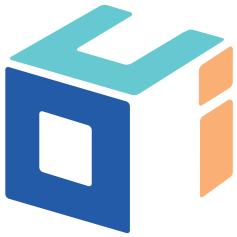


OBJECT COMPUTING

RE!MAGINE TOGETHER

9 Myths of Software Requirements Gathering

INTRODUCTION

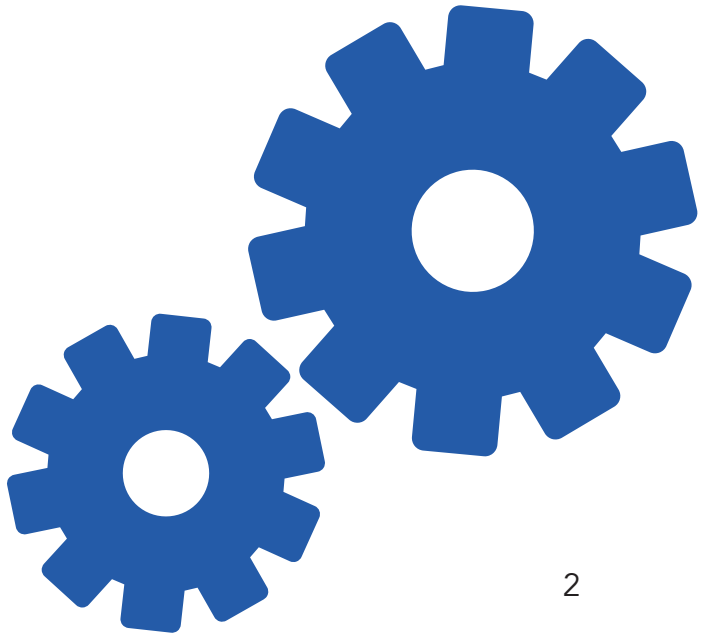
TOP 9 MYTHS OF SOFTWARE REQUIREMENTS GATHERING

Requirements gathering (or "requirements elicitation") is a critical process for any successful software development project.

If you fail to clearly define the key features of the product to be built, your team runs the very real risk of spending countless hours developing a system or app that fails to meet client expectations and user needs.

The more time and effort you put into requirements gathering, the more likely it is that development will progress smoothly and your deliverables will satisfy all stakeholders.

So, whether you're just getting started or you're struggling with your current processes, here are nine myths about requirements gathering that you can let go of, along with expert tips that will help you move forward with more confidence and manage your projects more successfully.



MYTH 1

Myth: If you work for a large organization, you need sophisticated tools and a complex requirements-management methodology.

Reality: Your mission, your organizational culture, and the intended outcome should be the prime considerations in selecting requirements-management tools and methodology.

The corollary to this myth is that if you work for a small organization, you need simple tools and a simple requirements-management methodology.

Although it is important to have requirements-gathering tools and methodologies in place, the size of your organization should not dictate the ones you use. Evaluate the tools and methodologies available and select ones that fit your team's needs.

Some features to look for in a requirements-management tool include the following:

- Requirement statement storage
- Requirement statement organization (by category, type, and other attributes)
- Ability to attach or reference external artifacts (documents, pictures, etc.)
- Versioning
- Prioritization
- Consistency enforcement
- Requirement gap identification (e.g., "to be defined" areas)
- Traceability
- API interfaces
- Testing tool interfaces

Some pitfalls to avoid in a requirements-management tool include the following:

- Letting a requirements-management tool become a burden; if it's difficult to use, it's the wrong tool for your team.
- Restricting access to your requirements-management tool; all stakeholders should have access.
- Failing to capture critical raw data (electronic info, paper docs, whiteboard, audio, etc.)
- Thinking that you can manage your requirements solely with spreadsheets.

MYTH 2

Myth: Requirements are contributed solely by the business stakeholders who pay for the product or application.

Reality: Requirements may come from numerous sources.

A corollary to this myth is that requirements are contributed only by users of the product or application. While business stakeholder viewpoints are important, it's risky to rely on them alone. The business stakeholders' vision of the product is likely to be incomplete.

In fact, almost any stakeholder you ask is likely to provide an incomplete list of requirements. This is because each person views the product through a different lens. Any reasonably sized product or application is likely to have numerous stakeholders, each with a unique perspective that can provide real value.

For example, you may find that your operations or technical folks can provide excellent business requirements, or you may find that your business stakeholders are able to articulate important nonfunctional requirements that could have been missed otherwise.

Also, keep in mind that different types of requirements can come from unexpected sources. In fact, it might seem counterintuitive at first, but not all sources of requirements are people.

Some excellent sources of requirements – both human and nonhuman – include the following:

- Use case diagrams
- Competitor analysis
- User or consumer observation
- Existing documentation
- Existing solutions and tools
- RFP/RFD documents
- One-on-one interviews
- Group interviews and focus groups
- Structured workshops
- Brainstorming
- Surveys and questionnaires
- Prototyping

MYTH 3

Myth: Only "technical people" can write or contribute requirements to a software project.

Reality: People with expertise in a wide variety of disciplines can (and should) contribute.

As a follow-up from the last myth, it's just as important to avoid relying solely on your development team for requirements.

When gathering requirements, different perspectives matter. Cast a wide net to identify potential stakeholders in many areas of your organization, and don't discount contributors from outside your organization.

Possible contributors could come from any or all of the following departments or groups:

- Business Leadership
- Legal
- IT
- Development
- Security
- Analytics
- Operations
- End users
- Independent third parties (outside experts)
- Research and Development
- Engineering and Maintenance
- Health and Safety
- Audit and Regulatory Compliance

MYTH 4

Myth: Democracy is the best way to identify and prioritize requirements.

Reality: Majority rule is probably the least productive way to manage requirements. Consensus is a close second.

When it comes to gathering and prioritizing requirements, giving everyone an equal vote is counterproductive.

For one thing, gathering requirements from a variety of stakeholders, both technical and nontechnical, can be somewhat of a balancing act. Getting everyone together and attempting to reach a consensus can cause you to miss out on important viewpoints. Some people are hesitant to speak up if they believe that their ideas are going to be evaluated and voted on by the entire group.

More importantly, while everyone's input can be valuable, certain stakeholders' understanding of the business objectives must take precedence over the opinions of the entire group.

You should identify both a facilitator and a final decision maker (the product owner) for the requirements-gathering effort. These two roles should not be held by the same person.

Make it clear that the product owner will make all final decisions. If necessary, take especially difficult decisions out of public view and let your facilitator publicly communicate those decisions once they're made.

Your facilitator should be practiced at encouraging participation while minimizing overzealous input. It's important to identify over-eager participants quickly when conducting group interviews and focus groups. While these people may have important contributions, their enthusiasm may overshadow equally valid viewpoints from the rest of the group. Try using other means, such as one-on-one interviews, surveys, or observation, to gather ideas from less vocal participants.

MYTH 4

When selecting your facilitator, you should also look for someone who is assertive enough to keep discussions moving and on topic.

To keep conversations on track, make prioritization a separate process from gathering.

You can avoid getting pulled into unproductive discussions or repeated discussions on the same topics by using your backlog (your collection of "all requirements") to table off-topic suggestions ("We'll add it to the backlog and discuss it in future grooming sessions"). By putting collected information into your backlog, you can diffuse conflict and affirm value.

Finally, avoid "secret" requirements meetings.

Secret meetings may occur if over-eager stakeholders meet formally or informally before all stakeholders are ready and start making decisions without input from the rest of the team.

Secret meetings may also occur if certain stakeholders want to avoid paperwork, have concerns about losing control, or think they know everything they need to know to successfully execute the project.

Allowing a few stakeholders to make decisions without including the entire team is a recipe for disappointment and loss of trust in the wider community. It is also likely to result in a lot of re-work.

MYTH 5

Myth: Requirements are just technical specifications.

Reality: Requirements should represent a solution to a business challenge.

When gathering requirements, understanding the problem your software is meant to solve is critical to success. Technical requirements may be included (generally as nonfunctional requirements), but they are secondary to the solution.

Start by dividing your contributions into two categories: functional and nonfunctional.

Functional requirements are those things the product or application must do. Consider these to be the business capabilities of your software. What features must it have to fulfill users' needs?

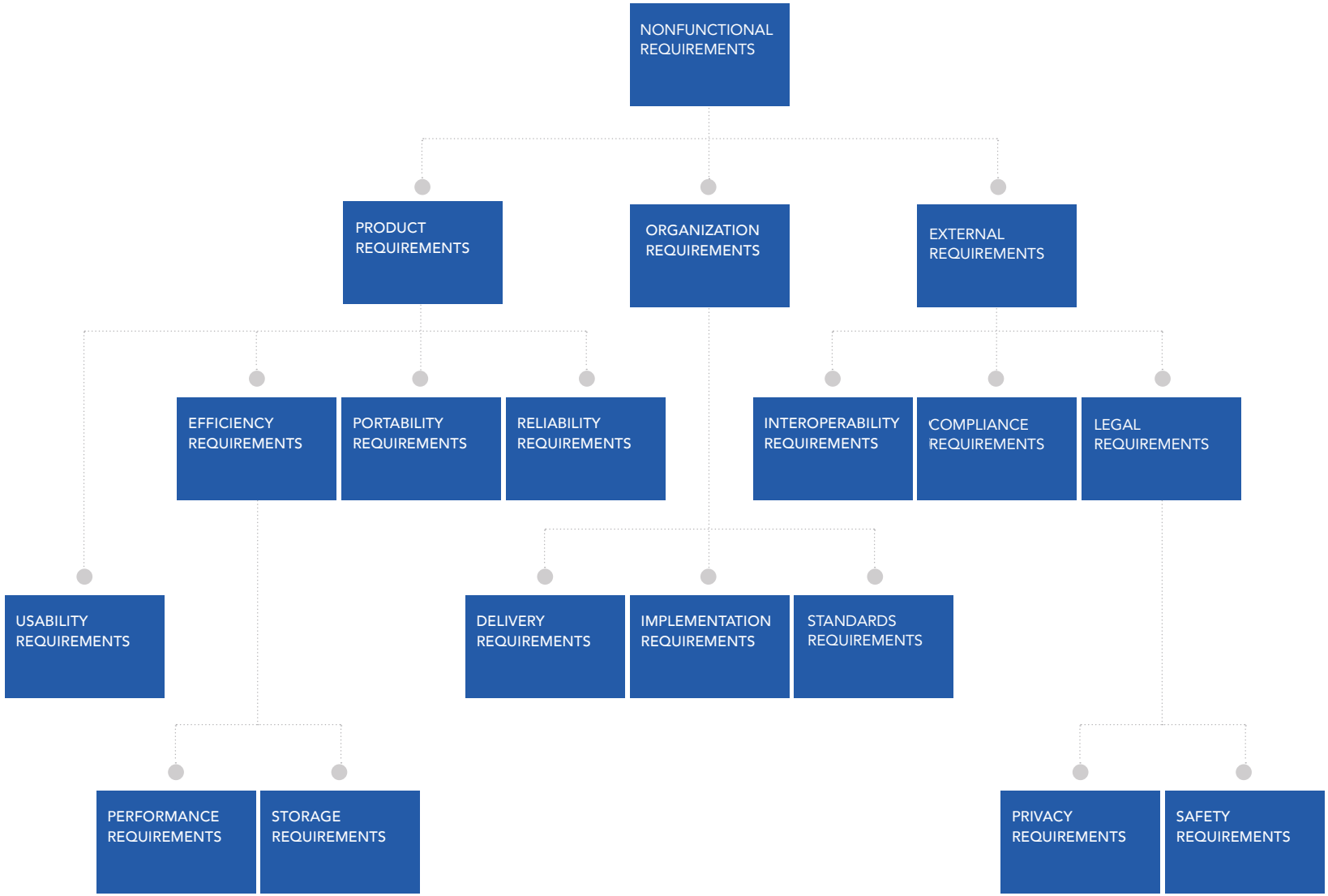
Nonfunctional requirements are qualities the product or application must have to be useful. How fast does it need to run? Does it need to be globally available? How secure must it be? What platform(s) must it run on?

Sometimes technical requirements are called "constraints." It's recommended that you minimize constraints to maximize your development team's options.

Although technical requirements fall into the nonfunctional requirements category, not all nonfunctional requirements are technical in nature.

MYTH 5

This diagram depicts many types of nonfunctional requirements:



(Not all projects are likely to have all of these requirements, and you may find you have nonfunctional requirements that are not represented here. If you are not familiar with all of these categories, engage experts to help make sure you're not overlooking any critical requirements.)

MYTH 6

Myth: Requirements should document the software architecture.

Reality: Software architecture and software requirements are not the same.

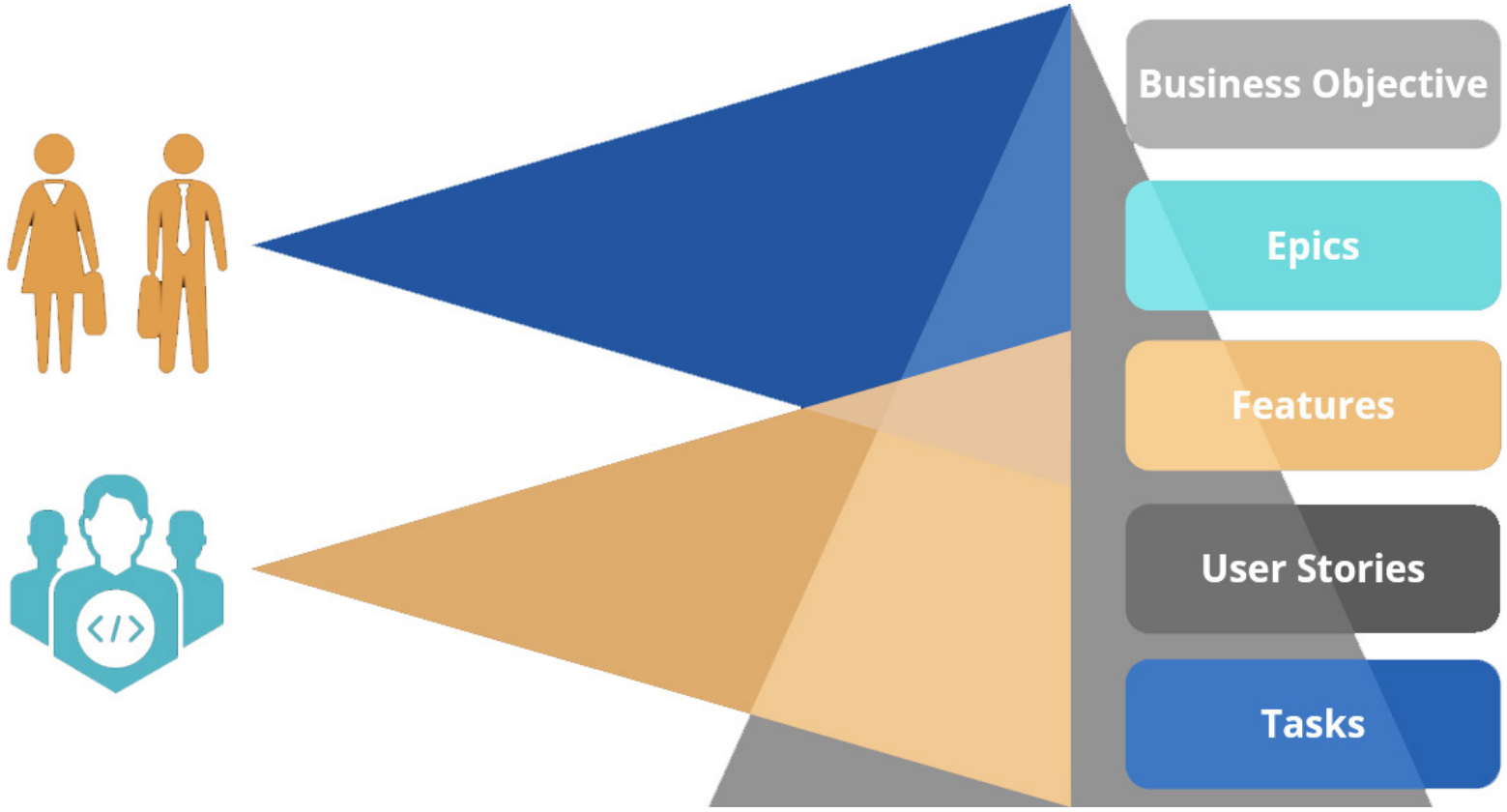
Requirements define "what needs to be done." Software architecture defines "how it is done." Of course, if the business solution is technical, the requirements will be, too.

The purpose of requirements is to describe a business solution. Good requirements follow clear, consistent criteria. Different methodologies outline these criteria using different terminology and key points, but they all define a specific and clear set of touchpoints.

MYTH 6

PYRAMID OF REQUIREMENTS

The Pyramid of Requirements is one tool that can help you visualize the hierarchy of well-organized requirements.



MYTH 6

A simple way to view each of the levels is to think of them as different levels of effort:

BUSINESS OBJECTIVE

At the top of the pyramid is the overall objective of the application, product, or large section of an enterprise capability.

EPIC

Next are large areas of the application or product; these may take months to complete.

FEATURE

Below the Epic requirements are related areas of the application or product; these may take weeks to complete.

USER STORY

A user story is a single work effort that can be applied to the application or product; these may take days (or a single day) to complete.

TASKS

Finally, tasks are individual work elements that, when combined, constitute the completion of a user story; these may take hours to complete.

Different stakeholders should provide input at different levels of the hierarchy, and stakeholders shouldn't worry about "staying at their level." It's up to the experts who are gathering and refining your requirements to organize the collected information in a meaningful structure.

MYTH 6

It's also recommended that you use a defined methodology to evaluate whether each individual requirement truly contributes to your business solution. Two well-known methods are the S.M.A.R.T. and the Project Management Body of Knowledge (PMBOK) methods.

S.M.A.R.T REQUIREMENTS METHOD

The S.M.A.R.T Requirements Method outlines five necessary characteristics of a well-defined requirement:

S Specific

M Measurable

A Attainable (Actionable, Achievable, Appropriate)

R Realistic

T Time Bound (Timely, Traceable)

MYTH 6

PROJECT MANAGEMENT BODY OF KNOWLEDGE METHOD

PMBOK, published by the Project Management Institute, provides these criteria:

COHESIVE
The requirement defines a single aspect of the desired business process or system.

COMPLETE
The individual requirement is not missing necessary or relevant information, and the entire set of requirements covers all relevant requirements.

CONSISTENT
The requirement does not contradict another requirement.

MODIFIABLE
Like requirements are grouped together to allow similar requirements to be modified together in order to maintain consistency.

CORRECT
The requirement meets the actual business or system need. An incorrect requirement can still be implemented resulting in a business process or system that does not meet the business needs.

OBSERVABLE
The requirement defines an aspect of the system that can be noticed or observed by a user. This is often referred to as "implementation agnostic," meaning the requirement does not specify aspects of system architecture, physical design, or implementation decisions (these aspects of a system should be defined separately as constraints).

FEASIBLE
The requirement can be implemented within the constraints of the project, including the agreed-upon system architecture or other physical design or implementation decisions.

UNAMBIGUOUS
The requirement is written objectively such that there is only a single interpretation of its meaning.

VERIFIABLE
It can be demonstrated that the requirement has been met by the final solution via inspection, demonstration, test, or analysis.

It doesn't matter which method you use, as long as you use your selected method consistently.

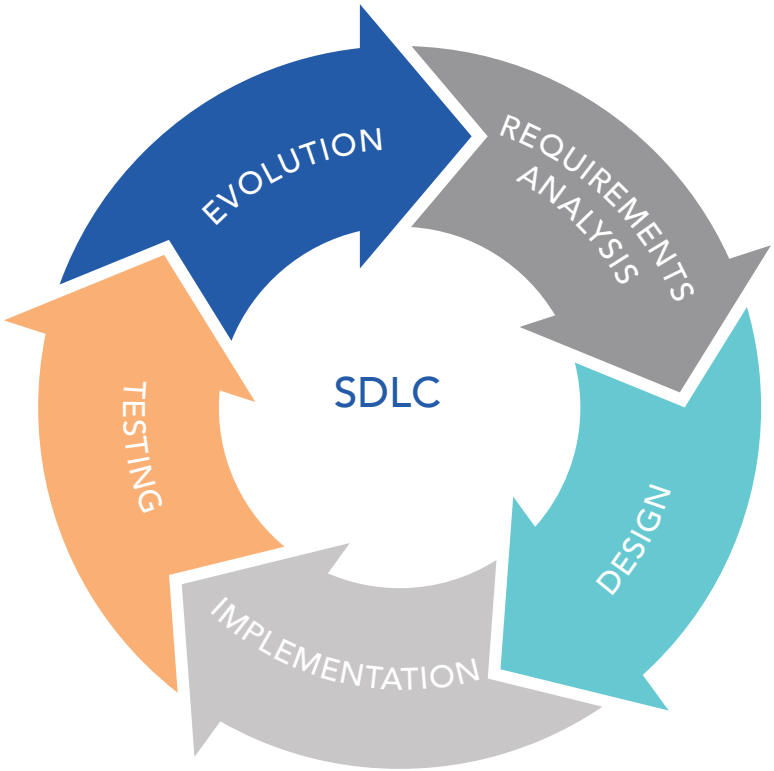
MYTH 7

Myth: Requirements must be complete and perfected before the development team can begin implementation.

Reality: Identifying all requirements up-front is difficult and rarely happens.

In fact, requirements gathering is rarely ever "complete," even after a product is deployed or shipped!

Requirements gathering is part of the the software development life cycle (SDLC) and should be repeated continuously. Plan to continuously collect and review requirements and take advantage of what you learn in each cycle.



MYTH 7

It's important to set expectations with your stakeholders about the iterative nature of requirements gathering and refinement. Not all stakeholders understand (or have experienced) the process of managing and developing complex software solutions.

Staying open to the idea that requirements may change or expand once you receive results and feedback allows you to be flexible enough to adjust and improve your product or application over time.

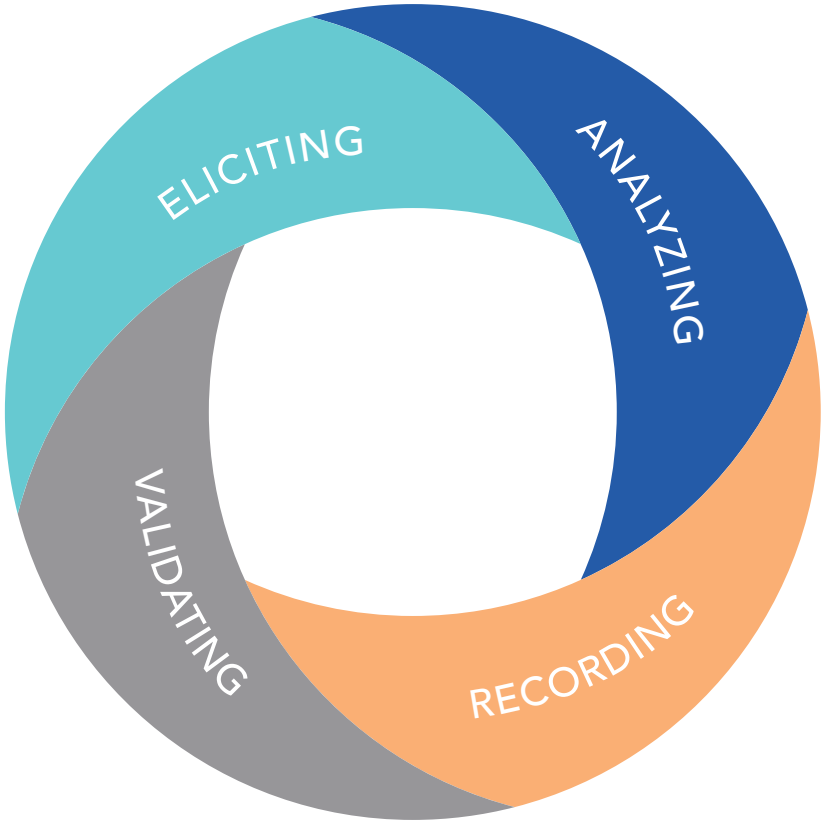
Here are the four key steps in the requirements-gathering process:

ELICITING
Asking questions and listening to the answers

ANALYZING
Organizing and consolidating requirements

RECORDING
Documenting and storing requirements

VALIDATING
Confirming that requirements meet the business need



MYTH 8

Myth: After collecting requirements, they only need to be prioritized once.

Reality: The marketplace is ever-changing. What customers (or end-users) want or need can evolve over time. Adapting to change is critical for business success.

As with the Myth #7, this myth emphasizes the importance of continual evaluation and iteration. Just as gathering requirements is an ongoing process, so is prioritization.

You can greatly improve the overall quality of a product or application by evaluating priorities of remaining work after every cycle.

One way to prioritize your requirements is to use the MoSCoW method of prioritization, which applies the following labels to requirements:

M – Must Have

S – Should Have

C – Could Have

W – Won't Have*

* Variant of W: "Wish to Have"

Scope creep is what happens when the W-labeled requirements sneak into your work cycle.

MYTH 9

Myth: Once requirements are handed off to the development team, implementation is entirely up to them.

Reality: If you gather requirements for the development team, then you are a part of the team.

The "over-the-wall" approach is full of risk, disappointment, and re-work. Development team success (or failure) is yours.

You're a part of the team.

Stay engaged.

Collaborate.

CONCLUSION

Instead of summarizing all 9 myths, we'll conclude with a summary of best practices and tips. Ignoring the common myths and following these guidelines will help keep you on the path to success.

1. Choose requirements-management tools that make gathering and managing requirements easier and more efficient, regardless of the size of your organization or team. Don't overdo it (or underdo it). And please don't use spreadsheets!

2. Requirements come from many sources, both people and documents. Look high and low.

3. Anyone can (and should) contribute to the requirements-gathering process, not just technical people.

4. Don't rely on democracy or consensus to gather and prioritize your requirements; assign decision-making power to a product owner to streamline the process.

5. Most requirements are a collection of business features. Leave the technical details to your technical experts.

6. Requirements define "what needs to be done." Software architecture defines "how it is done."

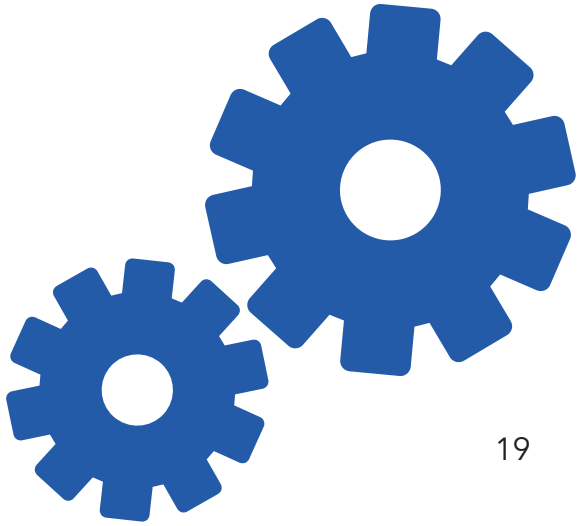
7. Stay engaged with the development team during the entire SDLC. Help improve and refine requirements continually.

8. You can greatly improve the overall quality of a product or application by prioritizing remaining work after every cycle.

9. If you gather requirements for the development team, then you are part of the team. Stay engaged. Collaborate.

And finally, a few more things to keep in mind:

- Experience makes a difference. You will improve over time.
- It helps to work with experts. Take advantage of experience.
- Keep it simple. Requirements are not the objective; they're just a way to get there.
- Seek advice if you have regulatory or compliance needs.



OBJECT
COMPUTING

CONNECT WITH US

+1 (314) 579-0066

@objectcomputing

objectcomputing.com