy

Consider a team developing a robotic-assisted surgical device with a proactive Requirements Definition process, guided by expertise like MedAcuity and Kinova. From the outset, the team conducts a thorough risk analysis, aligning with standards such as IEC 60601-1 and ISO 14971. They identify essential performance requirements—like precise positional accuracy—and integrate single-fault tolerance into the initial design. Electrical safety is also prioritized early, with patient-contact risks assessed and addressed through proper insulation and leakage current protections.

The payoff? Hardware and software teams work from a unified plan, catching potential issues during early prototyping, such as a software delay in responding to hardware faults. Testing efficiently validates the system's safety features, and regulatory submission sails through on the first try. Development time and testing costs drop significantly, and the product hits the market ahead of schedule, positioned as a reliable, compliant solution.

Example 2: Consequences Driven By Urgency Rather Than Intent

Now, imagine a different team building a similar system, but with a weaker upfront approach. Two critical oversights emerge:

First, the team skips an in-depth risk and failure mode analysis, prioritizing speed over thoroughness. Essential performance requirements, such as positional accuracy, aren't fully defined. Late in development, testing reveals the system lacks single-fault tolerance—a critical safety gap. Fixing this requires adding redundant components, triggering a cascade of changes: hardware redesign, software updates, and even tweaks to the industrial design of tightly integrated parts. The rework adds months to the timeline and racks up significant costs.

Second, the team delays a comprehensive 60601-1 analysis and risk assessment, focusing instead on early prototypes. They overlook the potential for patient contact, a key factor in electrical safety. When this surfaces during validation, the electrical architecture needs a complete overhaul—new insulation, revised leakage current protections, and different components. What started as a shortcut balloons into a costly detour, pushing the schedule back by nine months and adding over $2M in redesign and retesting expenses. Regulatory approval stumbles, requiring multiple submissions, and the delayed launch cedes market share to competitors.

These examples underscore a clear lesson: skimping on Requirements Definition invites inefficiencies, while investing upfront yields a smoother, more cost-effective path to success.

Development Lifecycle: A Strategic Framework

The project lifecycle underscores the value of early diligence. Figure 2 contrasts two paths: a robust upfront phase shortens overall timelines by minimizing late changes, while a rushed start extends rework cycles.

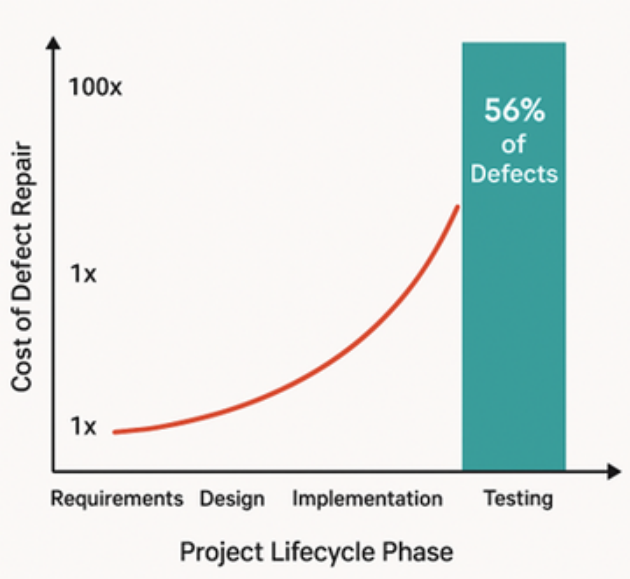


Figure 2: Project Lifecycle Phase Comparison

The ROI of a Strong Requirements Definition Process

Investing in a comprehensive, well-defined Requirements Definition phase is not just a best practice but the cornerstone of successful robotics development. The most significant advantage of addressing requirements early in the development lifecycle is the substantial Return on Investment (ROI) it yields across several critical areas: cost savings, accelerated time to market, improved product quality, regulatory compliance, and minimized risk. The data and examples below demonstrate the significant ROI of a solid Requirements Definition process, highlighting its financial and strategic benefits to organizations.

- **Cost Savings**: Up to 30% less rework by catching issues early [PMI, 2014]. By identifying risks like integration mismatches upfront, teams avoid expensive late-stage redesigns, saving significant labor and material costs that often escalate when flaws surface during testing or post-launch.
- **Reduced Time-to-Market:** 6-12 months saved via streamlined cycles [CycleLabs, 2020]. A clear roadmap from the start accelerates design and validation, eliminating delays from scope creep or unclear objectives, and positioning products to capture market opportunities ahead of competitors.
- **Improved Product Quality & Reliability:** 50% fewer failures with proactive design [IEEE, 2002]. Defining safety and performance needs early reduces defects, ensuring reliable systems that withstand real-world demands and minimize costly post-market fixes or recalls.

- **More substantial Regulatory Compliance:** 20-40% faster approvals with robust documentation. Comprehensive requirements aligned with standards like IEC 60601-1 and ISO 14971 streamline regulatory reviews, reducing submission iterations and speeding entry into regulated markets.
- **Mitigating Long-Term Risk:** Long-term risk can cripple any robotics project if functional safety and system requirements aren't defined clearly. Risks such as cybersecurity breaches, operational failures, and poor integration can undermine years of development, lead to costly product recalls, or even harm patients.

Conclusion: A Requirements-Driven Future

A comprehensive Requirements Definition phase, as modeled in the V-model (Figure 3), is the bedrock of robotics success. Our partnership aligns requirements with design and validation, ensuring efficiency and safety. Engage MedAcuity and Kinova early to unlock your project's potential—delivering innovative, compliant solutions faster and smarter.

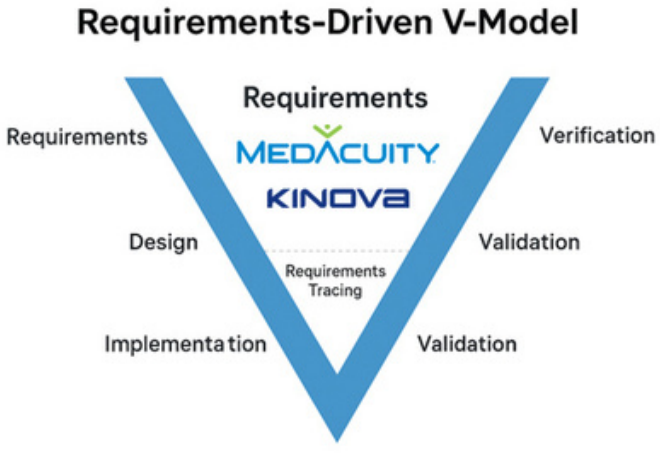


Figure 3: V-Model of Requirements-Driven Development

Next Steps:

For a consultation or workshop, contact us at svanseth@medacuity.com fboucher@kinova.ca.

For resources, visit our Robotics Development page

References:

1. PMI. (2014). Pulse of the Profession.
2. IEEE. (2002). Software Engineering Best Practices.
3. CycleLabs. (2020). Defect Cost Analysis.
4. Macenski et al. (2022). ROS 2: Design and Architecture.

About MedAcuity

About MedAcuity

MedAcuity, a premier software engineering firm specializing in the development of complex, software-intensive medical and robotic solutions. MedAcuity brings deep expertise in defining software and system requirements, as well as full lifecycle software development and verification. With a decade of experience in highly regulated and compliance-driven environments, MedAcuity helps accelerate innovation while ensuring safety, reliability, and regulatory alignment. Together, Kinova and MedAcuity empower medical robotics innovators with the technology and expertise needed to bring next-generation robotic solutions to market with confidence.

Contact info:

medacuity.com
Info@medacuity.com
866-376-1931

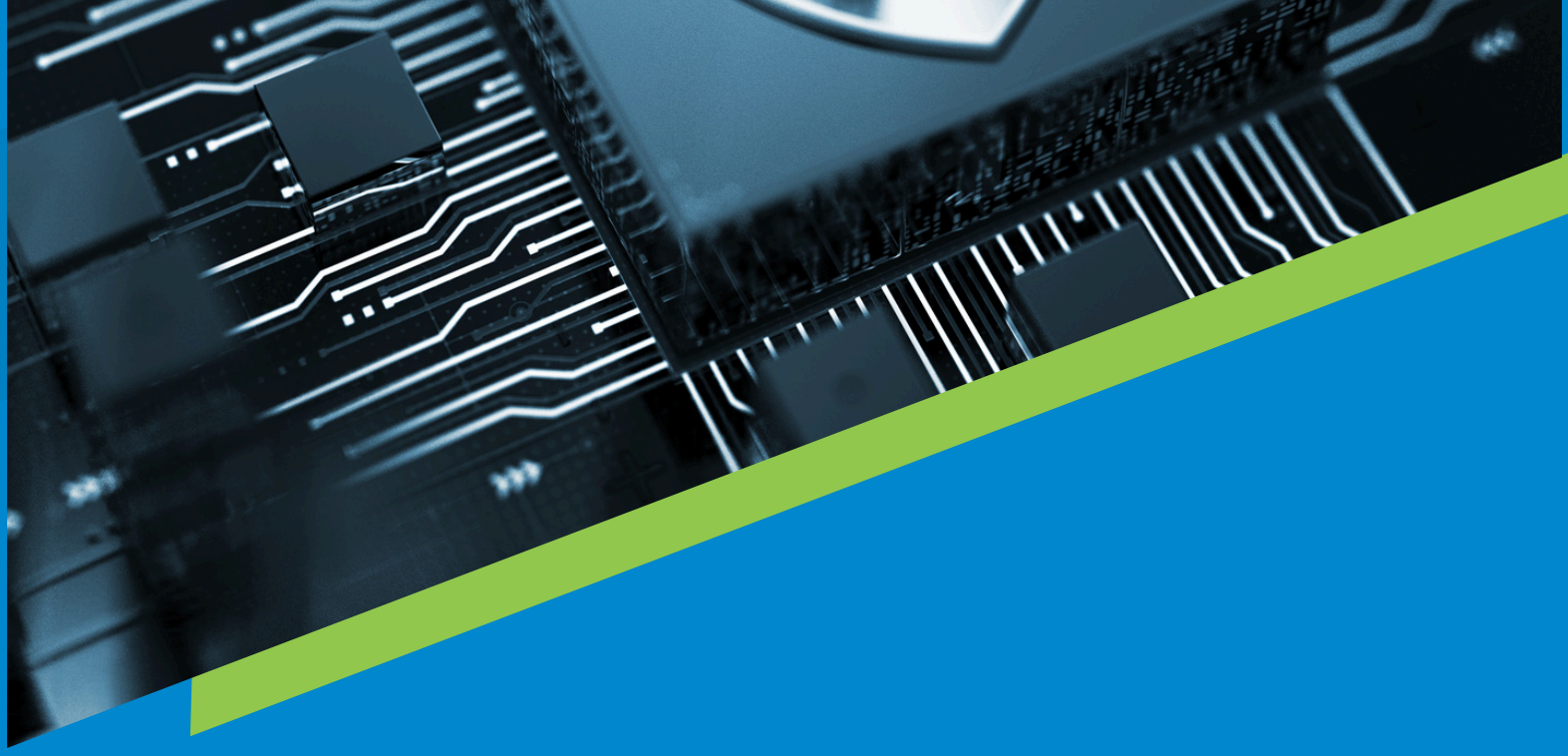
About Kinova

About Kinova

Kinova accelerates the journey to market for medical robotics companies by offering both off-the-shelf and tailored solutions for the development and production of medical-grade robotic systems. Through state-of-the-art medical arms, actuators, tool drives, and expert product development services, Kinova enables its customers to enhance their value proposition and bring innovative, high-quality solutions to life. For the development of a new medical device or the exploration of the next frontier in healthcare innovation, MedAcuity and Kinova can bring your vision to life.

Contact info:

Kinovarobotics.com
Info@kinova.ca
1-(514)-277-3777



MEDACUITY

medacuity.com
210 Littleton Rd. Westford, MA 01886
info@medacuity.com