
Interacting with Stakeholders
Clarifying User Requirements
Simulation

Testing: Theory vs Practice

Testing Theory

So far in this class, we've stuck to the "hard science" part of testing: equivalence classes and boundary values, manual vs automated testing, unit testing syntax.

All very important.

... but there's more to life than theory.

"Marge, I agree with you... in theory. In theory, communism works. In theory."

-Prof. Homer J. Simpson

The "Soft Sciences" Aspects of Testing

As a tester, you're often going to be at odds with development, management, and other stakeholders.

More than a developer, you're going to have to be prepared for conflict resolution with management.

More than a manager, you'll be discussing edge cases in requirements with project analysts.

More than an analyst, you'll be dealing with developers on holes in their code.

People Skills!

"I already told you: I deal with the G** d*** customers so the engineers don't have to. I have people skills; I am good at dealing with people. Can't you understand that? What the h*** is wrong with you people?"

-Tom Smykowski, Office Space

People Skills!

Software development as a whole is much more social than many non-developers think.

The best software in the world will not be successful if nobody knows about it.

Communication (of one form or another) is vital.

People Skills!

Knowing how to generically deal with people one on one is important, but you should also know -

1. To whom should I talk about a problem?
2. When should I talk about it?
3. How do I give presentations, to co-workers and upper management?
4. How should I discuss things with technical vs non-technical people?
5. How do I push back respectfully when somebody disagrees with my assessment?
6. When, how, and why to use CYA documentation

Meetings

If you're bored at a meeting, you probably shouldn't be there. You should either be contributing or so out of your depth that you're constantly writing down acronyms to look up.

That being said.. what is the appropriate way to contribute in meetings? What kinds of meetings is it important for you to attend (and which ones could you probably skip?)

Meetings

Meetings have a bad rap, but they're really just scheduled discussions.

Discussions about the product, which hopefully you care about, or discussions about developing the product, which you also hopefully care about.

People Skills!

No matter what field you end up in - testing or development, management, or even a non-technical field entirely - people skills will serve you well.

People skills are also something that can be learned.

In this lecture...

We are going to discuss:
1. Who and what stakeholders are, and how to interact with them
2. Reporting and communicating with co-workers, management, users, etc.
3. Clarifying user requirements
4. Respectfully disagreeing and working together

Stakeholders

What is a Stakeholder?

A person or group who has an interest in the completion and/or execution of the system under development.

Example

Customers - The people who pay for the system.

Users - The people who will actually use the system (not always the same as the customer)

Project Management - The people who are actively managing the project to completion.

Upper Management - The people who care about the financial ROI (return on investment) of the development of the product.

Examples

Developers - The people who write and maintain the software.

Testers - The people who test the software for defects and quality assurance.

Support staff - Administrators of the system and help desk.

Assessors - The people who oversee the legal and regulatory aspects of the system.

The Classes of Stakeholders

In software development, the split is usually between technical and non-technical stakeholders.

Interaction is going to be very different between them.

Note

A stakeholder is DIRECTLY impacted by the completion and execution of the system.

True, the electrical lineman is kind of impacted if more electricity is used to run your software, but he or she is not a stakeholder.

The janitor may have more pizza boxes to clean up if developers stay late working on the product, but he or she is not a stakeholder.

Note

Although I am specifically talking about software, the term "stakeholder" and the rest of the terminology I'm using is pretty standard across corporate project management.

Identifying Stakeholders

Brainstorm about all of those groups or classes of people who would be impacted by the software.

Narrow it down by removing those who

1. Don't really care
2. Have no power over decision-making
3. Refuse to be a part of it (RED FLAG)

Choose representative(s) from those groups (the exact number will vary based on the size of the project)

Try to Include All Relevant Stakeholders in the Development Process

Software does not get developed in a vacuum.

Even if you are developing it yourself, it's for a reason, even if it's just to learn a new technology.

Speak the Language of the Stakeholders

Software developers speak in terms of technology.

Testers will speak in terms of quality.

Project managers in terms of deadlines.

Upper management in terms of financials.

Care Enough to Learn What They Care About!

Engage Early, Engage Often

"Engagement" is more than just talking.

Give the stakeholder a chance to let you know their opinions. Avoid coloring them with your own.

Get their feedback early on in the process and as often as possible.

"No battle plan ever survives contact with the enemy."
-Helmuth von Moltke the Elder

Be Explicit about Problems, Benefits and Drawbacks

Be as clear as possible when giving status updates.

It is always better to let stakeholders know of problems early.

If you're ever wondering whether or not you should communicate something, do it.

Managing Stakeholder Expectations

UPOD - Under Promise, Over Deliver

Be reasonable.

Keep in mind that unexpected problems will occur.

Identify and Manage Risks

Be upfront with the fact that there are certain factors beyond your - or sometimes anyone's - control.

You can mitigate but never avoid risk.

1. Identify Stakeholders
2. Involve Stakeholders
3. Speak the Language of the Stakeholders
4. Engage early, engage often
5. Be explicit about risks, benefits, & drawbacks
6. Manage Expectations
7. Identify and Manage Risk

What is a Stakeholder?

A person or group who has an interest in the completion and/or execution of the system under development.

Example

Customers - The people who pay for the system.

Users - The people who will actually use the system (not always the same as the customer!)

Project Management - The people who are actively managing the project to completion.

Upper Management - The people who care about the financial ROI (return on investment) of the development of the product.

Examples

Developers - The people who write and maintain the software.

Testers - The people who test the software for defects and quality assurance.

Support staff - Administrators of the system and help desk.

Assessors - The people who oversee the legal and regulatory aspects of the system.

The Classes of Stakeholders

In software development, the split is usually between technical and non-technical stakeholders.

Interaction is going to be very different between them.

Note

A stakeholder is DIRECTLY impacted by the completion and execution of the system.

True, the electrical lineman is kind of impacted if more electricity is used to run your software, but he or she is not a stakeholder.

The janitor may have more pizza boxes to clean up if developers stay late working on the product, but he or she is not a stakeholder.

Note

Although I am specifically talking about software, the term "stakeholder" and the rest of the terminology I'm using is pretty standard across corporate project management.

Identifying Stakeholders

Brainstorm about all of those groups or classes of people who would be impacted by the software.

Narrow it down by removing those who

1. Don't really care
2. Have no power over decision-making
3. Refuse to be a part of it (RED FLAG!)

Choose representative(s) from those groups (the exact number will vary based on the size of the project)

Try to Include All Relevant Stakeholders in the Development Process

Software does not get developed in a vacuum.

Even if you are developing it yourself, it's for a reason, even if it's just to learn a new technology.

Speak the Language of the Stakeholders

Software developers speak in terms of technology.

Testers will speak in terms of quality.

Project managers in terms of deadlines.

Upper management in terms of financials.

Care Enough to Learn What They Care About!

Engage Early, Engage Often

"Engagement" is more than just talking.

Give the stakeholder a chance to let you know their opinions. Avoid coloring them with your own.

Get their feedback early on in the process and as often as possible.

"No battle plan ever survives contact with the enemy."
-Helmuth von Moltke the Elder

Be Explicit about Problems, Benefits and Drawbacks

Be as clear as possible when giving status updates.

It is always better to let stakeholders know of problems early.

If you're ever wondering whether or not you should communicate something, do it.

Managing Stakeholder Expectations

UPOD - Under Promise, Over Deliver

Be reasonable.

Keep in mind that unexpected problems will occur.

Identify and Manage Risks

Be upfront with the fact that there are certain factors beyond your - or sometimes anyone's - control.

You can *mitigate* but never *avoid* risk.

1. Identify Stakeholders
2. Involve Stakeholders
3. Speak the language of the stakeholders
4. Engage early, engage often
5. Be explicit about status, benefits, & drawbacks
6. Manage Expectations
7. Identify and Manage Risk

Reporting to Management, Co-Workers, etc.

The Big Jump - Technical vs Non-Technical

Interacting with developers and other testers tends to be simpler than dealing with non-technical personnel.

This is because you share a common *frame of reference*.

The manner in which you think of the system will depend on your familiarity with the system, your background, and your role.

The Problem in a Nutshell

It's an issue of translation.

Remember that different stakeholders speak different languages. Your job is to translate into their language.

Examples

Upper Management - How does this impact the financials of the product?

Product Management - How does this impact the scope or timeline of the project?

Users - How does this impact the functionality important to me?

Customers - How does this impact what I'm getting or how much I'm paying?

Remember, Care Enough to Speak Their Language

Don't talk about impact on quarterly earning with developers.

Don't talk tail-call optimization with managers.

Be Succinct, but Prepared to Dive Down

Keep in mind how much time and energy the person to whom you are reporting has *for this topic*.

However, be prepared for questions.

A Template for Dealing with Management

Red = This system or subsystem has extremely serious issues which will almost certainly impact the timeline or financials unless they are remediated.
Yellow = There are some warning flags, but they can be dealt with. Some outside help may be necessary.
Green = Everything is A-OK, or there are only minor issues.

Example

- Login/Security System: one minor SQL injection issue found, currently being worked.
- Inventory System: Latency on sharded item database is very high. Will require some research to fix.
- User Interface: No non-trivial issues. UI rebuild is
- System performance: All KPIs in normal range.
- System Performance: Web page currently takes 20 minutes to render, and no known way to remove the 27 MB of JavaScript currently being loaded.
- System Administration: Currently all tasks are being done manually by one person, who has repeatedly threatened to quit and has refused to document any processes.

Example

- Login/Security System: Numerous flaws. Outsiders have achieved superuser access thirteen times last week.
- Database System: System does not scale past 100 KB of storage.
- User Interface: Choice of 4 or 8 ways keybindings popular among Computer Science students, but not for our target audience of first and second graders.
- System Performance: Web page currently takes 20 minutes to render, and no known way to remove the 27 MB of JavaScript currently being loaded.
- System Administration: Currently all tasks are being done manually by one person, who has repeatedly threatened to quit and has refused to document any processes.

When Reporting to Management...

Respect their time.
Respect their (different) skillset.
Respect them.

When Dealing with Users and Customers...

Talk about how this impacts them.
Think about Steve Jobs presenting the iPod... don't mention storage size, processor speed, etc., but talk about "1,000 songs in your packet."
Respect their input.

That doesn't mean you have to take every suggestion they give.

Homer: I want a horn here, here, and here! You can never find a good horn when you are med! And they should all play La Cucaracha! Moving on, how about when you are on a long trip and the kids will not shut up?
Engineer #3: How about a built in video game?
Homer: You are fired!
Engineer #4: A bubble dome that will isolate the kids?
Homer: Bullseye!

but RESPECT them.

In general - respect people.

They have different and valuable insights.

"On two occasions I have been asked, – 'Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?' I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question."

-Charles Babbage, *Passages from the Life of a Philosopher*

The Big Jump - Technical vs Non-Technical

Interacting with developers and other testers tends to be simpler than dealing with non-technical personnel.

This is because you share a common *frame of reference*.

The manner in which you think of the system will depend on your familiarity with the system, your background, and your role.

The Problem in a Nutshell

It's an issue of translation.

Remember that different stakeholders speak different languages. Your job is to translate into their language.

Examples

Upper Management - How does this impact the financials of the product?

Product Management - How does this impact the scope or timeline of the project?

Users - How does this impact the functionality important to me?

Customers - How does this impact what I'm getting or how much I'm paying?

Remember, Care Enough to Speak Their Language

Don't talk about impact on quarterly earning with developers.

Don't talk tail-call optimization with managers.

Be Succinct, but Prepared to Dive Down

Keep in mind how much time and energy the person to whom you are reporting has *for this topic*.

However, be prepared for questions.

A Template for Dealing with Management

Red = This system or subsystem has extremely serious issues which will almost certainly impact the timeline or financials unless they are remediated.

Yellow = There are some warning flags, but they can be dealt with. Some outside help may be necessary.

Green = Everything is A-OK, or there are only minor issues.

Example

- Login/Security System One minor SQL injection issue found, currently being worked.
- Database Storage Latency on sharded Item databases is very high. Will require some research to fix.
- User Interface No non-trivial issues. On schedule.
- System performance All KPIs in nominal range.
- System Administration Some shell scripts left to write and automate, but on schedule.

Easy to see status at a glance, but able to zoom down and discuss individual issues.

Example

- Login / Security System - Numerous flaws. Outsiders have achieved superuser access thirteen times last week.
- Database System - System does not scale past 100 KB of storage.
- User Interface - Choice of vi or Emacs keybindings popular among Computer Science students, but not for our target audience of first and second graders.
- System Performance - Web page currently takes 20 minutes to render, and no known way to remove the 27 MB of JavaScript currently being loaded.
- System Administration - Currently all tasks are being done manually by one person, who has repeatedly threatened to quit and has refused to document any processes.

When Reporting to Management...

Respect their time.
Respect their (different) skillset.
Respect them.

When Dealing with Users and Customers...

Talk about how this impacts them.
Think about Steve Jobs presenting the iPod... don't mention storage size, processor speed, etc., but talk about "1,000 songs in your pocket."
Respect their input.

That doesn't mean you have to take every suggestion they give.

Homer: I want a horn here, here, and here! You can never find a good horn when you are mad! And they should all play La Cucaracha! Moving on, how about when you are on a long trip and the kids will not shut up?

Engineer #3: How about a built in video game?

Homer: You are fired!

Engineer #4: A bubble dome that will isolate the kids?

Homer: Bullseye!

but RESPECT them.

In general - respect people.

They have different and valuable insights.

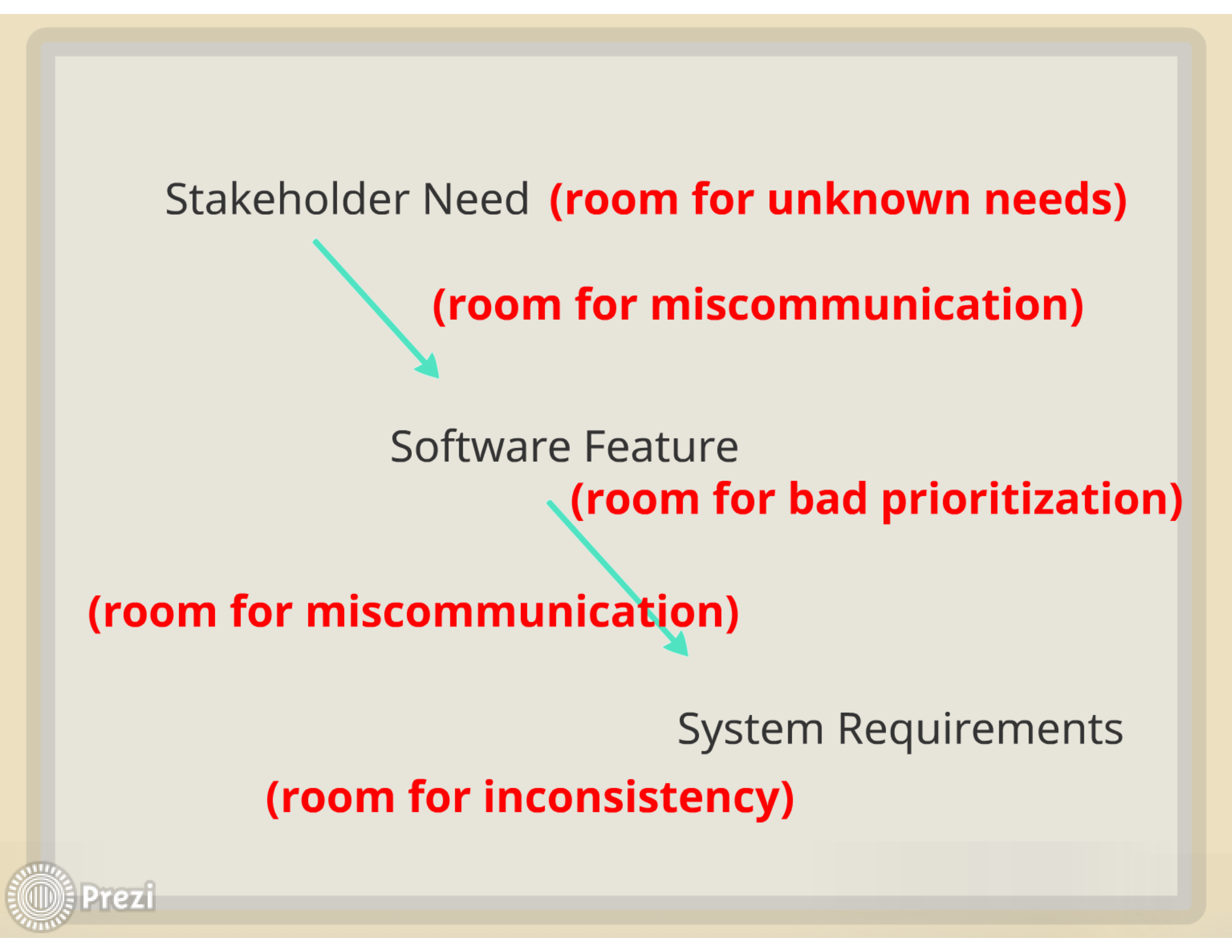
Clarifying User Requirements

You are in charge of testing a function, acount.

SYS-ACOUNT - The subsystem shall accept text and return the number of A's in the text.

Any questions?

1. Is the string ASCII, UTF-8, EBCDIC, other?
2. Does this count capital and lowercase?
3. What about a's with accent marks?
4. Are there performance requirements?
5. How does the text get in (network, Unix pipe, prompt)?
6. What's the return type? (int, BigInt)
7. Does it need to be distributed?
8. What size data is going in?
9. What architectures does it need to run on?
10. Is the data coming from a trusted source?
11. What to do in the case of exceptions?
12. Do I need to worry about thread safety?


Stakeholder Need
(room for unknown needs)
(room for miscommunication)
Software Feature
(room for bad prioritization)
(room for miscommunication)
System Requirements
(room for inconsistency)

More communication is better

You should develop a good relationship with project managers, customers or users (if appropriate), and developers.

It works on my machine!

Remember that requirements are for the user, not the developer.

They guide the developer to create something for the users.

Don't Be Afraid to Push Back

"A tester is someone who stands athwart development, yelling Stop, at a time when no one is inclined to do so, or to have much patience with those who so urge it."

- with apologies to Wm. F. Buckley, Jr

On Being the Bearer of Bad News

Be prepared to give bad news without being a pessimist.

Testers often see a system at its worst. Don't let this color your view of the product.

Be respectful when clarifying requirements. Don't belittle people not being clear... it's difficult work.

"Be respectful and open to others' opinions"
will get you about 90% of the way there.

That being said...

A great way to clarify requirements is to write tests, THEN go back to project management or the requirements analysts and ask if the tests are testing the right thing.

It's easier to talk about a concrete issue and what can be done to modify the tests than just discuss things in the abstract.

Example

Unit tests for acount

Simulation

Role-Playing

In each example, I will be a relevant stakeholder, and you will be a tester.

I am the project manager for a new web application which will revolutionize time-sharing of cats, CatShare++.

I want the status of the three major pieces of functionality:
1. Security
2. Discovery of local cats to share
3. Sharing cats with others

CatShare++ has gone live! You've found several bugs, but need to justify how bad they are (or not!) to upper management.

1. Approximately 1% of users cannot log in due to their password being stored incorrectly.
2. It is possible, but unlikely, that two people could rent a cat at the same time.
3. The entire CatShare++ site could go crash - and take days to recover - if someone searches for "dog" instead of "cat".

You have a meeting with several users of CatShare++. What kind of questions can you ask to get more of an understanding of what to test?

Role-Playing

In each example, I will be a relevant stakeholder, and you will be a tester.

I am the project manager for a new web application which will revolutionize time-sharing of cats, CatShare++.

I want the status of the three major pieces of functionality:

1. Security
2. Discovery of local cats to share
3. Sharing cats with others

CatShare++ has gone live! You've found several bugs, but need to justify how bad they are (or not!) to upper management.

1. Approximately 1% of users cannot log in due to their password being stored incorrectly.
2. It is possible, but unlikely, that two people could rent a cat at the same time.
3. The entire CatShare++ site could go crash - and take days to recover - if someone searches for "dog" instead of "cat".

You have a meeting with several users of CatShare++. What kind of questions can you ask to get more of an understanding of what to test?


Interacting with Stakeholders
Clarifying User Requirements
Simulation