

TRR Product Preparation Workshop

Mastering Project Demos,
Transitioning within ISD

Software Engineering

CS577b, 2001

Mastering Project Demos

- Project demos are critical to the success of your project and your future career. Take them as seriously as a final exam or more!
- What we'll discuss today:
 - The TRR ARB
 - Demonstrating requirements
 - Demonstrating utility
 - Demonstrating process
 - Organizing your demo
 - How to “WOW”
 - What you'll need to have at the TRR



TRR: Your Last ARB

- There's one more fun presentation in store for you: the transition readiness review (TRR), AKA your project demo.
- The project demo has various goals:
 - Show the audience that your product works as expected.
 - “Sell” the audience on your product (show “value”).
 - Go through some relevant parts of your IOC documentation (show “quality” development).
 - Describe your transition plan, and/or status (provide confidence that product is or will be ready to be used).
 - Mention support requirements/plans (show sustainability).

TRR Presentation Facts

- About the project presentations:
 - Format your presentation like a “real” product demo; That is, try to sell your audience on the product, by demonstrating its features and talking up it’s business case.
 - You must actually demo something. Use slides and discussion to augment, set up, or clarify only.
 - The professors, TAs will be there, along with your customer. You may invite others as well, if you want.
 - Each and every group member has to participate (i.E., Speak) in the presentation.
 - Don’t present the 5 LCO/LCA MBASE documents (OCD, etc) again. Only show major or critical changes.

Demonstrating Requirements

- Part of demonstrating your project is to show the audience what it does. Preferably, you'll show that what it does matches the requirements.
 - Therefore, you might focus your presentation around your system levels of service and capabilities, as in OCD, SSRD.
 - Show, directly or indirectly, that your system meets its primary (visible or otherwise) capabilities, goals.
 - Don't be too exhaustive with minor requirements.

Demonstrating Utility

- Along with showing off your system, make a case (think sales pitch) for its usefulness.
 - Show, directly or indirectly, why your system is worthwhile to the customer and/or end users.
 - Use your business case analysis from FRD: does your system yield the gains you promised?
 - Use “current system shortfalls” from OCD: does your system improve on its predecessor?
- Enthusiasm can make a difference.
 - Do you believe your product is valuable?

Demonstrating Process

- In convincing the audience that your product is good, you may want to bring in some discussion of your awesome SE skillz:
 - Use a portion of your IOC presentation to show your production and construction phase decisions (i.e., IOC documents).
 - Show how much testing you did.
 - Describe anything odd that happened (dropped requirements, outstanding bugs, architecture issues).
 - Discuss your quality management techniques.
- If your project doesn't quite work right, you'll want to draw on your IOC in explaining why, the impacts, and what (if anything) to do about it.

Organizing your demonstration

- You **MUST** organize your demo
 - Do not try to “wing it” or ad-hoc present
- Possible organizing themes for your demonstration:
 - A series of use cases.
 - A series a demonstrations of particular features (good) oriented around a realistic example.
 - A formal summary of your production effort
- Know your audience and orient the demo towards them (not yourself).
- Three words of advice: organize, Organize, **ORGANIZE!**
 - Your demo should be well prepared, well rehearsed, targeted and to-the-point, and flow “naturally”

Good Technique

- Some good ideas for your product presentation:
 - Have your system available for live demonstration (this is a must).
 - Use slides (but not too many!).
 - Create a script (who, when, which jokes, etc), and rehearse it.
 - Every group member should present something. Work together; Have fun and be interesting.

Wow: to do

- Things to do for Wow!:
 - Juxtapose your system and predecessor (i.e. compare and contrast)
 - Demonstrate a really cool feature.
 - Demonstrate an implemented evolutionary requirement (or any other instance of doing more than you had to).
 - Script your presentation for maximum effect (hook, exposition, rising tension, climax, denouement).
 - Describe a particular implementation element that your audience will consider difficult (i.e., impress with your godlike coding, design, or management skillz).

Wow: not to do

- Things that will not be Wow!:
 - Excessive description of implementation.
 - Boring stuff.
 - Avoiding hard questions.
 - Carefully not revealing bugs.
 - Demonstrating a visually ‘unappealing’ product.
 - A disorganized/unprepared presentation.
 - A very long presentation. Be aware of how much time you’re taking, and be willing to leave some things out. Presentations should not go over an hour. Leave time (and expect) comments, questions, and discussion

Demonstration as song+dance

- Remember that you're selling your product to your audience:
 - Let your pride in your product show. Convince your audience why your system is the best.
 - A certain amount of showmanship and flashy (but tasteful) presentation will serve well.
 - Remember to smile! 😊

Demonstration as technical review

- In terms of the class, the product demonstration is your chance to show off the difficulties in your system, and all the hard work you did.
- It's also the chance for the attending support staff (i.e, the poor buggers who get to maintain your stuff) ask pointed questions about your implementation and process.
- Be prepared to defend (give rationale for) your system, and answer pointed, specific questions about it.

Summary: Things to Have in Presentation

- Specific requirements for your presentation:
 - Your product! (i.e., a fully working version)
 - A salesman-like discussion of your project's usefulness, from your business case, etc. Why is the system going to be Really Great™ for the customer?
 - Transition issues: if you've delivered your product (you should have by presentation, if possible), how did it go? If not, when? Go through your transition plan.
 - Support issues: how will you support the product, once it's deployed (next term, for instance). It's ok to say that you will never touch it ever again, and everything's up to the customer. 😊

Transitioning Within ISD: Unsupported Machines

- ISD allows most anything to be developed and deployed on “Unsupported” machines
 - Although there is some installed resources such as web servers, Java, and databases, ISD will not guarantee anything. You must install anything that is critical.
 - Must not violate university regulations
 - External access from outside USC IP’s may be limited
- Your customer will ultimately be responsible for maintaining the system on an unsupported server
- Some resources can be found at:
 - <http://www.usc.edu/isd/doc/>
 - <http://www.usc.edu/uscweb/authoring/>

Transitioning Within ISD: Production Machines

- If your customer wishes to deploy your system within ISD and utilizing ISD support you will need to plan the transition carefully.
- ISD has a particular process they follow in regards to transitioning new software to supported machines.
 - Deploy on unsupported machine first, then move to production.
 - Work with ISD directly on determining requirements for moving from unsupported to supported.
 - Contact person is cfb@usc.edu.
- ISD has many security considerations. Review these and design your system accordingly. You may have to explicitly show that your system is compliant.

Transitioning Within ISD: Production Machines (cont.)

- Do not expect that ISD will be willing to install and support any COTS product.
- Do not expect that ISD will write or modify your code.
- If you design your system to utilize as much of existing ISD supported software, configuration, administration, etc. It is more likely to be accepted and supported.
 - Find out what they use and how they typically use it.
 - If your program runs in `unsupported.usc.edu` and conforms to the standard configuration there, then it will likely run on any production machine.
- Beware of hidden costs due to ISD support. Your customer will be responsible for any overhead to ISD stemming from support of your system.