



To be agile at requirements modeling you need to be in a situation where it is possible to succeed, and for many project teams this unfortunately is not the case. Very often requirements modeling efforts are undermined by your environment – it is common to discover that an organization’s culture isn’t conducive to effective software development efforts or project stakeholders do not understand the implications of their decisions. In this essay I identify common challenges, or at least issues which are perceived as challenges, that many development teams face when it comes to requirements modeling and discuss potential solutions for dealing with those challenges.

These common challenges are:

1. **Limited access to project stakeholders**
2. **Geographically dispersed project stakeholders**
3. **Project stakeholders do not know what they want**
4. **Project stakeholders change their minds**
5. **Conflicting priorities**
6. **Too many project stakeholders want to participate**
7. **Project stakeholders prescribe technology solutions**
8. **Project stakeholders are unable to see beyond the current situation**
9. **Project stakeholders are afraid to be pinned down**
10. **Project stakeholders don’t understand modeling artifacts**
11. **Developers don’t understand the problem domain**
12. **Project stakeholders are overly focused on one type of requirement**
13. **Project stakeholders require significant formality regarding requirements**
14. **Developers don’t understand the requirements**

Challenge #1. Limited or No Access to Project Stakeholders

This happens time and again – for some reason senior management is willing to invest in the development of a system, sometimes spending millions of dollars on it, but is unwilling or unable to provide you with the right people to tell you what the system needs to do. This is often the result of a serious lack of understanding of how software is developed, that developers need active support of project stakeholders to be successful, or because they’ve always worked this way and simply don’t know any better.

The first step to address this problem is to communicate to your project stakeholders that you need to work with them closely, that they must invest some of their valuable time to work with you. In some situations there isn’t an identifiable user of the system yet, this is often the case

when you are developing a shrink-wrapped product for sale or a new web-based system for use by your (potential) customers, and therefore a surrogate user should be identified. Good candidates for surrogate users are marketing and sales staff that are familiar with your customer base, the person(s) who had the initial vision for your system, or the potential customers themselves (you may need to work with your existing client base to discover what they want of a new system or simply hire people off the street that fit the identified customer profile for your product). My experience is that you can always find someone to provide requirements for your system, and I've been in many situations where people were convinced they couldn't possibly find someone to work with. Remember that your project stakeholders include more than just direct users of your system, although also recognize that you are going at risk if you do not include some direct users in your software development efforts.

Challenge #2. Project Stakeholders Are Geographically Dispersed

An issue related to having limited or no access to project stakeholders, and often a reason why your access is limited, is because some or all of your project stakeholders are at another location(s) than the development team. This is a common problem for projects within large organizations, projects that have been outsourced to an external organization, or projects performed by a consortium of organizations.

There are several ways to address this situation. First, attempt to co-locate the developers and project stakeholders that can actively participate with the team – as I discuss in the **communication essay** the best way to communicate is face-to-face at a **POW**. If that fails your next best strategy is to have your project stakeholders on site with your developers on a part-time basis and available via other means the rest of the time (in my **communication essay** I discussed alternative means of communication such as email, videoconferencing, and the use of collaborative modeling tools). I was once involved with a 300 person project where some of our project stakeholders were on the floor actively working with the developers one week out of every three and available via phone and email the other two weeks and that worked reasonably well. The next best solution is to fly the developers to the project stakeholders and have them work with the stakeholders there. I've tried this on several projects, successfully, although have noticed that this is very tough on the people who end up doing the traveling (usually me, but luckily I like to travel). Another alternative is to place business analysts, people responsible for working with project stakeholders to understand their needs, at the disparate locations and have them work with the development team to define the requirements. Of course you can combine these techniques as needed.

Challenge #3. Project Stakeholders Don't Know What They Want

This issue should motivate you to spend some time modeling, exploring with your project stakeholders what their needs are and what is important to them. The business world is complex, project stakeholders will very often sense that there is a problem or opportunity that should be addressed but they don't know how. This is completely and utterly normal. Have you ever redecorated a living room, or simply re-arranged furniture? You know that you want a better living space, but you often don't know what you want. Perhaps you look in some magazines

at pictures of other living rooms, you visit furniture stores, or simply look at how your friends have organized their homes. If you've had difficulties defining a vision for a living room, imagine how difficult it must be for your project stakeholders to identify what they want a system to do.

Start by accepting that this situation as the norm. Tell your users that if they can't tell you exactly what they want that it's okay, all they need to do is tell you what they want you to work on right now. Staying with the redecorating analogy, perhaps you don't have a vision for the entire living room yet but you could start by focusing on installing book shelves. Focus on the aspects of the system that they have the best vision, model that small portion, and implement a working system (along the lines of the practice **Prove it With Code**) that they can work with and provide you feedback about. If they aren't able to identify requirements for even a small part of they system, you may not even be sure that you want books in your living room at all, then start by exploring how they currently work. By modeling existing work processes you will gain a better understanding of what they do and what their potential needs may be and worst case can suggest some potential requirements to them.

Challenge #4. Project Stakeholders Change Their Minds

Project stakeholders are people, yes they really are despite your darkest suspicions, and people change their minds. Ever arranged furniture in your living room into a configuration that you just knew was going to work, but when you stepped back to look at it you realized it wasn't what you wanted? If you've done that with something as simple as furniture arrangement what are the chances that your stakeholders will change their minds about something as complicated as the system you are building for them? One hundred percent.

The first step to addressing this issue is to accept reality and follow AM's principle of **Embrace Change**. The next step is to explore whatever is changing, that's what modeling is all about, because you will often find that either they didn't know what they wanted in the first place (see above) or perhaps you didn't understand what they were asking for. As you are working with your stakeholders you want to build an understanding of their perspective and their terminology, to both ensure that you do not have differing assumptions and to improve the quality of communication between you. I was involved with the development of an administration system for an e-commerce system, for which we had fairly specific requirements. The first week we put up two web pages, one that presented a home page containing menu items of critical functionality and another that enabled administrators to perform a high-priority function. We built exactly what our stakeholders asked for and when we showed it to them they realized that what they originally wanted wasn't going to work well and had us refactor it. Change happens. Finally, when you are exploring an area of your system that your users have changed their minds about you may discover that the real problem is that your stakeholders simply don't understand the problem they are trying to address, an indication that you need to hold some sort of visioning session with them, or individual project stakeholders have different visions that are not being properly managed. Your best bet is to adopt an **agile change management approach**, not a traditional **change prevention approach**.

Challenge #5. Conflicting Project Stakeholder Priorities

The system that you are building must reflect the needs of several constituencies, including direct users, senior management, your operations and support staff, and your maintenance developers. Each of these groups have different priorities and goals, as do individuals within these groups. There will be conflicts, conflicts that must be addressed.

Once again, the first step is to accept the situation. Somebody is going to have to negotiate to identify what the priorities for your system will be. Note what I just said – you need to identify the priorities for your SYSTEM. Each project stakeholder has their own priorities, and that's fine, but your goal is to settle on priorities for the system and that will often be different than those of individuals. One approach is to simply identify someone(s) who can represent the larger group of project stakeholders and let them work this out. You may also decide to lead the negotiation efforts yourself, but this can be time consuming and could result in ill-will towards you and your project team – my advice is to let someone else do your dirty work if possible. If the negotiation efforts fail then your last resort should be to call in the gold owner, the person paying for your project, and ask them to arbitrate a solution.

I once worked on an e-commerce system that was to be deployed internationally. Some of my stakeholders wanted to support several languages – American English, British English, Spanish, German, Japanese, and Cantonese at first – so we could serve each major market uniquely. Other stakeholders wanted to get our system up and running quickly and wanted to deploy it in American English only because they felt that this would be acceptable to our customers (okay, sometimes your project stakeholders are clueless). We couldn't come to an agreement on this issue so I had the issue pushed up the management chain and the decision was to support American English only for the first release and then evaluate the need to support other languages when it proved necessary (which it quickly did).

Challenge #6. Too Many Project Stakeholders Want to Participate

Sometimes you discover that you have too many people offering to work with your project team. This is often the case at the beginning of a project when excitement about it is too high or your project is a “political winner” that people want to be associated with.

The best solution is to thank everyone for their enthusiasm, to make them aware that you have more help than you currently need, that you have selected a portion of them to work with you, and that you will call on them in the future if you need their help. Whenever I'm in this situation I'll try to pick the best people for the job, looking for project stakeholders that are likely to provide the best insight and who are willing to invest the time to work with my team. I will also ensure that I do not alienate the people that don't immediately work with my team, we may want to work with them in the future to provide specific expertise or simply to act as a sounding board for what we're doing – often the project stakeholders who are actively participating with a project team will become too familiar with the system and lose perspective and be unable to identify potential problems, thus it is valuable to have access to a qualified outsider.

Challenge #7. Project Stakeholders Prescribe Technology Solutions

I've been in situations where a project stakeholder has said that we need to use a specific technology, such as "Oracle vX.Y.Z" to solve a problem when what I really needed from them was behavioral requirements such as "Customers need to be able to deposit money into an account". Yes, there are always technical constraints that your team needs to be aware of, perhaps what your stakeholder was really trying to communicate was the fact that Oracle is your organization's corporate database standard. Often times the real issue is that your stakeholders are having difficulty differentiating between requirements for a system and architectural alternatives for a system, perhaps the person is technically oriented and hence likes to focus on technical issues, or perhaps it's something as simple as them having just read an article in the most recent issue of Business Week that described a successful project using Oracle databases.

The best approach to dealing with this issue is to define the **rights and responsibilities** of project stakeholders to help put a framework in place which enables you to focus their efforts on defining what the system will do and the developer's efforts on how the system will do it. Another tack would be to ask the stakeholder "if you had that technology in place, what would it do for you that's important" or "how would you use this technology" in an effort to identify the actual requirements.

Challenge #8. Stakeholders are Unable to See Beyond the Current Situation

In many organizations people have been doing their jobs the same way using the same tools for years – they may not have ever seen another way to do things nor have they thought it could be different. They may also be scared of change, perhaps they're afraid they won't have the skills required to work with the new system or they will be replaced by it, and are motivated to formulate requirements in terms of what they are comfortable with.

The best approach is to talk about the current situation with them, to identify what works well and what doesn't, and to explore how the current situation came about. For example, you're working on the **SWA Online system** and are identifying requirements when one of your project stakeholders tells you that customers can only order ten things on any given order? What? Appears that's the way that it's always been done in this company and that's the way it has to be in the new system. What? Needless to say you decide to question her about this and soon discover that the current system is based on a process where customer service representatives input orders that come in via mail and fax, and the order forms only have ten lines on the page and knowing this the builders of the original green-screen system built screens that only allowed ten order items. You point out to her the true reason for only supporting ten order items, lack of room on the paper form, and show explain to her that you can easily build a system that allows more than ten items. After awhile she realizes that it's possible to do so, particularly when you visit the web sites of your competitors and see that they don't have this preset limit.

Challenge #9. Project Stakeholders Are Afraid to Be Pinned Down

Project stakeholders will sometimes give vague requirements because they don't want to commit to a specific answer. The problem is that they're afraid to be wrong – their previous experiences with serial approaches has taught them that it's too expensive to reimplement a changed requirement and therefore you have to get them correct up front. They very likely sense the impossibility of doing so and therefore choose not to commit.

To address this problem impress on your project stakeholders that you are ready to embrace change, that you are taking an iterative and incremental approach that reduces the cost of change, and that your goal is to identify and implement the best solution for them which means that some requirements will evolve over time. Then do exactly as you say, be receptive and supportive of change – over time your stakeholders will learn that they can make a decision today and safely change it tomorrow if needed. When you deliver working software each iteration, software that your stakeholders see evolves as their understanding of the system evolves, their fear of committing will quickly dissipate.

Challenge #10. Project Stakeholders Don't Understand the Artifacts You're Creating

The vast majority of project stakeholders, and I suspect the vast majority of developers, have not received a formal education in modeling. It is very likely that they don't know how to read a UML activity diagram, or a data model, or a UML use case diagram because that's not a skill that is required of their day-to-day job. The problem is that they need to understand the artifacts that you are working with in order to understand what it is that you are trying to communicate to them and to become active members of your modeling efforts.

Your first step is identify artifacts that your project stakeholders will need to understand so you know which ones to focus on. This is where the practice **Use The Simplest Tools** helps you – if you strive to model with simple tools such as index cards, paper, and **POWs** you reduce the learning curve for your stakeholders. The second step is to teach these techniques to your stakeholders, I prefer a just in time (JIT) approach where I do a brief tutorial during a modeling session on the proper application of a technique when we first need it and then dive right into applying the technique at that point (I'm also a firm believer in hands-on experience). I've also given brief tutorials that overview common modeling techniques at the beginning of a project to give stakeholders a feel for what they'll be doing and provided them with reading material, typically **The Object Primer 3/e: AMDD With UML 2**, that describes the techniques in detail. Following the JIT approach to training I've taught techniques such as CRC modeling and essential UI prototyping, both of which use simple tools, in less than fifteen minutes each to project stakeholders that had never modeled before. I've taught more complex techniques, such as flowcharting or class modeling, to stakeholders over a period of time so as to build up the techniques slowly. The third step is to build working software based on their models, providing concrete feedback as quickly as possible. When you do this the models are no longer abstract, your project stakeholders readily see that their essential UI prototype created from flip chart paper and Post It notes has become a functional HTML page and that their **CRC card** describing the concept of a customer has been reflected in the functionality of the software.

Challenge #11. Project Stakeholders Are Overly Focused on One Type of Requirement

Sometimes you will find that your project stakeholders are providing you with great usage requirements, perhaps in the form of **use cases** or **usage scenarios**, but are falling short when it comes to non-behavioral requirements (or vice versa).

This is a very good indication that you aren't working with the right people, perhaps you're missing someone that represents a major constituency such as your operations and support staff, and therefore need to reconsider the mix of project stakeholders that you're working with to model requirements.

Challenge #12. Developers Don't Understand the Problem Domain

A common problem at the beginning of a project is that the developers don't understand the problem domain, making it difficult to communicate with your project stakeholders. This is understandable, just like it isn't part of the day-to-day job of a stakeholder to understand modeling it isn't part of the day-to-day job of a developer to be an expert at jobs of their stakeholders. This is why both groups of people need to actively participate in requirements modeling, they both have something of value to offer to the overall effort because everyone can learn from everyone else. Overtime, by **modeling with others** and pair programming, developers will become **generalizing specialists** with an understanding of both software development and the problem domain.

Developers need to invest the time to learn the domain, they'll eventually learn it as the project progresses but will often find that this isn't quick enough and therefore must do something to enhance the learning experience. Years ago I worked for a company that truly understood the importance of developers understanding their business. The first three days that I worked for the company myself and another new hire sat through training sessions presented by Vice Presidents, this was a multi-billion dollar financial institution, who described the fundamentals of what their divisions did. One Vice President trained two, yes two, developers for several hours in each session. If your organization isn't this enlightened then I've found that simply picking up a book that overviews the domain, introductory college text books are ideal, are a great starting point to learn the fundamentals. I'm also a firm believer that developers should read widely. I read The Economist and National Geographic on a regular basis in addition to publications such as Software Development, The Communications of the ACM, and IEEE Software. By reading widely I have a broad knowledgebase from which to work when I get into a new situation.

Challenge #13. Project Stakeholders Require Significant Formality Regarding Requirements

Many stakeholders perceive formality – scheduled meetings, official requirements documents that they review and sign off on, and formal presentations – as inherent to professionalism. In my experience this expectation is often the result of the serial software process mindset that our industry has had for the past two generations, our stakeholders often don't know that there

is another way to work. It is also the result of the bad relationship that the IT community has with the business community – our project stakeholders don't trust us to deliver and insist on greater formality in the belief that it gives them greater control over the development process which they often don't understand.

To address this issue you need to communicate that there is another way of operating, a more agile way, and communicate the inherent downsides (slower development, less chance of understanding the requirements, greater cost, ...) of their current approach. Ask your stakeholders what their real goals are. Is it to develop a working system or is it to have meetings and produce documentation? If it's the first goal, which it should be, then ask them why they insist on this formality. Don't accept their initial answer and keep digging until you're at the root cause. As I indicate in **Agile Documentation** chances are very good that they don't trust you and this is their way of holding your feet to the fire. The next step is to ask them to trust you, which is often difficult for them depending on how bad their previous software development experiences have been, and to remove some of the formality. Then deliver working software, showing them that it is possible to be successful without being burdened by too much formality. Iterate through cycles of reducing the formality and then delivering working software until you reach a point where you can work effectively.

Challenge #14. Developers Don't Understand the Requirements

A common complaint on non-AM projects is that the requirements artifacts that are being created by the business analysts with their users aren't understandable by the developers expected to implement those requirements. Luckily, this is a problem that you should not have on an AM project.

First, you should know your models and know your tools, implying that developers should have sufficient training and experience with the artifacts being generated and the tools used to do so. If they don't then they must be given sufficient training and mentoring, removing the problem of simply being unfamiliar with the techniques. Second, the principle **Incremental Change** advises that you work on your system in small chunks, building it up over time, a philosophy that is reflected in the practice **Model in Small Increments**. This avoids the problem of developers being swamped by a large requirements document that defines too many requirements to comprehend all at once. Third, the practice **Active Stakeholder Participation** ensures that your project stakeholders are available to explain their requirements to developers and that developers are open to doing so. Fourth, the practices **Create Simple Content** and **Depict Models Simply** ensure that your requirements models are as understandable as possible