2015

Successful Demos

STRATEGIES FOR BEFORE, DURING, AND AFTER

JAMES WUCHER

THE REALTIME GROUP

Contents

Overview

At no point in your career will the expression “say what you do, do what you say” have a more personal and visceral reaction than the day you discover you have signed up (voluntarily or otherwise) to deliver a demonstration of a product your team is developing. The “demo” is the opportunity for your team to shine, casting a warm glow on all who behold what they have wrought. A successful demo rectify past mistakes, generate customer satisfaction, invigorate your development team, and capture critical future funding. It can also be the moment when the project is at its greatest risk of being misunderstood or cancelled, even when the demo was not originally intended to be meaningful. This white paper outlines some best-practices and ground rules for planning and executing the demo so that the glow is from its excellence and not from going down in flames.

Why Demos Matter - Seeing Is Believing

At the outset, you have to accept that executing a demo is not a proposition that will be greeted with great enthusiasm. Your team may chafe at the prospect of bringing features into the light of day before they are polished. Your management may still want you to deliver internal milestones in parallel with the demo timeline. Why would anybody want to stand up in front of a crowd of people and show them something only “half baked?”

If you stand in front of investors and show them the Power Point slides about your new product, you may get some yawns, some feedback you can use, and maybe even some long term interest. If you take the same opportunity, but show them a working prototype and then let them “play” with it, you will get a tsunami of a reaction compared to the trickle before. They may love it. They may hate it. But it cannot be ignored as some “far off” possibility. It is a fact of human nature that having something tangible in front of you lends far more “weight” to the reality it represents than a mountain of presentations and status reports.

What Constitutes a Demo

For the purposes of this paper, a “Formal Demonstration” (Demo) is a planned execution of a system under development in front of stakeholders in the project. The general ideas in this paper are not limited to this particular instance. Informal demonstrations in front of your peers (who may or may not be friendly) or at your desk in front of your manager are a staple of solid developmental practices. We use a combination of internal and external demonstrations on all our major projects.

Demo Motivations

The very act of delivering a demonstration is an exhilarating, exhausting, and unnerving experience. Why would you plan on having demos at any point except as the last milestone? Why go through the trouble of going through the extra work? Some of the work may not even be the final product? It may even get thrown away entirely? So why bother?

Going beyond the “seeing is believing” theme of the last section, programmatically, there many solid reasons to hold demos at regular intervals over the course of your project:

Driving Forward Development

First and foremost, the prospect of an impending demo will drive your development and your development team to move the project forward. Remaining mired in uncertainty or "analysis paralysis" are simply not options when there are near-term timelines with solid deliverables to be demonstrated. Knowing that you must do "something" and knowing what that something will remove a large number of progress blockers.

To enable this requires that the scope and goals of the demonstration be realistic and technically feasible. There is a huge motivation difference between "We think we can" and "We are not sure we can". Naysaying and sandbagging aside, if the team really believes they will not make it at the outset, they are probably correct. Having a successful demo can be a boon to the morale of a team which will carry through to the next stages. Creating a "death march" situation where the team does not believe "they can get there from here" will have the opposite effect.

Fiscal Responsibility

A client that receives a steady stream of invoices without seeing proof that progress is being made can easily wander into doubt-filled territory. Having planned demonstrations at regular intervals reduces the chances of such wandering and also gives the client a chance to directly influence the direction the project is taking. This also serves the purpose of solidifying use cases and customer expectations when requirements are weak or vague.

Human Factors

In the immortal words of Confucius: "I hear and I forget, I see and I remember, I do and I understand."

You have a product to build and you want to get your audience to interact with you, to get involved in the process of creating it for them. This is a difficult proposition when your product is just an idea. A few slick glossies may get you a bit further down the road. But to engage them, you must give them something to "play with". Something they can pick up, press, touch, see, and occasionally break.

Driving Brand Identity

A demo is not just about the product, it is also about "you" as a developmental organization. Showing up, executing well, answering questions, dealing well with inevitable unexpected and unintended consequences, all build confidence in your brand while building confidence in your development. "Doing what you say you can do" is a powerful statement about your brand.

You cannot buy this kind of marketing on a billboard. It has to be experienced. If it is done well, they **will** remember you.

Doing the Science First

Before the user interface looks "beautiful" or all the blinking lights start pulsing, there is a core set of features that make your product special and economically viable. This core set of features (see Demo Preparation Fundamentals) constitutes the main offering that your product will offer. The first goal of the project should be to drive out all the "unknowns" in how these features will be implemented.

Having a planned series of demonstrations where the execution of these features is central is a solid way to work through the problems before they become problems.

Organic Technical Debt Reduction

The development of complex systems does not follow a strictly linear flow. Many individual components have to come together for even basic features to manifest. Because of this, a component may be developed which "works", but perhaps, does not work as well as is intended, at least not in the first round of implementation.

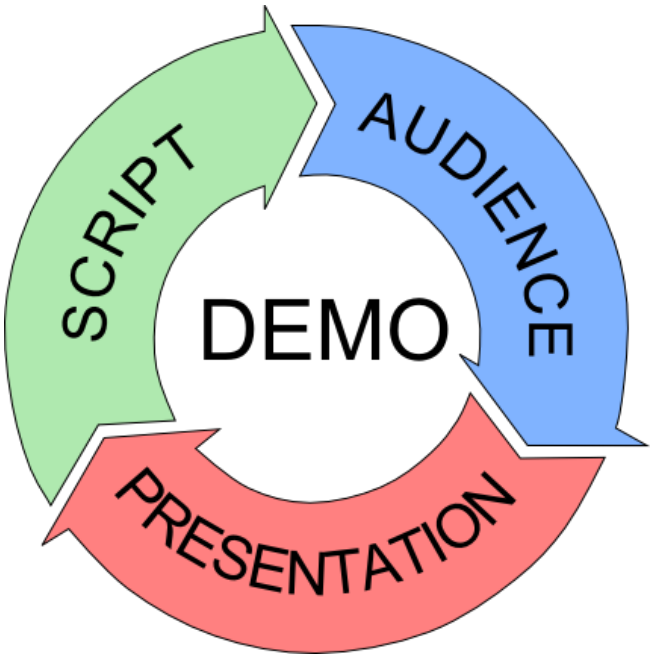
Consider the case of a system that displays a web page of dictionary entries. Dictionary entries are in alphabetical order, and algorithms that "sort" text into order are well known. Some are very fast and complex while others are slow and trivial to create. A developer may implement a "slow" sort initially because they just want to get it done so they can get the system working. They will come back at some point in the future to enhance it, but not "today". In doing this, the project has accumulated a "technical debt", a piece of work that will have to be improved at some point in the future for the success of the project (and which probably was not planned on initially).

Holding a demo for the system will either show that (1) the system's response time is adequate, in which case there is no need to improve it and the technical debt has been removed, (2) the response time is inadequate and must be addressed now, or (3) it is inadequate but can actually be postponed till later. In (3), at least it is known that it needs to be addressed later and can be scheduled, so it does not occur as a surprise.

The demo has forced a potential issue to the surface when it may otherwise remain unhandled or ignored till much later. Your teammates may forgive a slow response time, but your stakeholders are a different story entirely.

Demo Decomposition

Before venturing into the execution strategy for successful demo, it will be helpful to take a step back and look at the demo from the perspective of the interrelated parts: The Audience, The Presentation, and The Script.



As we move forward, every action taken towards executing the demo will touch on one or more of these.

The Audience

The Audience, aside from other the discussions about “other motivations” for holding a demo, is the primary reason for holding the demo.

Knowing your audience is important for a number of reasons:

1. It will set the technical detail level of your presentation. For example, suppose you are giving a demonstration about a new computer program that will increase productivity dramatically. The people who use that tool will have many questions about the tool itself and how it works in their day-to-day job. If you are giving it to the management team, you have to discuss the consequences of using the tool such as the potential increase in productivity, deployment time, training costs, long term support, product cost, etc. If it is a group of investors for your product, you are trying to convince them about the financial opportunity the product represents to them. The composition of the Audience will lead to decisions about the content and key points of the demo.
2. It will affect the makeup of the team giving the demo. Users of technology, creators of technology, and management often speak “different languages”. You want to pick the composition of your demo team, especially the person who will be your “front person”, based on their ability to communicate to the target group in your audience.
3. It will help you plan ahead for “weak” points. Knowing that <fill in your favorite executive> will be in the room (also known as “Sparky”) will allow you to plan accordingly (static mats, rubber gloves, maybe just a big “Do Not Touch” sign).

The Presentation

The Presentation is what your audience is going to see and experience. When putting together the presentation, consider the following:

1. Amplify the impact of the visual. Sometimes a blinking light just looks like "a blinking light." But if it is "the point" of the demo, you must ensure the audience understands that.
2. How is the demo going to look from the perspective of the audience? If you are showing them something that they cannot see from the back row, should you project it?
3. If part of your demo content includes a presentation, is it easy to read or filled with lots of extraneous text and details the audience will not care about? Has it been checked for spelling and grammar? Are the numbers in it accurate?
4. The presentation approach has to help the audience move "through" the demo. It has to set them up for what is going to happen (agenda), let them know what is going on while it is happening (e.g. "you are watching it bend, fold and mutilate your child's report card"), and let them know when it is all over.

The presentation can make the difference between a successful or failed demo, even when the product works perfectly.

I once took part in a demonstration for an acoustic system that lifted me out of my chair and sat me in a box where Taylor Swift was singing directly to me, acoustically. While most demonstrations are visual, the impact of the presentation in this case was profound: I got "it" (the effectiveness of the product) immediately. It does not matter whether or not I like Taylor Swift; it matters that the **presentation** was extremely effective. That is the kind of impact you want to make.

When you do a good job of showing them that you are ready and describing what is going to happen, the audience will become vested in your success.

Finally, if you have made concessions in order to get the demo on track (e.g. substitute parts or functionality stubbed out), bring the audience in on this early on and communicate how this does not affect the main goal of the demo. This will build trust. If they think it is important and you take a "let's not discuss that" approach, it will become at the least a liability, if not an actual "confidence erosion event" for the demo.

The Script

The Script is what you plan to say to the audience and what you plan to show them while you are saying it. It should be obvious, without stating it, that you should rehearse the script. That being not stated, you should rehearse the script. Many times. With your team. With your bosses. If necessary, with disinterested strangers. You should record it and watch yourself, if possible.

Aside from this, there are several other considerations for the script, with regards to its construction:

- Is it too long or too short? Most people have an attention span of roughly 50 minutes. Consider this when defining the scope of what you plan to show and discuss.

- Are there opportunities for the path of the script to be derailed by a single failure? If so, plan for those points and have at least one contingency ("we won't show this feature if this happens but we can just reboot the box and start from point 13").
- Plan for cycles of "preamble, show something, and reflect on what just happened". This will set a good cadence for the audience (similar to setting the rising and falling action in a story). It will also set a good cadence for the team giving the demonstration.
- Avoid dead time. Standing for 3 minutes waiting for something to power on is going to be a stall point in your demo. If there are going to be unavoidable gaps, plan to use these as exposition periods, shorter discussion sessions, etc. The goal is to keep the audience engaged while you set up for the next "magic trick".
- Prepare for the obvious questions. Even if you do not answer them, be prepared to.
- Who is the Audience?
- What technical detail level will the Audience be comfortable with?

Demo Preparation Fundamentals

Before the main work for the next demonstration gets underway, you must make sure you and the development team have asked the basic questions regarding the work.

Question	Discussion
What will you show?	These can be requirements, use cases, specific scenarios, etc. Before you begin work, you need to ensure these are well established and understood by all the stakeholders, including the team that will implement them. The features you will show at the demo (referred to as "core features" in this document) will have to be picked based on the expectations of the demo viewers (customers, your boss, shareholders, etc.), the current state of the system, the time available till the demo, the reality of getting those features functional in time, etc. It is easy but not entirely practical in general to say "show everything". It is also impractical to expect to "show nothing" and have a successful outcome. It is usually better to under promise and over deliver, though there may be negative consequences in the near term. Over promising and under delivering also has consequences, though these usually show up later. Striking balance is the key. Instead of asking "what can the system do", it is often better to ask "what use cases will the audience be interested in." This will lead you to a set of features that must work (at some level) in order to make the demo work at all (think Minimum Viable Product).

When will you show it?	The time and date of the demo should be known well ahead of time. Will anybody be on vacation?
How will you show it?	Are you going to have a tabletop demonstration with blinking lights or will the missile be launched into the air and strike a drone? Setting the expectation of your audience is a key factor of a successful demonstration.
Who will be at the demo?	The demo could be internal only, just for your team. Or it could be in front of investors in your project. These two groups may have vastly different ideas about what is "important" in the demo, and this must be taken into account. It is also important to understand that "tech talk" is great for technologists, and generally boring for everybody else.
Who will be giving the demo?	Having practice sessions with your team will help you to make this decision. **Do not let it happen by default.** The "guy who knows the system best" may be a terrible presenter, but he can be on standby for the "guy who speaks most eloquently."
Are there other dependencies?	Does your demo depend on another team showing up to do their part? If you know this, you can manage this. If you do not know this, you need to find out. There may be internal dependencies as well (e.g. availability of new equipment, new boards, etc.) These are all factors in your timeline that have to be managed. **Dependencies outside of your team's direct control are a risk that must be taken very seriously**. Planning on timelines for these dependencies and contingencies if they are not met is not only prudent, it is required.
Where is the demo?	If the demo is in your facility, you have to ensure a room will be available well before the demo to set it up. Unless it is impromptu or infeasible, having a demo at an engineer's desk is probably not your best choice for "high impact". Consider your audience. If the demo is off-site, you will need to ensure all the equipment you need for the demo will be at the location well before the demo so you can set it up, make sure it works, etc. You will need time before the actual demo to survey the room, set up the equipment, place any handouts or other media, etc. True Story: In a former life, we worked on a device that, among other things, acted as a WiFi access point. For a seemingly innocuous demo, it was taken to a trade show floor. Where there were hundreds of people. With laptops. Actively trying to connect to the open WiFi node our device had on it. This was not a good day.

	When considering the demo location, you need to identify the environment and assess not only how will affect the demo, but the system being used for the demo.
What are the stakes?	Every demo will be taken seriously, but some demos will be taken more seriously than others. Success could mean the continuation of a contract, the start of a new business, funding of your project, or it could have little impact at all. The consideration here is not the importance of the demo, but the flexibility in timeline and the level of performance. You can probably move a demo for your boss a day or two, but not for investors (who will view it as a red flag on the ability to execute).

Powers of Two

Time management is fundamental to the success of your demonstration. Not only doing the right thing, but doing it at the right time is a key factor. Assuming that you have just gotten started and you are either pushing for or pushed into having a demo, there are three key times before the demonstration that you should focus on: Two Months, Two Weeks, and Two Days.

This timeline is somewhat idealized (i.e. if you could choose to have two months to prepare, you would). Real projects run on schedules subject to many forces, rarely scheduled around what is convenient for you. You may have a demo scheduled a year from now. You may have one tomorrow that you just found out about.

The "Powers of Two" schedule can be compressed or expanded as needed to fit your particular timeline, within reason. It is very unlikely you are going to compress two months of preparation into twenty minutes and have a good outcome. That being said, the general flow is about the same regardless of the time allotted; you just have to decide which pieces are most important given the time allotted.

This timeline should be understood and tailored to your specific needs. This means you will have to read and incorporate the necessary parts into your plans before you get started. **Do not wait till "two days" before the demo to read the section on what you should be doing two days before the demo.**

Two Months

Two months before the demo is the time to start preparing the plan for what you will show and how you will show it. The following major tasks should be performed:

1. Identify the core features you wish to be demonstrated, if they have not already been defined for you.
2. Examine the general state of the system and determine if the targeted feature list is viable. Your goal is to have all these features in a working state by two weeks before the actual demonstration date.
3. Verify the timelines of other stakeholders and contributors, especially for other pieces of the system that will require extensive shaking out (e.g. new hardware). Depending on when these external

dependencies are coming into the mix, you may have to hold off on showing the system with the "newest and greatest" in favor of showing the system with "older but working". A component that needs to be "shaken out" should not be introduced unless it is at least a month away from the demo.

4. Write the rough outline of your demo script and go over it with your team. They need to know what they are working towards and you need their buy-in that it can be done.
5. Resist the temptation to change the scope of the demo unless good project management principles are followed.
    a. Increasing the scope without increasing resources or demonstration timeline appropriately will leave you in a situation where you may over-promise and under-deliver. This is not a success oriented strategy. Also consider that a last-minute change in scope, even a minor one, is a risk. It is best to have no issues, but it is better to have one you know about and can work around than one you don't know about until it is too late.
    b. Decreasing the scope is usually either a time/cost-saving tactic or a genuine change in scope for the project, and not a request from a customer to have "less". Bear in mind that it still takes a minimum amount of time ("nine women cannot make a baby in a month") to get "something working". A scope reduced to "zero" can be shown immediately, anything more will still take some time.

Two Weeks

At this point, all new components have been integrated. New features have been worked into the system and are roughly working.

First and foremost check EVERYTHING in to the configuration management system and TAG it. This should be done at the beginning of the two week period.

The following activities should be executed:

1. Fine tune your demo script by executing it against the system. Have several members of your team execute it as well. The table below shows the kinds of results you should be looking for:

Execution Result	Options
Features work as expected.	Can they be improved slightly with a small amount of effort? Do they need to be?
Features work, but are "rough" around the edges.	Decide on how to improve them or accept them as they are and work on improving other unacceptable results.
Features don't work, but system is stable.	You ran out of time or just forgot to get something into the system. If you have enough time to finish it in the following week, you may decide to take the gamble. Or you may decide to accept it, be honest about it at the demo, and focus on other features that are working well.
Instabilities, anomalies, and the unexpected.	Your options here depend on how serious these issues will affect your demo. • If you have complete control over the demo and you can steer it away from instabilities, take the safer path. If not, you will

	either have to warn your audience or use your best showmanship to smooth over the situation if it happens during the demo. • If you have time and hammering down a few oddities seems realistic, you should do so. If you have to choose between stability and aesthetics…this is a question for your audience.

2. Work with your team and experiment with what happens when they go "off the script" for the demo. This will give you the opportunity to decide on how much latitude your audience will get at the demo (if this has not already been decided). It will also give you a chance to shake out a few more bugs in the system. It is invaluable to watch other team members execute your demo script and see how they interpret it, where they go off it, and what events occur as they execute the script.
3. During the first week only, only very-small-low-risk changes are allowed. Every change should be atomic and reversible. And only for the purpose of addressing something directly for the demo (no core features that are off the demo script). Every change should be checked into the configuration management system at a good working point.
4. After the end of the first week, "Shrink-Wrap" your system. Tag all the code in the configuration management system, put a version of working hardware and mechanicals aside and make sure that (1) your demo can be run on this and (2) nobody can touch it. This is your "golden" system for the demo. No matter what, if you do any more polishing, you have something that works.
5. Create your presentation for the audience (e.g. Power Point). At the very least, you should have an "Agenda" so your audience is prepared. This will also focus you on your script.
6. Rehearse! Rehearse! Rehearse! Practice your script with the system as it is. This will let you get in tune with what is and is not working. Practice in front of your teammates and get their feedback.
7. Document the state of the system and the expectations of the demo. Any items that are not known to work should be documented, as well as substitutes for expected components (e.g. COTS part to replace real hardware not ready yet).

Two Days

The day of the demo has nearly come. At this point, you and your team should only be dry-running the demo. **NO CHANGES TO THE SYSTEM SHOULD BE MADE AT ALL**.

Below is a checklist of items to complete.

Done	Have you...
	Ensured all the stakeholders are going to be at the demo?
	Ensured the venue time and place are known to all the stakeholders?
	Shipped all equipment to the venue (if you are not carrying it in)?
	Reminded all team members that are going with you that the demo is in two days?

	Gathered all materials to be delivered with the demo?
	Documented the state of the system (release version, etc.) and the purpose of the demo?

Demo Day

The day has finally come. This is not a day to be unrested, over-caffeinated, or unprepared. Your demo may be flashy, and this should be a day for your team to shine. You want your audience focused on your demo, not on your droopy eyes, restlessness, or mumbling. You want to show them the fabulous work your team has done and leave them waiting for the next demo.

Demo Day Outline

Listed below is a general outline of the big steps you should take during the demo. There may be some deviations, but most demos have at least these as a minimum and are considered central to "giving a good presentation":

Agenda

You want to start out by letting your audience know what is going to happen and when it will happen. Whether you speak the agenda as a few short sentences or actually present it on a display (preferred), you must have one and let your audience know what it is.

- The agenda will let your audience know what to expect.
- The agenda will let you tamp down a rush of questions during the demo, as you have told the audience there will be an opportunity to get to them at the end. You will still get a few, but you can decide on the fly to answer them or to ask them to hold on till the later.
- The agenda will set the tone by showing you are prepared and take the demo and the audience seriously.

Follow Your Script / Do Your Demo

- Give the demo that you have prepared and rehearsed.
- Resist the temptation to go off the script. If you want to show something else, do so after the main demonstration. The audience will forgive an excursion that goes awry, but if it happens during the main flow, you may not have a smooth recovery.
- Answer questions with "short answers" (< 60 seconds) if you see fit. Do not get into longer answers or extended/detailed conversations with a single person in the audience, unless this demo is "just for them". You do not want the rest of your audience to watch a personal interaction, you want them to watch the demo. Postpone longer questions until either the Q&A section or till after the demo entirely.
- Have a member of your team take notes about any anomalies, audience reactions, feedback proffered by the audience, questions asked, etc.
- Always "keep your cool", even if you have a total meltdown of the system.
- Always be polite, professional, and upbeat.

- "Um" is not a word.

Reiterate Key Points

- Tout your successes.
- Discuss, as needed, but do not belabor, any shortcomings.

Future Plans

- Discuss planned improvements.
- Discuss the timeline for the improvements.

Questions and Answers

- Answer honestly.
- Be prepared to hear feedback you do not like.
- Be prepared to answer hard questions. If you do not know the answer, do not lie or "um" your way through it. Tell the questioner you do not know and will get back to them. Some credit is better than discredit.
- Be sure a member of your team is taking notes about the questions and answers.

Thank Your Audience

Before leaving the "stage", be sure to thank your audience for their attendance, their questions, and their feedback.

After Demo Day

Just because the demo has passed does not mean it is "over". The most important part of "doing what you say you do" is, not surprisingly, actually doing what you said you were going to do.

1. If there were any questions that you gave a postponement for, you must research it and give at least a minimal answer. Leaving lingering questions fosters doubts and damages your credibility. Whether or not these answers go to the entire audience or the specific person who asked the question depends on how sensitive your audience is to the answer. The team's representative (usually the PM) will be responsible for ensuring the information is disseminated properly.
2. If there were any holes or bugs found during the demo, you must add these to your work list. It is reasonable to dismiss your system failing because the user actually did set it on fire, but not because the pressed one key followed by another that you did not expect. Do not hold to the belief that poor performance of the system is a "user problem", it is YOUR problem.
3. Discuss the questions and performance of the demo and your presentation of it, with your team. What could be done better? What worked? What did not work? Why?
4. Follow up with key stakeholders. Ask for their feedback directly. This responsibility is assigned to the team's representative.
5. Begin the update cycle on requirements for anything exposed at the demo.
6. Start planning for the next demo.