Tech-Clarity

Requirements and Validation Engineering Buyer's Guide

The Expert Guide to Requirements and Validation

Tech-Clarity

Table of Contents

Executive Overview

Fierce global competition means companies have little leeway to bring the wrong product to market. However, today's products have gotten so complex, it has become increasingly difficult to capture customer and market needs, translate those needs to product requirements, manage them throughout a complex development process involving changes and different configurations, and then make sure the requirements are truly satisfied. A lot of work goes into the initial definition of those requirements, but we do not live in a static world. Changes are inevitable. Tracing all the impacts of those changes, notifying everyone involved, and getting everything updated, including the test case, can be a nightmare. However, with expert requirements and validation engineering practices, combined with the right technology, the process will be far more manageable.

Tracing all the impacts of those changes, notifying everyone involved, and getting everything updated, including the test case, can be a nightmare.

This guide consists of four major sections covering requirements management and validation software tool functionality, service requirements, vendor attributes, and special company considerations (Figure 1). Each section includes a checklist with key requirements to investigate when selecting software tools to support requirements and validation processes.

To set the foundation for expert requirements and validation engineering practices, companies should focus on the entire lifecycle of requirements, not just the definition. This will enable companies to ensure the product they want to bring to market, is the one they actually do. In addition, it will improve the efficiency of the entire process, with fewer errors, leading to higher product profitability.

To set the foundation for expert requirements and validation engineering practices, companies should focus on the entire lifecycle of requirements, not just the definition.

This guide is not an all-encompassing requirements list. It provides a high level overview for requirements and validation engineering.

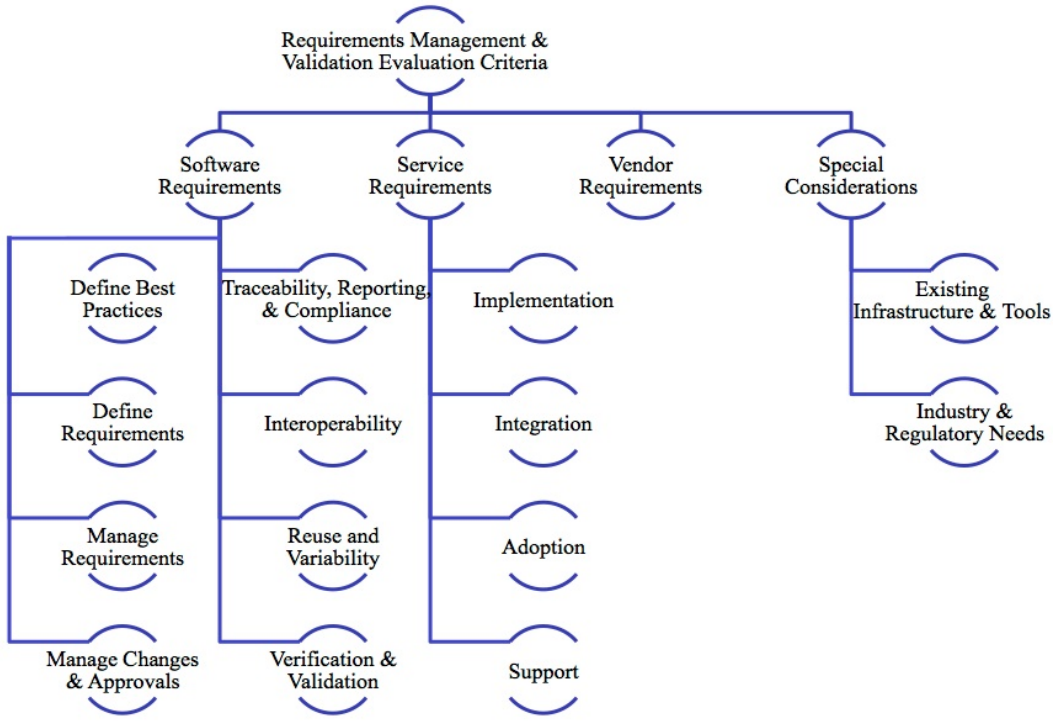

Figure 1: Requirements and Validation Engineering Evaluation Framework

The Business Value of Requirements Management

In today's global environment, companies face stiff competition. It is harder than ever for products to stand out. To be successful, it is absolutely critical that companies bring the right product to market. Meanwhile, development teams must balance competing demands for innovation, high quality, and lower cost within shrinking time frames. Making this even harder, products are growing in complexity. Tech-Clarity's research, Product Lifecycle Management Beyond Managing CAD finds that over the last five years, the top three areas that have increased the most in complexity are:

- Mechanical Design
- Configuration
- Mechatronics

Forty-six percent (46%) reported that mechanical design was a top area of increased complexity. This reflects the growth of more complex components as well as an increase in the number of components. An equal number, 46%, reported that configurations are a top source of increased complexity. This is attributable to both more variants and more complex configurations. By offering more configurations, companies can offer customers

more individualized products or meet different regional needs. Mechatronics are another common source of complexity. Mechatronics are products that consist of systems of mechanical and electrical components as well as software. Components that span multiple engineering disciplines are inherently interdependent. With many different engineering disciplines working on an individual product, each typically operating in its own silo of knowledge, product development becomes very complicated. The business impact of this increased complexity can be seen in Figure 2.

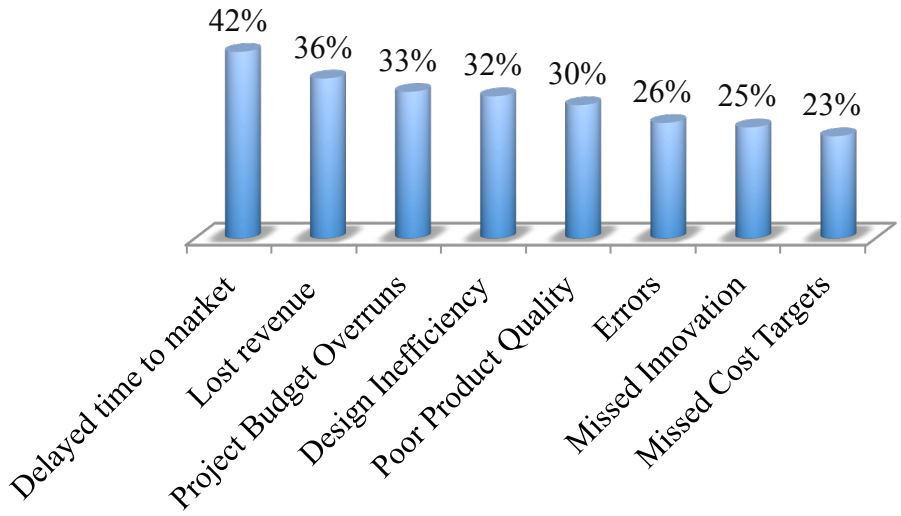

Figure 2: Impact of Increased Complexity (Last Five Years)

It is clear from Figure 2 that the impact of complexity drags profitability down. With more complexity, there is more risk things will go wrong, delaying time to market. Late to market means less time to collect revenue and increased risk of losing market share. The time spent fixing problems drives costs up and leads to inefficiency. Mistakes hurt quality, especially if they are not caught until later in the development cycle. This can tarnish a brand and may mean lost customers. All of these problems may mean cutting innovative features from the product, but losing innovation can also make the product less appealing which means more lost revenue. To be successful, and avoid this downward spiral, companies need to find ways to deal with this complexity.

The impact of complexity drags profitability down.

Let's look specifically at what is hard about developing today's complex products. Tech-Clarity's Developing Software-Intensive Products: Addressing the Innovation - Complexity Conundrum looked at the top challenges of developing software intensive products (Figure 3).

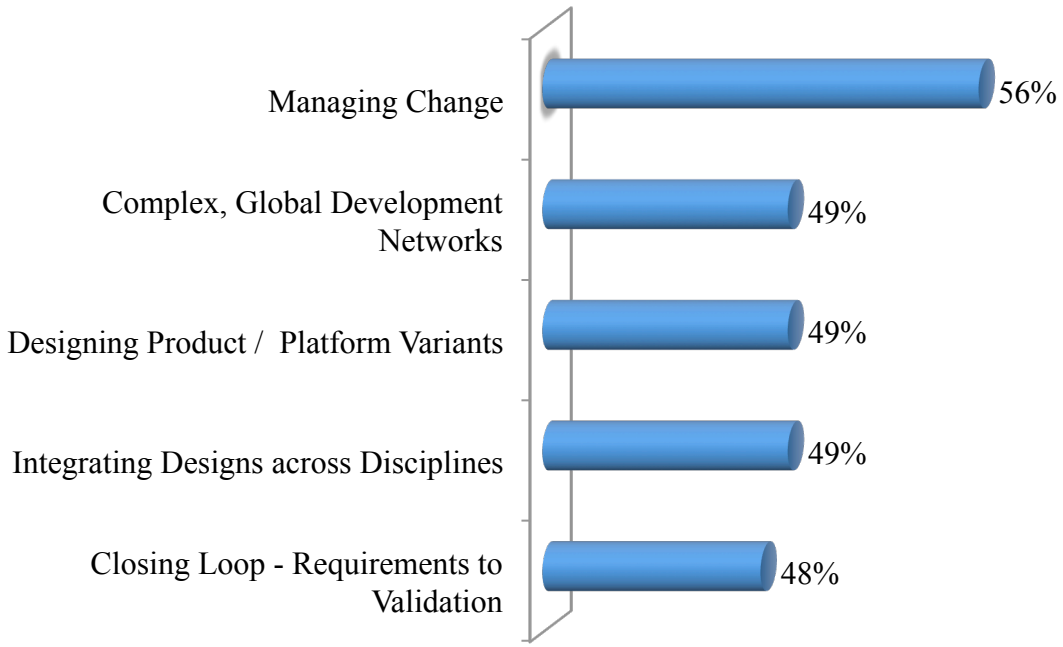

Figure 3: Challenges Developing Software-Intensive Products

Software-intensive products, also considered mechatronic products, are products that derive a significant amount of functionality and innovation from software. With that in mind, products that are purely software would experience these challenges as well.

Companies need agility to quickly respond to market changes as well as the ability to promptly correct conflicts and errors. While that means lots of change, added complexity makes it hard to understand the impact of these changes. Global development networks make it hard to collaborate and communicate changes as they happen. While the complexity of managing multiple configurations is challenging; ensuring that changes are correctly implemented across all variants becomes even harder. The need for visibility into work done by others, especially when there are interdependencies across components, creates additional challenges. Finally, making sure the product does what it is supposed to can be particularly difficult. Companies who are unable to solve these challenges will suffer from the impacts reported in Figure 2, which will hurt their profitability.

Many companies successfully avoid negative business impacts with excellent requirements and validation engineering practices.

Many companies successfully avoid these negative business impacts with excellent requirements and validation engineering practices. Requirements and validation

engineering involves defining, documenting, managing, and validating requirements. Technology plays an important role in enabling expert requirements and validation engineering practices. Technology can help companies address the complexity of today's products, by helping to capture requirements, enabling traceability, automating communications, improving visibility, and linking test results to the requirement they validate.

Start with Defining Best Practices

When dealing with the complexity of developing today's products, a lot can go wrong. However, capturing best practices and using them to define standard processes can be a great first step to improving efficiency. As seen in Figure 3, managing complex global development networks is a top challenge, but standard processes help the team understand what is expected of them and their peers.

Defining processes also helps identify those practices that work best and ensures they are consistently followed.

Clearly defined processes improve communication and productivity. Defining processes also helps identify those practices that work best and ensures they are consistently followed. Deploying consistent best practices will lead to a more efficient development team, which means getting to market sooner. This provides a competitive advantage as well as optimizes the window of opportunity to recoup development investments, both of which will improve profitability.

The role of technology to support the process is:

- Streamline and simplify process authoring
- Facilitate process adoption
- Provide governance to ensure it's followed
- Capture metrics to assess how well the process is working
- Enable continuous improvement
- Manage process and task ownership
- Integrate with product deliverables

The following is a list of software requirements that will support this and should be considered to support best practices for the requirements engineering process:

Requirement	Considerations
Authoring tools to capture best practices	Documenting best practices provides consistency that will lead to greater efficiency. The authoring tool should simplify the documentation process and automate as much as possible.
Automated best practices through automatic notifications and prompts for next steps	Process automation makes the processes easier to implement and follow. This leads to better adoption. Functions such as automated to-do lists and automated email notifications streamline handoffs, helping to overcome process bottlenecks and improve efficiency.
Process modifications	The ability to easily make changes leads to continuous improvement. Lessons learned can be applied which leads to efficiency and also captures processes that will lead to higher quality and lower cost.
Sub-process changes automatically applied to all other processes that use that sub-process	This leads to greater process consistency. It also saves time for process authors by minimizing manual updates. In addition, it supports continuous improvement to consistently apply and take advantage of lessons learned.
Library of existing processes available to define new processes or project specific processes	A library will save time because existing proven processes can be reused rather than creating them from scratch. Also, best practices learned on previous projects can be applied to new ones.
Project specific processes based on existing best practices	To save time, existing best practices should be copied to create project specific processes. Processes should be editable for individual project needs too. This ensures new project plans are based off of the latest best practices, but provides the needed flexibility to accommodate specific project needs.
Automated auditing	This ensures processes are followed as defined.

Figure 4: Functional Requirements for Defining Best Practices

Define Requirements

Requirements establish the foundation for a product. They provide guidance for development activity. The ideal solution for requirements and validation engineering

should make best practices executable and actionable. Part of this should involve making it as easy as possible to capture and document the requirements, but also take advantage of work previously done.

The ideal solution for requirements and validation engineering should make best practices executable and actionable.

The following is a list of software requirements to support defining requirements.

Requirement	Considerations
Easy to use authoring tool	Microsoft Word and Excel are the most common editing tools for defining requirements. Requirements management solutions that allow the continued use of these tools for requirement authoring will not require changes to current workflows, thus facilitating adoption.
Support for changes made in other authoring tools.	Importing changes made in Word and Excel enables those who are more comfortable with those tools to continue using them, yet their contributions are captured without manually entering the changes.
Flexible viewing options	When reviewing complex requirements, different viewing options such as lists and hierarchical can make it easier to define, manage, and collaborate on requirements as well as understand the relationships between requirements.
Support for previously created business based requirements	The ability to import business based requirements, typically created in MS Word or Excel, supports a single repository for all requirements, plus enables links between both business and technical requirements.
Support for previously created technical requirements	The ability to reuse previously created technical requirements improves efficiency. Technical requirements can then be distributed across different disciplines such as hardware and software while pointing to a single source of truth for all requirements.

Requirement (continued)	Considerations
Support for parameters within technical requirements	Parameters enable flexibility within the requirement definition. Parameter values can be changed once based on technical requirements and all requirements using that parameter will update. This can be especially useful when creating an updated version of an existing product or for different options within a product line.
Support for detailed and supporting information	For clear requirement definitions, sometimes supporting information is essential to clearly explain the requirement. Supporting information may include images, tables, attachments, attributes or other content.
Ability to identify untraced requirements	This ensures all requirements have a deliverable and test case associated to them.

Figure 5: Functional Requirements for Defining Requirements

Manage Requirements

The ideal solution for requirements and validation engineering needs to manage the complete requirement hierarchy from high-level requirements all the way down to low-level functional requirements. It should also manage who has access and what they can do.

The ideal solution for requirements and validation engineering needs to manage the complete requirement hierarchy from high-level requirements all the way down to low-level functional requirements.

The list below details key criteria to manage requirements.

Requirement	Considerations
Definitions for both top level and low level functional requirements	Managing requirement hierarchies helps manage complexity. Product quality is easier to maintain, while keeping cost low and streamlining compliance. Requirements created elsewhere should be importable.
Support for requirements across all disciplines including mechanical, electrical, software, and others	A single source of requirements across all disciplines reduces the chance of errors and improves communication.

Requirement (continued)	Considerations
Management of incremental baselines	Baselines help keep all teams working with the latest information which is important when working concurrently.
Requirements diagrams	Visual layouts of requirements make it easier to understand the relationships across requirements. The ability to connect a requirements diagram to a product model supports different design variants.
Requirements allocated to every element of the product model	Mapping requirements to the product model helps to manage complexity by visually managing interdependencies across the requirement hierarchy.
Access control to requirements	Access control allows flexibility over who can view, add, edit or remove a requirement. This protects sensitive information and ensures only authorized users can make changes.

Figure 6: Functional Requirements for Managing Requirements

Manage Changes and Approvals

As Figure 3 shows, the top challenge is managing change. Product complexity makes it hard to fully understand the impact of a change. There is risk that something impacted by the change will be missed or not everyone will be notified about changes that impact him or her. Impacts include not only design and development deliverables, but test plans and other requirements and specifications.

Poorly executed changes lead to wasted effort and rework, which adds extra time and drives up cost.

Poorly executed changes lead to wasted effort and rework, which adds extra time and drives up cost. Solutions that can manage and automate this process will improve communication and lower risk. Teams will then be empowered to make better decisions about changes and identify all the impacts.

The following is a list of software requirements to support managing changes and approvals.

Requirement	Considerations
Change impact analysis	When a requirement or related item is changed, the tool should identify its impact on other requirements and items. This saves the time to execute change orders and make it easier to understand the full impact of a change.
Version management and history for requirements	A version history makes it possible to understand what changes have happened. The version history should capture discrete changes to individual requirements as well as the complete collection, container, or document of requirements.
Version comparison tools	Visual tools that highlight differences between versions make it easier to analyze changes.
Audit trail of changes	An audit trail makes it easier to understand the changes that have been implemented, supporting better collaboration across the team. It can also provide valuable information such as who made a change and when.
Real time notifications of changes	This allows engineers to know immediately when their work has been impacted by a change, minimizing wasted effort and improving collaboration.
Conflict resolution	This supports better collaboration. Requirement authors can be notified of changes. When multiple users are editing requirements, it manages conflicts between submissions.

Figure 7: Functional Requirements for Managing Changes and Approvals

Enable Traceability, Reporting, and Compliance Management

Part of why requirements and validation engineering is so complex is because of the relationships and interdependencies across requirements and downstream deliverables. A lot of work goes into the initial requirements definition, but the process would be much simpler if requirements stayed static after they are defined. Unfortunately, that is rarely the case. Companies need to respond quickly to market changes and this will drive changes. Better ideas, new solutions, innovation, and design defects lead to further changes. Once there are changes, without an easy way to understand the relationships across the requirements, understanding the impact of a change is extremely difficult. This is where traceability becomes helpful. Traceability makes it possible to trace the

requirements from definition, to deliverables, to tests across all engineering disciplines and so that interdependencies can be identified. With traceability, it can be easier to understand the true status of a project and get reports on it.

Traceability also serves as the foundation of compliance reporting. Throughout the product lifecycle, significant documentation is required by both manufacturers and regulators in order to demonstrate the existence of design controls. Producing this documentation requires consolidation of key requirements and test documentation, with change control governing all assets.

Once there are changes, without an easy way to understand the relationships across the requirements, understanding the impact of a change is extremely difficult.

Figure 8 lists supporting capabilities for traceability, reporting, and compliance.

Requirement	Considerations
Traceability across requirement hierarchy and interdependencies	The ability to trace requirements from the high level down to the detailed design makes it easier to understand the impact of changes and implement them. It's also easier to identify the root cause of problems found during validation. Managing and developing configurations is also simpler with end-to-end traceability.
Traceability across downstream deliverables, design, implementation, and test.	Traceability across requirements, design, implementation, and test is needed to properly understand the impact of changes and ensure all impacted components are updated. Traceability is also needed for regulatory compliance.
Requirement approvals	Companies that must be in compliance with regulations such as FDA 21CFR or need to track the approval process should consider solutions that offer the needed audit trail, complaint e-signatures, and other functionality that will support compliance requirements.
Querying, charting, reporting, and dashboard tools	This empowers management and others with easy visibility into progress and enables access to project status without disrupting the workflow of others. Status reports can include if the requirement has been met, by who, and when.

Requirement (continued)	Considerations
Compliance Reporting	The ability to provide auditable change records with complete traceability is key to satisfying regulatory reporting requirements.

Figure 8: Functional Requirements for Traceability, Reporting, and Compliance Management

Ensure Interoperability

Requirements and validation engineering is such a complex process, it is important that previous investments are protected, especially when parts of the process are working well. A solution with good interoperability, allows companies to leave what is working well intact, and build upon it to optimize the process. In cases where a solution replaces an older solution, it is important that that work can be reused. Good interoperability should have support for other formats so that work is not lost. Companies that collaborate with third parties such as suppliers, partners, and customers may receive work in different formats. An interoperable solution should support collaboration with those working with other solutions.

Requirements and validation engineering is such a complex process, it is important that previous investments are protected, especially when parts of the process are working well.

Software capabilities required to support interoperability are listed below.

Requirement	Considerations
Integration with other requirement management tools	This allows the flexibility to take advantage of investments in existing tools that are working well within an organization.
Bi-directional integration with other requirement management tools	Bi-directional integration means changes in one solution can be seen in the other and vice versa. This is important if multiple solutions are used so time is not wasted entering information into two systems and there is no risk of working with outdated, conflicting, or incorrect information.
Integration with other enterprise systems	Integration with enterprise systems such as PLM makes it easier to have traceability from the requirement to the design assets.

Requirement (continued)	Considerations
Support for industry standards	Standards such as Requirements Interchange Format (ReqIF/RIF) can be used to exchange requirements across different requirements management solutions from different vendors. ReqIF/RIF is an Object Management Group (OMG) standard.
Application programming interface (API) for integrating with other tools	Support for options such as standards-based interfaces, JavaTM, SOAP, or RESTful web services enables data to be exposed to other systems.
Interoperability with domain specific design tools	This enables links between the requirements and the design deliverables, regardless of which engineering domain satisfies the requirement.

Figure 9: Functional Requirements for Interoperability

Support Reuse and Variability

Reuse helps to reduce time to market by reducing design, development, and test time. In addition, with complex products, reusing proven and validated work means fewer errors. By reusing requirements, companies can save time by avoiding recreating existing work.

Another aspect of reuse is design variability. As Figure 3 shows, 49% report working with design variants as a top challenge. Developing a design platform and or design variants means reusing different combinations of components based on requirements for an individual configuration.

With complex products, reusing proven and validated work means fewer errors.

Figure 10 lists software capabilities to support reuse and variability.

Requirement	Considerations
Requirements reuse	Reusing requirements saves time. Consider how sophisticated the relationships between reused requirements should be. For example, when reusing a requirement, should there be a link to the original requirement? When changing a requirement, should that requirement be flagged everywhere it was reused?

Requirement (continued)	Considerations
Requirements reuse	Reusing requirements saves time. Consider how sophisticated the relationships between reused requirements should be. For example, when reusing a requirement, should there be a link to the original requirement? When changing a requirement, should that requirement be flagged everywhere it was reused?
Library of reusable requirements, linked to appropriate deliverables	A repository of existing requirements facilitates reuse, which saves time. A library with requirements linked to components with defined variants makes it easier to reuse in the future.
Search tools to find reusable requirements.	Functions such as indexing, classification, and tagging help make it easier to quickly find requirements that can be reused, saving both design and test time.
Library publishing tools	The ability to publish subsystems, their interfaces, components, use cases and requirements from models and interface files enables greater reuse.
Dashboards to measure reuse	Metrics help to identify what's working well and what is not. Tracking how often requirements are reused measures the effectiveness of the reuse strategy and enables continuous improvement of the library.
System model drives the creation of different product variants	With the system model as a visual guideline to create different product variants, it is easier to manage complexity. Also, by using a system model, it will be easier to visualize the impact of changes across product variants.
Variation options and variants are publishable to reuse library	This will aid in the creation of additional product variants and further enable reuse.

Figure 10: Functional Requirements for Reuse and Variability

Verify and Validate Requirements

Validation involves ensuring the right product is built. Verification confirms the product has been built right. With complex products, ensuring the final product does what it was originally intended to, is not easy. In fact, 48% report that closing the loop between requirements to validation is a top challenge.

Validation should be a continuous process throughout the development cycle.

Finding problems as early as possible is an important piece of making sure the product is right. Validation should be a continuous process throughout the development cycle. The ability to do this will reduce the risk of finding problems at the end, when they are more expensive to fix and can put product launch dates at risk. In addition, regular testing will mean higher quality products that will be more likely to be successful in the market. Functional requirements that support early and regular validation are listed below. These functions will provide companies with the ability to continuously validate they are building the right product, early and often, throughout the development lifecycle.

Requirement	Considerations
Test planning, definition, and authoring integrated with requirements definition	Requirement definitions are more likely to be testable when linked to a test plan. Associating each requirement with a test also leads to better test coverage, and makes it easier to validate requirements are met. The final product is then more likely to meet customer and market requirements.
Traceability between test cases and requirements	Linking each requirement to a test supports better test coverage as requirements without tests can be flagged. In addition, when a test fails, it will be easier to understand the impact on the requirements and the product.
Tests executed in context of individual component configurations	This supports product line engineering by providing tests not just for the main platform, but also for the product variants.
Central repository for automatically capturing test results	Central repositories provide stakeholders with real-time visibility into test coverage, findings, and defects for insight into issues impacting time to market, quality, and cost targets.
Interoperability between test case and analysis tools	This enables traceability between the requirement, test case, and analysis results.

Figure 11: Functional Requirements for Validation and Verification

Regular testing will mean higher quality products that will be more likely to be successful in the market

Assess Service Requirements

To achieve a return on investment, the software must first be implemented and then especially important, users must be able to adopt the software. Users need to feel comfortable with the software to use it. This is achieved through good training resources as well as support when users have questions or run into problems. Finally, given the number of engineering disciplines typically involved in product development, requirements and validation engineering tools should be integrated with other solutions. Integration will provide a more seamless workflow across engineering disciplines and will help to overcome the natural knowledge silos that exist within each engineering discipline.

Integration will provide a more seamless workflow across engineering disciplines and will help to overcome the natural knowledge silos that exist within each engineering discipline.

The following list provides support requirements to ensure users are able to take advantage of the software investment:

Requirement	Considerations
Qualified trainers	To ensure adoption, trainers who are familiar with the tool(s) and standards should be available to users.
Dedicated support staff	When help is needed, users should have a resource available to provide assistance. Assistance should be available from experienced engineers. Ensure that support hours align with when users would need it.
Documentation of tool installation, administration, and operation.	In addition to phone and consulting support, documentation should also be a resource for help.

Figure 12: Service Requirements

Consider Vendor Attributes

When making an investment into something as strategic as requirements engineering, it is important that the software vendor has the ability to be a trusted partner.

The following list provides criteria for what to look for in a vendor.

Requirement	Considerations
Financial stability	Is the vendor profitable? Are they making investments in R&D?
Support for industry standards	Given the number of tools involved in developing products, the ability to share information across those tools will help coordinate the development process. Support for industry standards makes it easier to integrate those tools.
Invests in products	Is the vendor continuously investing in its portfolio to improve and expand it?
Has broader solutions available	Requirements engineering is very broad, so tools must integrate with other engineering disciplines. Does the vendor offer extended product development solutions such as PLM and/or ALM that will scale to your needs or are there options to integrate with another vendor's solutions?
Willing to partner with other vendors	Because requirements engineering crosses multiple engineering domains, there will be need to integrate with other competing / complementary tools.
Flexible licensing	Licensing schemes such as floating licenses that are released when a user exits the tool give users flexible access to the tool, when they need it.
Experience	What kind of experience does the vendor have solving systems engineering problems? Do they have expertise in your industry?

Figure 13: Vendor Requirements

When making an investment into something as strategic as requirements engineering, it is important that the software vendor has the ability to be a trusted partner.

Identify Specific Needs for your Company

Requirements engineering is complex, with many different functions involved. In some cases, your company may have existing point solutions that are working very well. In this case, it is important to consider other solutions you may need the tools to integrate with. In addition, your company or industry may have specific regulations you need to comply

with. It is important to consider the specific needs of your company. For example, what does your environment look like? Which programming languages do you need? Which CAD tools are used?

The following list provides a reference to think through the special considerations your company will need.

Requirement	Considerations
Existing enterprise tools	Will you need integrations with PLM, ALM, etc?
Existing design tools	What are you using for CAD, EDA, Simulation, and IDE?
Existing software infrastructure	Consider which existing tools must be integrated such as configuration management, data management, other requirements management solutions, engineering performance and analysis tools, modeling tools, etc.
Support for your processes	In cases when your process is not supported out of the box, how easy is it to customize the solution to support your process?
Localization	Are the languages your global team needs supported?
Software languages	Will you need support for C, C++, C#, Ada, Java, VB, etc?
Scalability	What size projects must it support and can it scale across projects of different sizes?
Regulatory compliance	Which regulations must you comply with: ISO 26262, DO-178B, DO-178C, DO-254, SPICE, IEC 61508, IEC 62305, AUTOSAR, etc.

Figure 14: Requirements for Specific Company Needs

Conclusion

Expert requirements and validation engineering practices are key to developing the right product to meet market needs. However, once the requirements are defined, the job is not done. Companies must be able to quickly respond to market changes, take advantage of better ideas, and efficiently manage product platforms and variants. This means that change will be inevitable. However, today's products are so complex, identifying all impacts of a change is difficult. This includes identifying who is impacted as well as all

the impacted dependencies. This can be an overwhelming amount of work that is destined to have errors. Not managing this well leads to market delays, cost overruns, and lost revenue opportunities. However, with the right requirements and validation engineering practices, combined with the right technology solution, this error-prone process can be streamlined so that companies bring the right innovations to market, on time and on budget.

With the right requirements and validation engineering practices, combined with the right technology solution, this error prone process can be streamlined so that companies bring the right innovations to market, on time and on budget.

However, there are so many aspects of requirements and validation engineering, it can be very difficult determining the right solution for your company. Using a high-level list of tool and process evaluation criteria such as the ones in this guide can help narrow down potential solutions by providing a quick "litmus test" to determine if a solution and partner are a good fit before conducting detailed functional or technical reviews. In the end, it's important to ensure that functionality, service, vendor, and special requirements are all considered when selecting a solution.

Recommendations

Based on industry experience and research for this report, Tech-Clarity offers the following recommendations:

- Identify software requirements based on company needs, existing applications, industry, and unique product and process requirements
- Use high level requirements such as the ones in this guide to evaluate solutions based on business fit before engaging in detailed evaluations
- Consider long-term business and process growth needs and the potential to scale across product lines, departments, and engineering silos
- Consider all aspects of requirements and validation engineering solutions from definition, management, changes, traceability, interoperability, reuse, variability, and validation.
- Select a vendor who has the ability to be a trusted partner

About the Author

Michelle Boucher is the Vice President of Research for Engineering Software for research firm Tech-Clarity. Michelle has spent over 20 years in various roles in engineering, marketing, management, and as an analyst. She has broad experience with topics such as product design, simulation, systems engineering, mechatronics, embedded systems, PCB design, improving product performance, process improvement,

and mass customization. She graduated magna cum laude with an MBA from Babson College and earned a BS in Mechanical Engineering, with distinction, from Worcester Polytechnic Institute.

Michelle began her career holding various roles as a mechanical engineer at Pratt & Whitney and KONA (now Synventive Molding Solutions). She then spent over 10 years at PTC, a leading MCAD and PLM solution provider. While at PTC, she developed a deep understanding of end user needs through roles in technical support, management, and product marketing. She worked in technical marketing at Moldflow Corporation (acquired by Autodesk), the market leader in injection molding simulation. Here she was instrumental in developing product positioning and go-to-market messages. Michelle then joined Aberdeen Group and covered product innovation, product development, and engineering processes, eventually running the Product Innovation and Engineering practice.

Michelle is an experienced researcher and author. She has benchmarked over 7000 product development professionals and published over 90 reports on product development best practices. She focuses on helping companies manage the complexity of today's products, markets, design environments, and value chains to achieve higher profitability.