Chapter 14. Understanding the Solution(s)

INTRODUCTION

The title of this chapter implies that there may be multiple options to choose from for satisfying a given need. It is certainly worth considering in some situations, and teams ignore this possibility to their detriment. In other cases, however, there is a clear-cut way of satisfying a given need. The trick is figuring out when you need to create a list of options to pick from, and when you need to figure out all the things you need to do to deliver a clear solution.

Confused? How about a few examples?

You are on the board at a small private school and find that communications are choppy between the administration and faculty and the parents. In addition, you have received multiple complaints about the need to provide the same information during registration year after year.

One of the members of the board took it upon himself to research various options for student information systems and has brought a proposal to the board to consider allocating funds to purchase such a system. The board starts discussing all the features a student information system should have, but something just doesn't sit right with you. Does the school really need a student information system? What problem are we really trying to solve?

You let the conversation go on for a while, until you finally ask the question: “What problem are we trying to solve here, and how will we know whether we’ve solved it?” A hush descends over the room. A couple of the board members nod in agreement. The person who suggested allocating funds to buy a SIS scowls at the table in front of him (but it’s really intended for you). Others look at the ceiling reflectively. Finally, the president of the board slowly starts nodding her head and says, “You’re right. Let’s back up and identify what we’re trying to do, and then think about ways of getting there.”

At this point you can suggest an approach such as impact mapping, which involves starting with a specific goal, then generating a list of actions by working through the people who can impact progress toward the objective, their behaviors, and what can be done to change those behaviors. This is an excellent situation in which to identify multiple options.

In another example, you are putting together a system to support the submission process for a conference. You know you need to replace the existing system, and it’s really important to provide the appropriate functionality to support the session selection process. Impact mapping in this particular case is not very helpful; in fact, it can be a waste of time. You already know you need to build the system, and why. But you do need to understand what things you need to include in the new solution in order to meet the stated objectives. In this situation, a story map and some collaborative modeling can be very helpful in figuring out everything you need to deliver in order to provide a complete, workable solution.

The moral of the story is that while many of these techniques are helpful, they are not always appropriate. In some cases (probably more than you initially may realize), you want to explore options, not get so hung up on the currently popular approach. You have to really understand what organizational need you are trying to satisfy and identify all the possible options to satisfy that need.

On the other hand, you may know what the specific solution needs to look like—at least you may have a clear picture of why you are delivering that solution—and it’s much more important to understand what you need to do to make the solution workable. In these cases, the project itself may have been launched as the result of an impact mapping exercise, but there is no value in trying to parse out the deliverable in terms of an impact map. It’s the wrong tool for the job. These situations call for a way to organize your backlog to reflect the key things needed to provide a complete solution. Ways of organizing things graphically can help put in perspective what things are needed, and when they will be delivered.

Picking the right tool to do this is essential. You don’t want to use an impact map to perform what amounts to a **functional decomposition**. You don’t want options for different things you could do; you want to know what you *have* to do, and which things—still an aspect of the complete solution—might be optional.

This chapter introduces the various techniques you can use to identify possible solutions (impact mapping) or define and describe solutions (story mapping, collaborative modeling). This chapter also discusses three techniques for describing aspects of the solution: models, acceptance criteria, and examples.

You may have noticed that I don't spend a great deal of time on features or user stories in this chapter, or in fact in the entire book. I made this choice because there is already so much on user stories in the literature, and most of it tends to overemphasize their role. User stories were really intended to be placeholders and reminders to have further conversations about the solution. I have chosen to focus on the specifics of those conversations as characterized by models, acceptance criteria, and examples, rather than the reminders to have the conversations.

IMPACT MAPPING

What It Is

Impact mapping combines mind mapping and strategic planning to help a team explore what behaviors they should try to influence in order to reach a particular objective. Teams use impact maps to discuss assumptions, align with organizational objectives, and develop greater focus in their projects by delivering only the things that lead directly to achieving organizational objectives. This also reduces extraneous activities.

Impact mapping structures conversations around four key questions:

• *Why* are we doing this? The answer to this question is the goal that the project is trying to accomplish as measured by an objective.

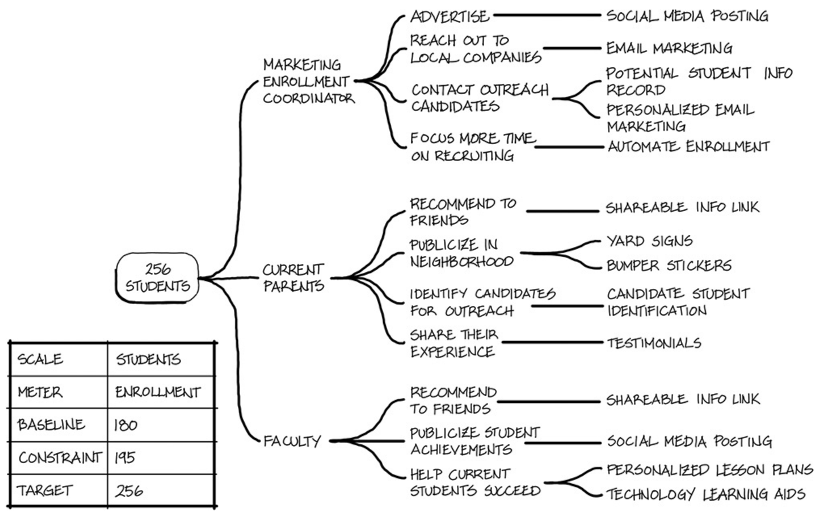
• *Who* can bring the organization closer to this objective, or conversely who may prevent us from reaching the objective? The answer to this question identifies the actors who can have some impact on the outcome.

• *How* should our actors' behavior change? The answers generate the impacts you're trying to create.

• *What* can the organization (specifically the delivery team) do to support the desired impacts? The answer to this question identifies the deliverables, which will typically be software features and organizational activities.

An Example

Figure 14.1 is an example impact map for Deep Thought Academy.


256 STUDENTS
MARKETING ENROLLMENT COORDINATOR
ADVERTISE
SOCIAL MEDIA POSTING
REACH OUT TO LOCAL COMPANIES
EMAIL MARKETING
CONTACT OUTREACH CANDIDATES
POTENTIAL STUDENT INFO RECORD
PERSONALIZED EMAIL MARKETING
FOCUS MORE TIME ON RECRUITING
AUTOMATE ENROLLMENT
CURRENT PARENTS
RECOMMEND TO FRIENDS
SHAREABLE INFO LINK
PUBLICIZE IN NEIGHBORHOOD
YARD SIGNS
BUMPER STICKERS
IDENTIFY CANDIDATES FOR OUTREACH
CANDIDATE STUDENT IDENTIFICATION
SHARE THEIR EXPERIENCE
TESTIMONIALS
FACULTY
RECOMMEND TO FRIENDS
SHAREABLE INFO LINK
PUBLICIZE STUDENT ACHIEVEMENTS
SOCIAL MEDIA POSTING
HELP CURRENT STUDENTS SUCCEED
PERSONALIZED LESSON PLANS
TECHNOLOGY LEARNING AIDS

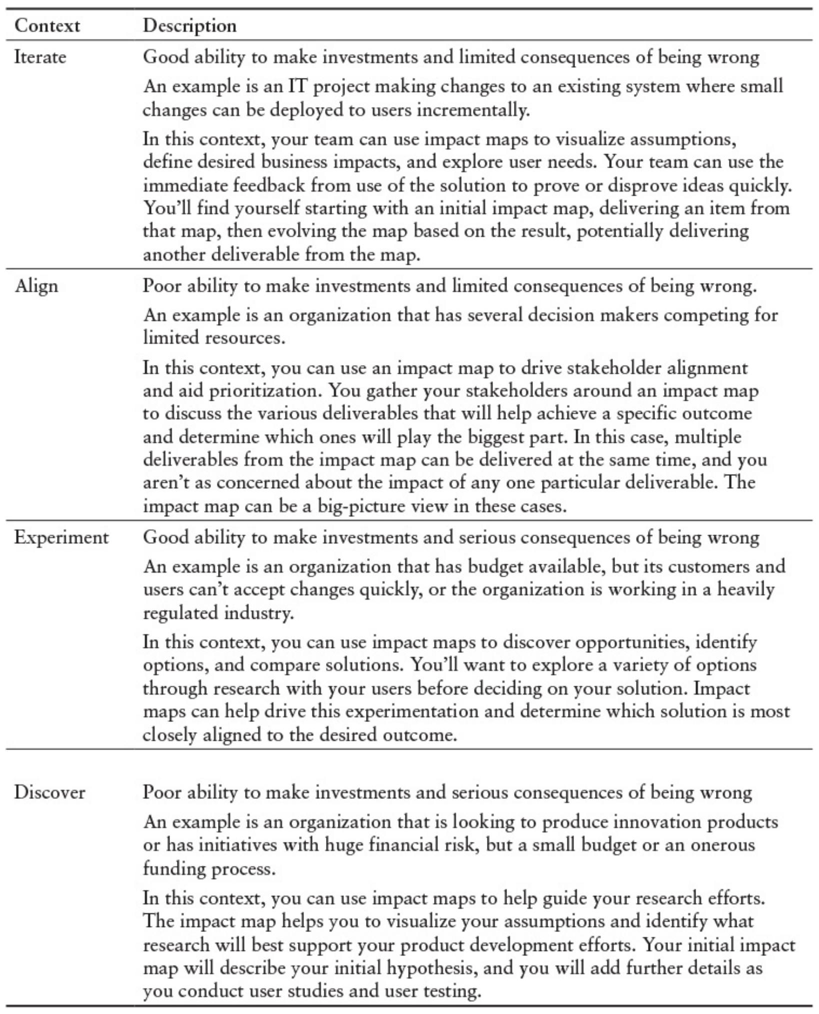
SCALE	STUDENTS
METER	ENROLLMENT
BASELINE	180
CONSTRAINT	195
TARGET	256

Figure 14.1 *Deep Thought Academy impact map*

When to Use It

Impact mapping does not work in every situation. If you used the Context Leadership Model (Chapter 12) to analyze your project, impact mapping would likely be a good technique to use in the colt and bull quadrants, especially if the uncertainty is from the business perspective.

Gojko Adzic, Ingrid Domingues, and Johan Berndtsson wrote an article on *InfoQ* titled “Getting the Most Out of Impact Mapping” (www.infoq.com/articles/most-impact-mapping (http://www.infoq.com/articles/most-impact-mapping)) that describes four different contexts where impact maps can be useful. These contexts are based on two key factors: the consequences of being wrong (making the wrong decision) and the ability to make investments. (See Table 14.1.)



Context	Description
Iterate	Good ability to make investments and limited consequences of being wrong An example is an IT project making changes to an existing system where small changes can be deployed to users incrementally. In this context, your team can use impact maps to visualize assumptions, define desired business impacts, and explore user needs. Your team can use the immediate feedback from use of the solution to prove or disprove ideas quickly. You'll find yourself starting with an initial impact map, delivering an item from that map, then evolving the map based on the result, potentially delivering another deliverable from the map.
Align	Poor ability to make investments and limited consequences of being wrong. An example is an organization that has several decision makers competing for limited resources. In this context, you can use an impact map to drive stakeholder alignment and aid prioritization. You gather your stakeholders around an impact map to discuss the various deliverables that will help achieve a specific outcome and determine which ones will play the biggest part. In this case, multiple deliverables from the impact map can be delivered at the same time, and you aren't as concerned about the impact of any one particular deliverable. The impact map can be a big-picture view in these cases.
Experiment	Good ability to make investments and serious consequences of being wrong An example is an organization that has budget available, but its customers and users can't accept changes quickly, or the organization is working in a heavily regulated industry. In this context, you can use impact maps to discover opportunities, identify options, and compare solutions. You'll want to explore a variety of options through research with your users before deciding on your solution. Impact maps can help drive this experimentation and determine which solution is most closely aligned to the desired outcome.
Discover	Poor ability to make investments and serious consequences of being wrong An example is an organization that is looking to produce innovation products or has initiatives with huge financial risk, but a small budget or an onerous funding process. In this context, you can use impact maps to help guide your research efforts. The impact map helps you to visualize your assumptions and identify what research will best support your product development efforts. Your initial impact map will describe your initial hypothesis, and you will add further details as you conduct user studies and user testing.

Table 14.1 *Contexts for Impact Mapping*

Most of the time when I have used impact mapping for IT projects, it has been in the iterate context. In these projects, we would create an impact map to identify potential deliverables, identify the deliverable we wanted to try first, deliver it, then check the resulting impact on behaviors, and more importantly on the objective, to see if the deliverable had the desired effect.

The other context that occurs most frequently in IT projects is align, where you are trying to get multiple stakeholders to agree on priorities. Gojko described how he has handled that situation in a recent email exchange:

People pretty much know what they want (a transaction accounting system doesn't need a lot of discovery, the domain is pretty clear to everyone), but there are too many things on the list and stakeholders need to align to agree on the priorities that will actually give the organization something big rather than a stream of stories.

In cases such as that, I've used impact maps to paint the big picture and get the stakeholders and tech leaders to agree on the key priorities related to impacts, where the work is then divided [among] several teams. Multiple things can get delivered at the same time, and teams don't rely that much on measurement of impacts to decide what to do next (I still recommend measuring it to ensure that the thing was complete, but it's not the driving factor as in the iterate part of the quadrant, because there is more certainty on internal effects).

Why Use It

Impact mapping offers several advantages when used in the proper context:

• **It reduces waste.** Teams using impact mapping properly and in the proper context will deliver one deliverable at a time and measure the impact of that delivery on the objective. If they meet the objective, they can stop work on that project, satisfying their stakeholders' need with a minimum of new code.

• **It provides focus.** Deliverables are selected based on how they contribute to behaviors that will enable the organization to meet its objective.

• **It increases collaboration.** The discussions that occur while the impact map is created can be very helpful for surfacing assumptions and establishing a sequence for the actions to take in the project.

• **It verifies that the team is building the right thing.** Using impact mapping helps teams ensure that they are focusing on the right outcomes. Teams are also provided with a mechanism to discuss and test their assumptions.

How to Use It

1. Get the team and stakeholders together.

2. Identify the objective (why).

3. Think about people whose behavior can help the organization get closer to the goal, or people whose behavior will move the organization farther away from the goal (actors—who).

4. For all the actors you identified, think about what behaviors you want them to start, or change, to help your organization get closer to the objective, or behaviors that are preventing your organization from getting closer to those objectives (impacts).

5. For each behavior, identify possible things that the organization can deliver to help drive changes in those behaviors (deliverables).

6. Decide which deliverable to deliver first to gauge its impact on the targeted objective.

Caveats and Considerations

Although the second branch is described as exploring a "how" question, the "how" is focused on how you would like your actors' behavior to change, not how to deliver a particular piece of functionality. I find it better to refer to this as the impacts rather than "how," to reinforce the idea that it is focused on behavior.

The third branch, deliverables, is the first mention of IT. The goal is to focus on behavior change first and then explore ways we can support that behavior change.

Just because you identified a lot of different options does not mean you should enact all of them. You want to reach the stated objective, but do it with the least amount of work possible, so once your team has diverged on a list of options, they should then converge on what they believe to be the best first option.

If your project has multiple objectives, do an impact map for each objective.

Additional Resources

Adzic, Gojko. *Impact Mapping: Making a Big Impact with Software Products and Projects*. Provoking Thoughts, 2012.

Campbell-Pretty, Em. "Adventures in Scaling Agile." www.prettyagile.com/2014/02/how-i-fell-in-love-with-impact-mapping.html (http://www.prettyagile.com/2014/02/how-i-fell-in-love-with-impact-mapping.html).

"Impact Mapping." http://impactmapping.org/ (http://impactmapping.org/).

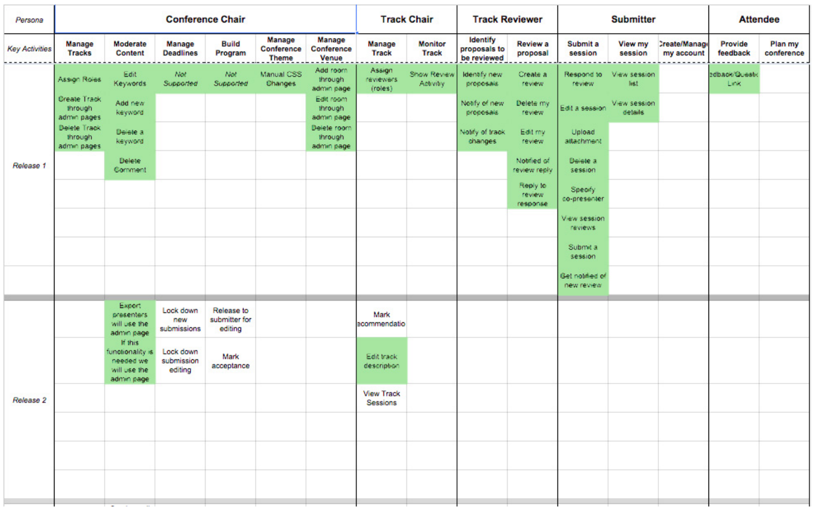
STORY MAPPING

What It Is

A story map is a visual representation of a backlog that provides additional context regarding how backlog items are related to each other and when the team is planning to deliver them. This context is typically presented in terms of the personas that will use specific features, and the particular user stories that are associated with the features.

An Example

Figure 14.2 shows an example story map for the conference submission system.



Persona	Conference Chair						Track Chair		Track Reviewer		Submitter			Attendee	
Key Activities	Manage Tracks	Moderate Content	Manage Deadlines	Build Program	Manage Conference Theme	Manage Conference Venue	Manage Track	Monitor Track	Identify proposals to be reviewed	Review a proposal	Submit a session	View my session	Create/Manage my account	Provide feedback	Plan my conference
	Assign Roles	Edit Keywords	*Not Supported*	*Not Supported*	Manual CSS Changes	Add room through admin page	Assign reviewers (roles)	Show Review Activity	Identify new proposals	Create a review	Respond to review	View session list		edback/Questic Link	
	Create Track through admin pages	Add new keyword				Edit room through admin page			Notify of new proposals	Delete my review	Edit a session	View session details			
	Delete Track through admin pages	Delete a keyword				Delete room through admin page			Notify of track changes	Edit my review	Upload attachment				
Release 1		Delete Comment								Notified of review reply	Delete a session				
										Reply to review response	Specify co-presenter				
											View session reviews				
											Submit a session				
											Get notified of new review				
		Export presenters will use the admin page	Lock down new submissions	Release to submitter for editing			Mark ecommendatio								
		If this functionality is needed we will use the admin page	Lock down submission editing	Mark acceptance			Edit track description								
Release 2							View Track Sessions								

Figure 14.2 *Conference submission system story map*

When to Use It

Story maps as originally described by Jeff Patton are a very helpful elicitation technique when trying to understand a solution that has a great deal of user interaction. Creating the story map guides the team as they talk through the business process, identify the key activities (represented as features), and lay them out in a logical sequence.

There are many cases where the solution does not inherently support a single process or a logical step-by-step order is not so clear. In those cases, story maps can still be useful for visualizing the relationships between features and user stories and representing when specific user stories will be delivered for a given feature.

Why Use It

The unique visual structure of story maps helps the team determine if they have a complete, viable solution.

Story maps also help teams identify the appropriate contents of a given release. The release goal should be to deliver the minimum acceptable functionality while still providing a viable, valuable output with enough useful functionality for stakeholders to provide feedback about. Story maps help the team identify that minimum feature set.

Finally, story maps provide a useful graphical representation that shows which stories are planned for a given release and relates that in context to the features they support. It also encourages discussions about what aspects of a feature really need to be delivered. In many cases, a feature represents a broad area of functionality, and the user stories identified for that feature represent things that have to be done, things that are nice to have, and other things that could easily be considered bells and whistles. The story map generates conversations where the team determines the things that need to be delivered and skips the items that are nice to have. This delivers on the objectives of the solution without wasting time and effort on functionality that doesn't add to its effectiveness.

How to Use It

The Story Map as an Elicitation Tool

Gather together key stakeholders and team members. You want to strike a careful balance between having different perspectives in the group and having an unwieldy number of people. A general guideline is to follow the **two-pizza rule**: a good-size group is one you can feed with two large pizzas.

You're also going to want a large surface, either a wall or a table, where you can lay out the sticky notes or index cards you use for the map. Some groups have even resorted to using the floor.

Jeff Patton suggests the following steps for using a story map in an elicitation setting. This technique is especially helpful when eliciting information about a process.

1. Write out your story one step at a time. As a group, talk through the various things that happen in the process and write each thing down on a sticky note or index card. Each of these items is a user task, which in this context is a short verb phrase that describes something people do to reach a goal.

2. Organize your story. If you weren't already doing so when you identified the tasks, arrange them from left to right in the order they occur. This creates a narrative flow and implies the phrase "and then I . . ." between the tasks. If there are certain tasks that happen at the same time or in lieu of each other, place them vertically in a column.

3. Explore alternative stories. Once you have the tasks in a rough narrative flow, play "What About." Discuss alternative things that could happen at various points during the story. Write these thoughts on additional sticky notes or index cards and place them in the appropriate column.

4. Distill your map to make a *backbone*. Review all the tasks and where they combine into common groups, and use an easily distinguished note (for instance, a different color or shape) as a group title, or activity. The activity should also be written as a verb phrase that distills all the tasks underneath it. These activities should also form a narrative flow and provide the outline of a high-level story.

5. Slice out tasks that help you reach a specific outcome. Identify a specific outcome that you want to accomplish, and then identify the specific tasks that are absolutely essential to arriving at that outcome. Leave all the activities at the top of the story map, but move the tasks that don't contribute to the particular outcome below the line for that outcome. This step allows you to focus on only the tasks and activities that are essential to accomplishing your desired outcome. Those outcomes can be thought of as a "happy path" through a process or a minimum viable product.

When you are ready to start delivering the solution represented by the map, you can think of each task as a user story. You especially benefit from the fact that the tasks are already written from a user perspective.

The Story Map as a Backlog Visualization Tool

Even if you are working on a solution that does not have a large amount of user interaction or does not clearly support a business process, the story map format can still help you understand the context of your backlog.

1. Frame the problem. Establish a shared understanding of what need the solution is intended to satisfy. The techniques described in Chapter 13 can be especially helpful here.

2. Map the big picture. Lay out the story map using features as the high-level items across the top of the map. If you have different types of users who can use specific types of features, it may be helpful to organize the features by user. If there are any obvious user stories that can be identified for those features, place them under the appropriate feature at this point.

3. Explore. Select the feature(s) you believe you will be delivering first and do a deep dive on them via conversation with the interested stakeholders. Sketch models to aid the conversation (you may find those sketches helpful later on when you start delivering those particular stories). As you have those discussions you may find that you will refine your map.

4. Slice out a release strategy. Look at the user stories identified for the features and determine the minimum user stories needed to deliver the desired goal. The idea is to identify the minimum output to deliver the maximum outcome. Organize these user stories into a set of releases by moving them vertically.

5. Identify the items to start with. Once you have identified a set of releases, you may find it helpful to identify the user stories you want to start with, as experiments where you are either trying to validate key assumptions or reduce risk. These stories become the topics of your first iteration.

Caveats and Considerations

It's very likely that you will identify and place more items on your story map than you actually deliver. This is appropriate and expected, as one of the reasons for a story map is to identify user stories that are needed and those that are extraneous in the broader context of what you are trying to accomplish.

If you have used the story map to elicit information about a process, it may be difficult to identify names for the activities, as the grouping may not be as natural as the individual tasks. When trying to name these activities, think about what your users/stakeholders would call them.

Do not try to use story maps in isolation from other techniques. They are ultimately an aid to collaboration and conversation, whether you are using them for elicitation purposes or to visualize the backlog.

Story maps can also be used to familiarize people with the process the solution is supporting. Members of the team can walk new folks through the process using the story map as a visual aid.

Additional Resources

Patton, Jeff. *User Story Mapping: Discover the Whole Story, Build the Right Product*. O’Reilly Media, 2014.

———. “User Story Mapping.” www.agileproductdesign.com/presentations/user_story_mapping/ (http://www.agileproductdesign.com/presentations/user_story_mapping/).

COLLABORATIVE MODELING

What It Is

Collaborative modeling refers to the use of well-known requirements analysis and modeling techniques in a collaborative fashion to build and maintain a shared understanding of the problem space and the potential solution. The main premise is that requirements models, which have long been viewed as documentation techniques, can also be put to great use as elicitation and analysis techniques in a collaborative setting with the delivery team and stakeholders jointly discussing the problem and solution.

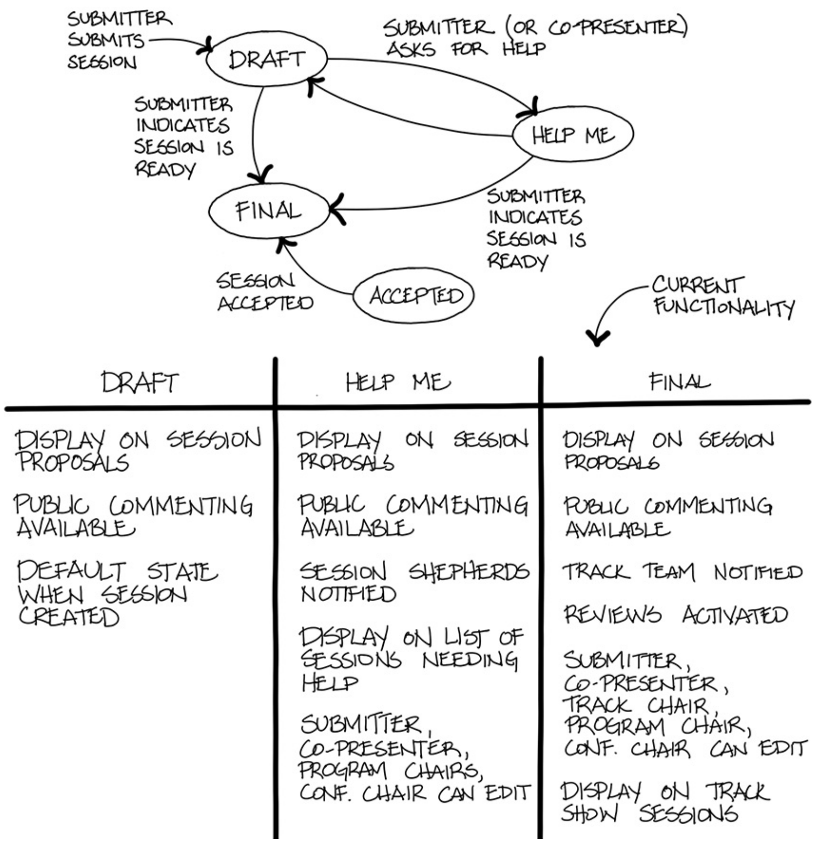
Modeling techniques that I find particularly helpful are listed in Table 14.2. Note that for consistency and familiarity I list each of these techniques based on the result they create, but I cannot stress enough that the artifacts are not as important as the discussions held to create them. The artifacts can be helpful to document the discussions and any decisions made, but the discussions themselves are powerful ways to build shared understanding. The resulting artifacts go from being the sole means of communication to aids for the overall communication.

Technique	Description
Data dictionary	Agree on entities and their attributes as well as the definitions and specific characteristics of both.
Context diagram	Understand the people, systems, or organizations impacted by a solution and the interfaces between those parties and the solution.
Logical data model	Relative to a possible solution, understand the data that stakeholders want to know and remember and how that data is organized.
State transition diagram	Understand the specific states a particular entity can be in and what causes the state to change.
Glossary	Agree on key terms and their definitions.
Organization chart	Understand the reporting relationships between people impacted by a solution.
Value stream map	Identify opportunities for improvement in the operations of an organization.
Functional decomposition	Understand complex processes, systems, functional areas, or deliverables by breaking them down into their simpler constituent parts.
Process flow	Understand the specifics of a particular process for the purpose of identifying changes to implement a solution.
Wireframe	Agree upon the nature of a user interface, including what information should be included.
Report mockup	Understand the information needs of stakeholders in order to help them answer questions or make decisions.

Table 14.2 *Collaborative Modeling Techniques*

An Example

Figure 14.3 is a state transition diagram I put together for the submission system to represent a change for Agile2015.

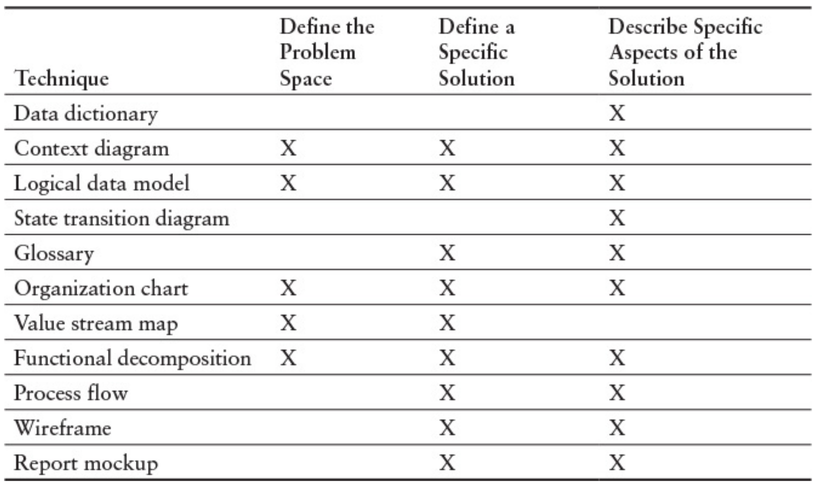

SUBMITTER SUBMITS SESSION
DRAFT
SUBMITTER (OR CO-PRESENTER) ASKS FOR HELP
HELP ME
SUBMITTER INDICATES SESSION IS READY
FINAL
SUBMITTER INDICATES SESSION IS READY
SESSION ACCEPTED
ACCEPTED
CURRENT FUNCTIONALITY

DRAFT	HELP ME	FINAL
DISPLAY ON SESSION PROPOSALS	DISPLAY ON SESSION PROPOSALS	DISPLAY ON SESSION PROPOSALS
PUBLIC COMMENTING AVAILABLE	PUBLIC COMMENTING AVAILABLE	PUBLIC COMMENTING AVAILABLE
DEFAULT STATE WHEN SESSION CREATED	SESSION SHEPHERDS NOTIFIED	TRACK TEAM NOTIFIED
	DISPLAY ON LIST OF SESSIONS NEEDING HELP	REVIEWS ACTIVATED
	SUBMITTER, CO-PRESENTER, PROGRAM CHAIRS, CONF. CHAIR CAN EDIT	SUBMITTER, CO-PRESENTER, TRACK CHAIR, PROGRAM CHAIR, CONF. CHAIR CAN EDIT
		DISPLAY ON TRACK SHOW SESSIONS

Figure 14.3 *Submission system state diagram*

When to Use It

Different collaborative modeling techniques are useful in different aspects of an IT project. The three aspects are listed here, followed by Table 14.3, which indicates which features apply when.



Technique	Define the Problem Space	Define a Specific Solution	Describe Specific Aspects of the Solution
Data dictionary			X
Context diagram	X	X	X
Logical data model	X	X	X
State transition diagram			X
Glossary		X	X
Organization chart	X	X	X
Value stream map	X	X	
Functional decomposition	X	X	X
Process flow		X	X
Wireframe		X	X
Report mockup		X	X

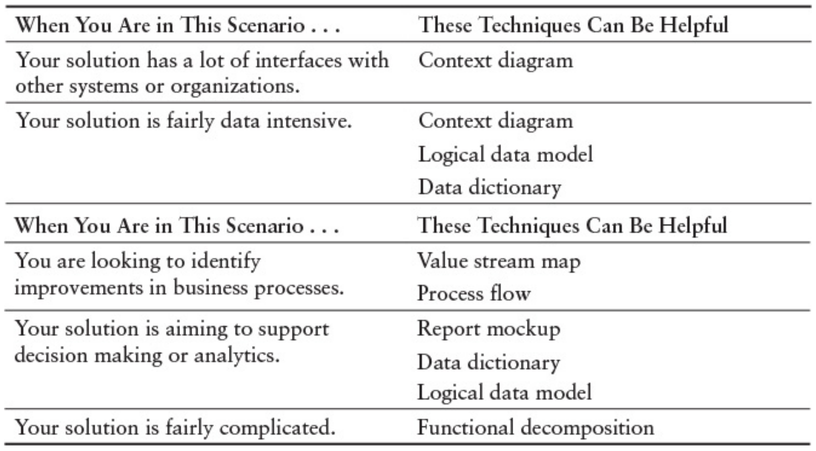
Table 14.3 *Applicable Collaborative Modeling Techniques*

• **Define the problem space.** Your team can use collaborative modeling when it starts work on a new project and needs to understand the context in which a problem occurs (I often refer to this as the "problem space") and how potential solutions might impact the problem space.

• **Define a specific solution.** Your team can use collaborative modeling to define a specific solution and provide a foundation for the team to identify implementation options. When used for this purpose, the models help your team identify features and user stories based on a full understanding of the solution.

• **Describe specific aspects of the solution.** Your team can use collaborative modeling to further describe specific backlog items. The models you use for this aspect may be ones you initially created to define the solution, or your team may find it helpful to create more detailed models to get a better understanding about a particular aspect of the solution.

There are a variety of different modeling techniques that are all very useful in specific situations (see Table 14.4), but none of the techniques is applicable in all situations.



When You Are in This Scenario . . .	These Techniques Can Be Helpful
Your solution has a lot of interfaces with other systems or organizations.	Context diagram
Your solution is fairly data intensive.	Context diagram Logical data model Data dictionary
You are looking to identify improvements in business processes.	Value stream map Process flow
Your solution is aiming to support decision making or analytics.	Report mockup Data dictionary Logical data model
Your solution is fairly complicated.	Functional decomposition

Table 14.4 *Scenarios Where Collaborative Modeling Techniques Are Helpful*

Why Use It

Collaborative modeling provides a way for teams to build a shared understanding of the problem and solution options first, without having to go down the path of breaking the solution into implementation chunks (i.e., user stories). This approach addresses a couple of issues that occur when backlog creation relies on brainstorming alone.

Backlogs Don't Identify a Complete Solution

By discussing the solution via models first, the team can identify all the changes that need to occur to implement a viable solution because they have a picture to fall back on. Brainstorming alone does not provide that big picture as a way of validating that the team has identified what is needed.

Backlogs End Up Becoming a Wish List

When a team collaboratively models the solution and uses the model as a way of identifying the changes that need to happen, they can also use the model to help identify changes that aren't necessary. Building a backlog via brainstorming can often generate backlog items that are not absolutely essential to the desired solution. By having a model to reference that shows the specific changes that are necessary, the team can identify extraneous items that are not essential in order to solve the problem.

How to Use It

The general steps for collaborative modeling are quite simple:

1. Gather the right people together. The definition of "right" is based on the subject of the discussion and the intended outcomes.

2. Make sure the place where you gather is near a whiteboard and/or flip chart paper (preferably both) and that there are plenty of sticky notes and the right type of markers available.

3. Identify the reason for the discussion. Are you there to discuss the overall context of the solution, analyze a specific process, or agree on a particular user interface or report? Identify discussion acceptance criteria: in other words, how you will know the discussion was successful.

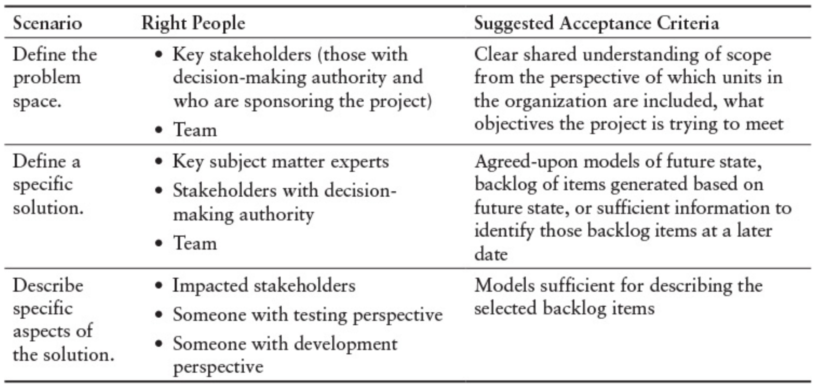
4. Make sure everyone has the same understanding of the current state. This is not as simple as asking if everyone is "on the same page." It's best to sketch out the current state quickly, or start with an existing representation of the current state, and explicitly ask if everyone is in agreement. If there is any disagreement, adjust the description of the current state until people indicate that it represents the true current situation.

5. Have the person with the best understanding of the desired change start describing the change by sketching on the whiteboard and talking through it at the same time. You may also find that it is helpful to have someone guide the discussion by asking questions of the stakeholder and sketching his or her interpretation of the answer on the whiteboard. The key is to talk and sketch things at the same time, to reinforce the conversation and lead to greater agreement.

6. End the discussion when everyone agrees that you have met the discussion acceptance criteria identified in step 3.

7. If anyone in the discussion thinks it would be helpful to keep the sketches that were made during the discussion, take pictures of them and save the pictures in a commonly agreed-upon repository for project documentation.

The specifics of these steps can vary depending on the reason for the discussion. Some of the main variations are described in Table 14.5.



Scenario	Right People	Suggested Acceptance Criteria
Define the problem space.	• Key stakeholders (those with decision-making authority and who are sponsoring the project) • Team	Clear shared understanding of scope from the perspective of which units in the organization are included, what objectives the project is trying to meet
Define a specific solution.	• Key subject matter experts • Stakeholders with decision-making authority • Team	Agreed-upon models of future state, backlog of items generated based on future state, or sufficient information to identify those backlog items at a later date
Describe specific aspects of the solution.	• Impacted stakeholders • Someone with testing perspective • Someone with development perspective	Models sufficient for describing the selected backlog items

Table 14.5 *Collaborative Modeling Variations*

Caveats and Considerations

The same models used to define a specific solution can also be used to further describe specific aspects of the solution. The models used to define a specific solution may be general in nature, and the discussions when describing specific aspects of the solution will describe parts of the models in further detail.

Collaborative modeling works best when the participants in the conversation are in the same room. If some members of your group work offsite, you can still do collaborative modeling with the aid of technology. Bring as many people together as you can in the same room, and then use a laptop or tablet camera to share the whiteboard with virtual team members. If all members of the team are distributed, use screen-sharing and handwriting apps to simulate an electronic whiteboard. The key in all cases is to supplement discussions with visuals.

Additional Resources

Agile Modeling. "Agile Models Distilled: Potential Artifacts for Agile Modeling." http://agilemodeling.com/artifacts/ (http://agilemodeling.com/artifacts/).

Agile Modeling. "Inclusive Modeling: User Centered Approaches for Agile Software Development."
http://agilemodeling.com/essays/inclusiveModels.htm
(http://agilemodeling.com/essays/inclusiveModels.htm).

ACCEPTANCE CRITERIA

What It Is

Acceptance criteria are the conditions that a solution must satisfy to be accepted by a user, a customer, or, in the case of system-level functionality, the consuming system. They are also a set of statements, each with a clear pass/fail result, that specify both functional and nonfunctional requirements and are applicable at a variety of levels (feature and user stories).

Liz Keogh notes that acceptance criteria are useful for several things:

• Further defining the boundaries of a story

• Serving as the functional requirements for the story

• Serving as a set of rules that cover aspects of a system's behavior, and from which examples can be derived

Acceptance criteria point to things that need to be in place in order for the product owner or stakeholders to accept that the user story meets their needs. As with all good functional requirements, the acceptance criteria should focus on business intent rather than detailing a solution, leaving those specifics for refinement during or right before delivery. Since we're not looking to declare a specific solution, acceptance criteria tend to be implementation agnostic—describing what the team wants to accomplish, not how they're looking to accomplish it. Those types of details can be left to models and examples, or to be figured out during actual delivery.

An Example

This example contains a set of acceptance criteria from the conference submission system, specifically the ability for a reviewer to provide a review of a session proposal.

Click here to view code image

```
As Reed the Reviewer
  I want to review a session
  So that I can provide feedback to a submitter.
```

Mind Map of Acceptance Criteria

Figure 14.4 is an example mind map of acceptance criteria for the conference submission system.

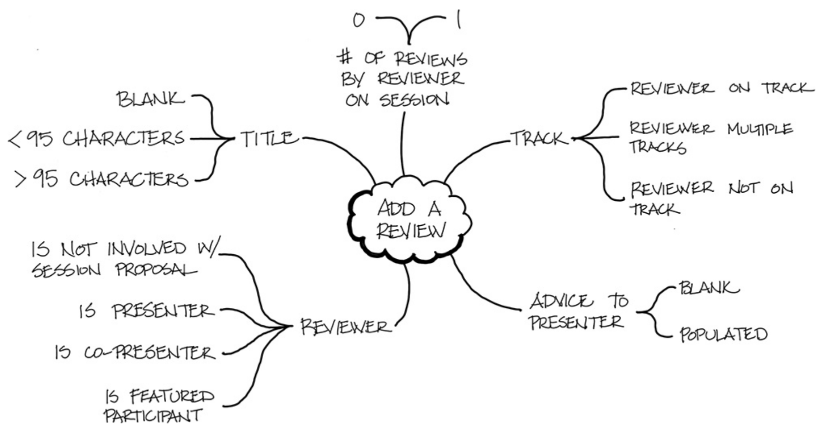


Figure 14.4 *Acceptance criteria mind map*

Potential Acceptance Criteria

• Reviewers must provide a title and description for the review.

• Reviewers may indicate whether they think the session should be included in the program.

• Reviewers may provide details of any conflicts of interest they have in reviewing the session.

• Reviewers may provide comments for the review committee.

• Submitters of the reviewed session can see only the title and description of the review.

• Submitters may see only reviews of sessions that they have submitted.

• Reviewers may review only sessions submitted to tracks on which they are reviewers.

• Reviewers may not review any session on which they are presenters or co-presenters.

• Reviewers may provide only one review for a session.

• The title of the review must contain 95 characters or less.

When to Use It

Acceptance criteria are used most frequently to provide further detail about features and user stories. Many teams find it helpful to identify some acceptance criteria fairly early, to assess the scope included in the story before they size it. Teams may then add acceptance criteria as they discuss the project further, understand the story better, and get closer to delivering the solution.

Why Use It

Defining acceptance criteria is a good way to start adding more detail to the skeleton of a story, and that's where I see most teams doing it. Acceptance criteria are helpful in describing the scope of a story even if (or perhaps especially when) you are using a model to further describe the story. The model may serve as a reference for multiple stories, so the acceptance criteria lend some perspective to specifically which aspects are being delivered with a given story.

How to Use It

1. Meet with the stakeholder(s) who is interested in the particular user story, someone with a development perspective, and someone with a testing perspective near a whiteboard or flip chart paper. Many teams refer to this group as the **three amigos**.

2. Discuss the user story to determine what the stakeholder hopes to accomplish.

3. Start discussing various things that the team should verify in order to make sure that they deliver the user story correctly. Use a mind map to aid that conversation so the group can see what has been discussed and use that to inspire additional thoughts.

4. Note the acceptance criteria from the mind map and discuss whether all of the acceptance criteria should apply to that user story, or whether the number of acceptance criteria indicates that the user story should be split.

5. Note the related acceptance criteria in your backlog repository of choice.

Caveats and Considerations

Some teams like to preface everything with "Verify that . . ." to reinforce that in order to call a user story done they have to verify specific acceptance criteria, but that seems a bit repetitive and fairly clearly implied. Other teams like to state acceptance criteria in the first person, such as "I need to provide a title and description in order to submit a review." A derivation of that is describing acceptance criteria in terms of the subject of the user story, for example, "Reed needs to provide a title and description in order to submit a review." Either way probably works fine and becomes a matter of taste for the delivery team, but this is certainly something helpful for the team to discuss and agree on.

RuleSpeak, created by Ron Ross, is a “set of guidelines for expressing business rules in concise, business-friendly fashion” (www.rulespeak.com/en/ (http://www.rulespeak.com/en/)). Since many acceptance criteria are a set of rules that cover aspects of a system’s behavior, it may be helpful to apply the precise ideas behind RuleSpeak when describing acceptance criteria, especially if you are in an environment with people who get very specific about how things are written. My only caveat is that you don’t want to get so hung up on getting the wording right that you spend more time on the form and lose site of the function. Again, the key is capturing information in a way that the members of the delivery team understand. Some precision is good. Excruciatingly exact precision at the expense of timeliness is not always so helpful. If the team chooses to use RuleSpeak in their acceptance criteria, the presence of a rule in the criteria indicates that enforcement of that rule is included in the delivery of that particular user story.

As with all the techniques described here, use your common sense to pick the pattern that works best. Don’t try to force a particular piece of information into a structure that doesn’t fit.

Some acceptance criteria items can also be conveyed by a model (such as the information included, and which pieces of information are required), but you may want to be explicit about how much you actually deliver with a given user story.

Also, since acceptance criteria are good for indicating the boundaries of user stories, you will often find that a long list of acceptance criteria is a good indication that the user story needs to be split into smaller stories.

Additional Resources

Keogh, Liz. “Acceptance Criteria vs. Scenarios.” http://lizkeogh.com/2011/06/20/acceptance-criteria-vs-scenarios/ (http://lizkeogh.com/2011/06/20/acceptance-criteria-vs-scenarios/).

Laing, Samantha, and Karen Greaves. *Growing Agile: A Coach’s Guide to Agile Requirements*. https://leanpub.com/agilerequirements.

Mamoli, Sandy. “On Acceptance Criteria for User Stories.” http://nomad8.com/acceptance_criteria/ (http://nomad8.com/acceptance_criteria/).

Ross, Ron. “RuleSpeak.” www.rulespeak.com/en/ (http://www.rulespeak.com/en/).

EXAMPLES

What It Is

Examples are concrete descriptions of the expected behavior of an aspect of a solution using real-life data. Examples are useful for describing a solution and providing guidance on ways to validate it. The use of examples to describe a solution is also known as specification by example, behavior-driven development (BDD), or acceptance test driven development.

There are two common forms used to convey examples. Both forms arose around the needs of automated testing frameworks.

The first format was created to support Fit, the **Framework for Integrated Test**. The intent was to enable stakeholders to provide examples in tools familiar to them (such as a word processor), which developers could then hook up to "fixtures" to produce automated tests. The examples are formatted into tables (which resemble decision tables) in HTML files. Table 14.6 shows how this format works.

Input1 Heading	Input2 Heading	Input3 Heading	Output1 Heading	Output2 Heading
Scenario 1 Input1 value	Scenario 1 Input2 value	Scenario 1 Input3 value	Scenario 1 Output1 value	Scenario 1 Output2 value
Scenario 2 . . .	Scenario 2 . . .	Scenario 2 . . .	Scenario 2 . . .	Scenario 2 . . .
Scenario 3 . . .	Scenario 3 . . .	Scenario 3 . . .	Scenario 3 . . .	Scenario 3 . . .
Scenario 4 . . .	Scenario 4 . . .	Scenario 4 . . .	Scenario 4 . . .	Scenario 4 . . .

Table 14.6 *Examples in a Decision Table*

Note: Fit was used before the letters were assigned the meaning of Framework for Integrated Test. So while Fit is an acronym, it should not be capitalized.

The second format is often referred to as **Gherkin**, which is a business-readable, domain-specific language created to support the automated testing tool Cucumber. Gherkin is written as a set of statements, each one starting with a keyword.

Click here to view code image

```
Feature: A brief description of what is to be accom
a user story.

  Scenario: <A specific business situation>
    Given <precondition>
    And <precondition, if needed>
    And <precondition, if needed
    When <action>
    And <action>
    Then <testable outcome>
    And <testable outcome, if needed>

  Scenario: <Another specific business situation>
```

An Example

This example contains a set of examples from the conference submission system, specifically the ability for a reviewer to provide a review of a session proposal.

Click here to view code image

```
As Reed The Reviewer
  I want to review a session
  So That I can provide feedback to a submitter.
```

The Fit format is given in Table 14.7.

Reed's Review Track	Presenter	Co-Presenter	Session Submitted to Track	Can Reed Add a Review?
Experience Report	Sam		Experience Report	Yes
Experience Report	Sam		Agile Boot Camp	No
Experience Report	Sam	Steve	Experience Report	Yes
Experience Report	Reed		Experience Report	No
Experience Report	Sam	Reed	Experience Report	No

Table 14.7 *Examples for Conference Submission System*

Click here to view code image

```
In Gherkin:
  Feature: Add Review
  As a track reviewer
  I want to add reviews

  Background:
    Given I am logged in as "Reed"

  Scenario: Review a session
    Given a session exists on my review track
    When I add a review to that session
    Then the review should be added

  Scenario: Able to review draft sessions
    Given a draft session exists on my review track
    When I add a review to that session
    Then the review should be added

  Scenario: Unable to review for other tracks
    Given "Sam" has created a session on another tra
    When I try to add a review to that session
    Then I should not be allowed
  Scenario: Unable to review my own session
    Given I have created a session on my track
    When I try to add a review to that session
    Then I should not be allowed

  Scenario: Unable to review sessions I'm a co-prese
    Given a session exists on my review track
    And I am the co-presenter on that session
    When I try to add a review to that session
    Then I should not be allowed

  Scenario: May only review a session once and must
    Given a session exists on my review track
    And I have already reviewed that session
    When I try to add a review to that session
    Then I should be taken to the "Existing Review"
```

When to Use It

Examples are most frequently used to describe specific aspects of the solution, often as a way of providing further detail about the behavior of the solution in relation to a specific user story. Examples are very helpful when the team is automating their acceptance tests, but they can provide value even if the team is not automating their tests because it drives conversations around the solution's behavior in specific situations.

The different example formats tend to be better suited for different situations as well. The Fit format works best for business rules that have several inputs and/or outputs. Fit provides a way to lay out all the possible combinations of input variables and gives the team a chance to discuss what would happen in each case and, equally important, strike scenarios that won't happen in real life.

The Gherkin format is better suited to situations where someone is interacting with the solution. Examples in this form tend to describe the initial state of the solution followed by some action and the resulting state.

If your team is using a specific automated testing tool, that tool may dictate the example format you use. But if your team is using scenarios to help build a shared understanding of specific details, feel free to use both formats in whatever way seems most appropriate.

Examples are often used as ongoing documentation, providing a fairly accurate—and ideally easy-to-understand—reference of what rules the system enforces and expected interaction behavior that the solution exhibits. We make great use of acceptance criteria in the conference submission system as a starting point when someone reports a defect, or has a question about what the submission system does or should do. The examples represent the actual scenarios we accounted for because we associate automated acceptance tests with all of the code we write. When someone asks a question about the submission system, I first check the examples we have and find that when something doesn't seem to be working right, nine times out of ten it's because we didn't account for that particular situation.

Why Use It

Examples are helpful as a way of structuring the conversation about how a solution should behave in specific situations, remembering that discussion for future reference (i.e., documentation), and providing a way for the team to agree upon how to validate that whatever the team builds is working properly. It's very helpful to create examples as a group so that you can discuss and agree upon how the system should behave when certain scenarios are encountered. This may include discussing what type of messages the solution provides when a particular rule is violated, or actions that the solution does and does not allow.

How to Use It

1. Meet with your "three amigos": the stakeholder(s) interested in the particular user story, someone with a development perspective, and someone with a testing perspective. Meet near a whiteboard or flip chart paper. You may find that you discuss examples at the same time you are clarifying acceptance criteria.

2. Discuss the user story to determine what the stakeholder is hoping to accomplish.

3. If you created a mind map of your acceptance criteria, it may be helpful to refer to that when discussing examples.

4. For each particular interaction, or rule, talk about the various scenarios that could occur. These scenarios may include

• Happy path

• Negative path(s)

• Alternative path(s)

• Edge case(s)

5. For each scenario you identify, discuss whether that scenario will really occur. If it will not, disregard it. If it will, discuss the specifics, either the input and output values, or the precondition, action, and result, depending on which format you use.

6. Once you have identified all the scenarios that come to mind, discuss if the number of examples indicates the need to split the user story.

7. Note the examples in your repository of choice.

Caveats and Considerations

Acceptance criteria can provide the starting point for identifying examples. However, keep in mind that you do not need to create an example for all acceptance criteria. Examples are helpful for identifying clear ways to explain the acceptance criteria and conveniently lead to tests, but they are not essential for every specific item. Acceptance criteria, not examples, provide some clearer definition of the scope of a given user story, such as "How far does it go?" Examples provide deeper understanding of some aspects of that.

Create examples with real data you would expect to find in your solution.

Wait to figure the examples out until you are getting close to automating them. Examples may get into some specific implementation details and as a matter of course will tend to get identified closer to the delivery time frame, perhaps most effectively during three amigos discussions shortly prior to delivery.

Additional Resources

Adzic, Gojko. *Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing*. Neuri Limited, 2009.

———. *Specification by Example: How Successful Teams Deliver the Right Software*. Manning Publications, 2011.

Fit "Customer Guide." http://fit.c2.com/wiki.cgi?CustomerGuide (http://fit.c2.com/wiki.cgi?CustomerGuide).

Gherkin description in Cucumber Wiki. https://github.com/cucumber/cucumber/wiki/Gherkin.

Keogh, Liz. Collection of BDD-related links. http://lizkeogh.com/behaviour-driven-development/ (http://lizkeogh.com/behaviour-driven-development/).

———. *Behaviour-Driven Development: Using Examples in Conversation to Illustrate Behavior—A Work in Progress*. https://leanpub.com/bdd.