PROFESSOR: One example will be what happens when the battery runs up on the computer and what bubbles up when it's running. And there's certain computers that might [UNINTELLIGIBLE] hard drives. Some computers just keep it in RAM when you restart it. There's no big difference. What happens when an IM window pops up in the middle of the game and leaves the control screen? And all the keys start [UNINTELLIGIBLE] IM window. Ideally, they give you pop ups that doesn't work. But catching those is one of our favorite things that we do when we're making digital games here in the lab is like slamming all of our fingers on the keyboard at once and see what happens.

And we're kind of looking for those things. We're also looking for subtler things like is the amount of RAM that those games are taking up in the computer slowly going up all the time? And of course, comparing it against a spec grade. So a character is supposed to be able to create a certain file hold style or something when you enter this key sequence, and 99 times out of 100 that works and that 1 out of 100 doesn't and we need to figure out why. That's our feat.

So in a lot of those situations, what happens is that output comes back in the form of other parts and-- how many of you actually do programming? Oh, OK. How many of you have found a book report? OK, not that many. There are a couple of best practices where you follow a book report and this one translates a little bit into when you're really working in a team in a game. How do you communicate that something is wrong to someone else who is also responsible for the game, but may not necessarily have seen the problem?

First of all, you describe what you were expecting to happen and what you were doing. I was expecting to go right when I push the right button. And then you write down what happened. You try to get as much detail as possible for the person to re-release this problem.

Now, when it comes to war games, in many situations, the kind of detail that you need to provide largely comes up against it. So 'a' had this much money, 'b' had that much money, they try to do this game mechanic and no one knew what to do because the way the rules are

written there's only one way to interpret it. And that gives your team enough information to be able to start acting on it. Because you are bargaining and your rules are kind of right out there in the open, there's usually less of a mystery to see where the problem occurs. But you know there's problem and it could end and you largely see where it lies. You may not know what to do to correct it, and that's actually where beta testing comes in. So but I'll get to the testing shortly.

But technical testing, a lot of it again is comparing a given spec. Sometimes that spec is created from your team. This is what we want our game to do. This is what we expected to happen when we made this game. We expect that mid in the game, a certain game mechanic will start to take priority over other mechanics. Early on, everyone's buying property is a monopoly. Later on, everyone's going to be building houses and then even at the end, people start to mortgage stuff. That's what we kind of expected out of it. And then you play the game and you start realizing you mortgaging something way too early here, something's out of whack here. So that's the kind of technical testing you do in games. This is what we expect, why isn't it happening?

The other side over here is kind of like being the defender of the player, being the player's advocate. How, can you be the person on the team that is actively looking for things in your game that are going to hurt your players and then mentioning it and saying, maybe we should do something about that. It's as simple as, here's the information that we want our players to have at a certain time. Here's the game state. Your characters have a certain state. They are this close to winning, but are this far away from winning to have these results here. Do the players understand that? Have you provided them with enough information to be able to make a good decision? If they know the game state, do they know what options they have in front of them?

So as we play more and more board games this semester, you can actually look at the way how rules are presented redundantly. Sometimes they're written up in the rule sheet, but and of course you've got your index card-sized thing which goes through step one, two, three, four, these are the four things you've got to remember every single turn. You've got to get it, and everybody has a copy right in front of you because somebody on the team decided that that was a good idea.

How did they come to that conclusion? Occasionally, they are doing it because they see there's a problem right away. There's too much information, and you just don't expect any

player is ever going to be able to grasp all this information. So that's come out with a couple of ways to simplify it. Sometimes that involves taking up the mechanics, making something that was written up a text graphic report and some icons, reducing all the numbers by two so the math becomes easier.

But sometimes, you do it by usability testing, and that's one kind of testing that often comes together with beta testing. I've invited Sara Verrilli, our development director, to come in here and talk about the kind of pre-testing that you can do, which is illustrated by this picture. And the book goes through several different kinds of environments where you do pre-testing that's sitting one-on-one with people and doing site testing for the group.

Because this style is all about board games and card games, for the most part, we're designing games for three or four people to play at a time. The chances are, almost every single test that you're going to do [UNINTELLIGIBLE] at a single time. That's just the environment you're in that's meant to be played. That's why you end up testing it. Obviously, a single-play computer game, that's not necessarily the environment you're testing in.

Actually, before I transition over, there is one thing I want to mention about technical testing. Sometimes the specs you're testing against are not the ones that came up that would necessarily come up with phi over t. If you're doing a computer game-- say for an Xbox or a PS3 or a Wii. There are these things called technical [? evaluation ?] requirements, which are basically huge lists of things that your game must do before Microsoft will sign your title or allow it to be released under that title.

PC obviously doesn't have this, but if you release it on speed or something, then chances are, you'll have requirements. It could be something as simple as generally, it doesn't crash-- and games do crash, even on PS3 and Xbox-- you get things like what happens when you're playing a game on the PS3 or Xbox-- I guess more the PS2 and the first Xbox-- and someone yanks out the cable? All you're thinking is that you're-- and then the keyboard comes out, what happens? Does it continue?

STUDENT: It buzzes.

PROFESSOR: It goes into pause, right? And usually it tells you, your controller is disconnected. And that's a requirement. Someone had to write that code specifically to detect that situation and then throughout the suite across the game. And the reason everybody had to do that was because Microsoft and Sony and Nintendo-- well, not so much Nintendo nowadays because Nintendo is

wireless-- but these are the kinds of requirements that they expect. If you lose your connection to Xbox live halfway through the game then something's going to happen.

And so again, for that kind of technical testing, you're sort of comparing it against spec, but spec may not necessarily be something that you're keeping up with. It may be something that the publisher or your platform developer, platform manufacturer mandated.

How does it affect the board games is that sometimes, board games are actually going to require half the requirements to that. If you're responsible for the design of the box, for instance, there are things they're going to require on the outside of the box, like what ages are the game applicable for, how long is it meant to be played, how many people are going to be playing your game. I think Days of Wonder, which is a publisher, actually requires a character to be on the box.

Even if your game has no characters, there must be a human-like presence on the box or your players are on the box because they think that it increases sales, and they probably have the numbers to back it up. That's why if you look at things like Ticket for Ride, for instance, has people. If you play Ticket to Ride, it has no people in it, but it has people on the box because they want people to think that's what users are.

So that's just an example of things that a board game publisher may require you to do. Very common that you'll have to do translations in a range of different translations for the different countries it's going to be released in. It may not be your job to do it, the publishers may end up being the ones helping you with the translation, getting all the publishing data formatted decently, but someone along the stage is making sure that happens before the game hits the shelf.

OK, that's technical testing. That's a little bit about becoming a player and so let's talk about data center a bit.

SARA VERRILLI: Let me bump you a little here, if you don't care.

PROFESSOR: Oh, right.

SARA VERRILLI: OK. Let's see. Let's actually go ahead. So focus testing. This is a very similar track for the summer program folks. I have about 10 minutes to go back and relook at my notes and relook at the lecture, so if it's a little stumbly in parts, it's because I haven't given it for a while and

haven't thought about it. It's much more geared, the original talk is much more geared towards computer testing, and testing specifically single-player computer games than it's testing board games, which are almost a multi-user.

So first of all, focus testing is actually dissimilar. Focus testing is only a specific case of user testing, which is to say, you're testing your game using actual users, not your development team, not people who play board games 12 hours a day because that's what they love to do all the time, but more the sort of average person who might want to pick up your game off the shelf, go buy it and then go play it with their family and then go out to dinner at Chuck E. Cheese or something.

Another type of user tester is usability testing, which is fairly similar to focus testing in as much as you're still using users for it. Focus testing is usually much more geared towards are people enjoying your game? Do people want to play your game? Are people going through the sort of play experiences you want them to have as they play their game? Because usually when you create a game, you're thinking, oh, like for Settlers of Catan, you expect people to spend a lot of time talking with each other trading things back and forth.

When you're making Diplomacy, you expect people to spend a lot of time arguing with each other over well, if you invade that country, I'll invade that country and we'll gang up on him and we'll all come out of the game and then we'll just divide four between the two of us and win while you're making secret underhanded arrangements with the other guys, but OK, so he's going to attack you on this ground, we both know that, that sort of thing, as opposed to, say, Transamerica which is just more putting pieces down and it's pretty much independent play in many ways.

Usability testing is much more about how you easy is it for people to play your game. Do they understand your rules? Are the mechanics clear? When they sit down and start sorting out the tokens, can they tell which tokens are the player pieces and which cards are which and how you use those? Do the icons on your game make sense to people? Do they understand them? Things like that. It's not really about whether or not people like your game or are enjoying your game. It's about can people understand your game and start playing it.

So here we go. One big problem we run into when people sit down and we have people playing your game and giving lots of comments, this was really fun or this really sucked, or I hate this part where I have to spend time talking to the other players, I want to be able to stare

at just my cards and board completely without interacting with anyone else. Focus testers will give you lots and lots of feedback, some of it is useful, some of it is not.

And you need to remember that focus testers are not actually people building the game. Focus testers are the people who you are trying to get information about your game from and one problem that we often see in experienced game developers and people making games doing is, oh, the focus tester said we have to make the whole game red, quick, let's make everything red.

So and that's usually, that's not a very good solution, right? Who knows what the problem with this actually was, but what you need to do is find out why did the players want to make everything red? Is it because they don't like the artistic design of the game? Is it because someone in the game is colorblind? Someone playing the game currently is colorblind, is having a hard time telling things apart so they'd like everything to be the same color so everyone's having the same problem they are. I mean, who knows? It's hard to tell what people are thinking. When they give you advice, and when they give you these comments, you have to have some way to filter those comments, some way to think about them and some way to decide what actually is the right reaction to take in view of what you want your game to do and how you want people to play the game.

Since you're the game designers and developers, you have to make those decisions. Focus testers can only give you data. You shouldn't ask them to design your game for you. Maybe you actually have good ideas that you want to use, but you want to make sure that they're good ideas that you do want to use.

So you may notice, these are both my lectures, and these are pictures that Mike [? Dropho ?] drew for a poster, and we love him so much. So in the ideal focus testing situation, you're really trying to get as much data as you can about your player playing the game without spoiling the information. What you'd really like to recreate in a lot of ways is, so you just bought the game and have gotten home, what happens? And that's what this picture is all about, there's somebody luring somebody to play the game without any information.

So things to think about when you're planning your focus testing is, what are you testing? What questions do you want to ask? What information do you want to get out of it? How are you going to test it? How many players do you need? What kind of interactions do you want to see? On the backhand, who are you testing with? What kind of people are going to make your

game? Are you making a game for children age 9 to 15? Are you making it for college-age students? Are you making a game that you think that older players will want to play?

Because if you're making a game that you think is going to be popular with the senior citizens market, testing it with a bunch of elementary school kids probably isn't going to go very will. Trivial Pursuit Classic Edition, a game that came out about 20-odd years ago, I suspect that if you sat down a group of high school students with it and had them play it, about three quarters of the questions they'd just stare at you and be like, I've never heard of that. So, that's certainly where the audience comes in.

So, the what, planning out the questions that you think you're going to get information for. When you know what information for focus testing, it's a lot easier to know what to keep track of, how to take notes, what reactions you're most interested in as people are playing the game. So, what questions are you going to ask? What data it takes to answer your question? And how are you going to get that data? Are you going to go all through observation taking notes? Are you going to just step in and out of questions? So why did you move that character from there, just out of curiosity I'm wondering why you made that move. Will you tell me about it? Or, you can ask them to fill out a questionnaire at the end of it.

Depending on what data you want, different methods will work best. In general, usually the best way to do it is observation. As soon as you ask someone questions, you sort of perturb the system and when you ask them to fill out a survey, it's after they've done it. They've kind of forgotten what they were doing. They may not exactly remember, and to be perfectly honest, people would really rather play your game than fill out a form. So they'll be more engaged more interested and more interactive when they're playing a game and that's when you're going to get your best data.

But there are times when you want to ask someone something sort of analytical and you don't want someone to be thinking analytically while they're playing the game in a lot of ways , and so there are times when that reflection comes out so much better and it's better to have that written down and get it in your own words.

STUDENT: You would hope it plays better than a survey.

SARA VERRILLI: If your game is better than a survey, you probably already observed a problem right there and you're probably glad you found it.

So how? In order to get all that data, you're going to have to actually interact with the user, someone who's never seen your game before, someone who doesn't actually care whether or not you get good information out of playing this game. They came here to sit down and play the game, and while they would like to help you by giving you information, that's not their primary goal in a lot of ways. Usually, their primary goal is to sit down, and play the game, and have a good time.

So you need to be able to structure the interaction with them so that getting the data to you is as easy for them as possible. So we need to think about what this player needs before he or she sits down to play. Usually, in a board game or a card game, that's the set of rules. In a computer, it's always trickier. Usually when we're bringing in people to test the first prototypes, we have a game that works, but we don't have any instructions in game. If we're going to have a tutorial or anything like that, that's usually one of the last things that goes in.

So when you're testing computer games early in the process, usually you end up sitting down and writing out, well, this is what the tutorial would look like if we had it, so read this and then sit down and play it. With board games and card games, it's actually a little bit easier, because you can say, a ha, this is the rules that are going to package with the games. Give you the rules, step back and see what happens. So, it's actually an easier aspect for testing board games.

And finally, treat the players politely. When they find a misspelling in your rules or when they discover that you've left out six of the key cards to play the game, oh, yeah, you're right, thank you for pointing that out, I wouldn't have noticed that if you hadn't shown it to me. Whether you would have or not doesn't really matter. They're doing you a favor by playing your game that isn't quite working well, so you want them to know that their contribution is invaluable and you appreciate the time and effort they've put into it.

A few things, the more they talk about your game and the mechanics they find that break or that do or don't work, that's all data right there, That's all great. Another great advantage of testing board games is you usually get groups of players, like two or three or four players, and when you have multiple players, they'll talk to each other about it. Oh, I noticed this. Did you notice this strategy? Yeah, I did, but I noticed that if you add that strategy that you can do this thing and this other thing and that's all really neat.

When you're testing single-player computer games like we usually do in the summer program,

you've got a single user that is sitting there between them and the computer. You have to watch and figure out what they're thinking. When you have a group, it's not usually that hard to get them talking about what they're doing and what they're thinking, and so that's a really good way to try and get more data.

So yeah, here's sort of the end summary. Have a specific question and questions in mind. Know what you're doing in focus testing, know why you're doing usability testing. If you don't know what you're trying to test for, then your data is going to be a lot harder to figure out what to do with. So you always want to we have that question in mind. That's actually a really big one.

What data you need to answer that question so you can collect it and know how to collect it, and make sure that you can give the information to the players, a standardized set of information to the players and step away so you're not sort of coloring the information and giving them the things you're talking about by, oh I forgot to mention, oh and did I say, so you've actually got sort of a standardized standard size play group scenario set down. OK, I think that's actually what [? I came for. ?]

PROFESSOR: So a couple of words about when to be testing, which is really as soon as possible. Usually, before you think it's possible, you should be already starting to plan your fist test because at that point, you're probably going for the lowest-hanging fruit, which will be probably somebody else in this class because they're going to give you feedback anyways, you can use that favor in exchange. The whole point of making testing as simple as possible is you want to be able to get information while you still have time to fix it, to do something with that data.

What has happened a lot of times in this class is we have a project team where they come up with a game requires say, generating so many cards or that they're going to spend most of the project deadline actually creating those cards. They'll test it, but they'll test it on themselves because the team already gets it and they're not looking for other people. They have a plan to ask their friends to come in for a specific time, they've set aside an hour or something like that. So what happens is that the last week comes around and they start playing it with people who haven't seen the game before and then discovered problems. Unfortunately, the game has 100 cards that all now need to be fixed. This is a problem because you just discovered that problem too late.

A couple of ways to handle that, though, one, don't decide on a game where making changes

across the entire game is going to take you more time than to test it because that's going to make it impossible to fix anything. But the other one is just testing early. As soon as you've got a couple of key mechanics down, sure you are planning to play this game with 100 cards, start with 20. Start with a small of group of cards, hence your core can see people and understand that they're going to discover things in the production of those 20 cards and in the testing of those 20 cards that maybe you don't want to be doing for the next test and you can redo those 20 cards pretty quickly the way that you can't redo 100 cards. Just an example. I'm really not encouraging people to do a 100-card game, by the way. A regular poker deck, 52 cards, that's all you need for a really interesting game.

Let's see. Some other things. Say you're doing a test, but it's not with your entire team present. Say it's just you and your teammates are all living in different dorms and you decided to play with those in your dorm. That's great, because it's a cool game and you've got your feedback, how do you communicate that back to your team? I talked a little bit about writing up a little report, but here's a couple of other tips because one thing you have to remember is that chances are, every single idea in the game, somebody in your team is probably really passionate about it and what you're basically trying to do is try to convince someone in your team that maybe this idea should be changed. And your game will probably be better for it, but there are some tactful was of telling that.

So let's see. Keep it short, this also applies to book reports. I talked about you giving people enough information to reproduce what the problem was so that they can understand the state of the game and why the problem is occurring. And the flip side of that is not too much information, not more information than is necessary to keep it short. Little [? farm ?] codes, give them something that's easy for people to scan. Don't make them read and essay for them to figure out what comments will be solved.

Now actually, one thing to keep in mind and if you keep in mind and think about what Sara said earlier about the data that you decide to collect, one problem with board games is with all the interesting conversations happening around the table and multiple players suggesting strategies to each other, you can come up with reams of notes in a single play test session. How do you decide to digest that is really important. And you're the only person there and you're the only person on your team, you need to try to let the rest of your team know. You need to put in the work of digesting all of that information before necessarily giving it over to your team.

SARA VERRILLI: One possible solution also is you can get permission from play testers for a tape recorder. Just get a digital recorder and record all of the conversation as it goes on.

PROFESSOR: But you still want to pick out the keys, you don't want to revisit the entire conversation. Because if the game took an hour, and every single one of your team members visits, by now you've lost three hours. So you want to find that, do that little bit of work to save yourself the detail.

But that's the way because you can focus, you can take down notes on the main points and just let the tape recorder record all the details so that you don't have to worry about recording and really, the main points are the things that you want to make sure you get across to your team. The tape recorder is just evidence. You could use video depending on who you're recording with. Sometimes people get kind of antsy when they know there's a camera. That being said--

SARA VERRILLI: And also, just put the video tilted at the board so you're not recording the players, you're recording the game.

PROFESSOR: Actually webcams are really good for that sort of thing.

SARA VERRILLI: Yeah.

PROFESSOR: You don't run out of tape, just record straight to hard drive.

SARA VERRILLI: Easy to fast forward through, also.

PROFESSOR: Yeah, with your editing tools.

Let's see. So when you're reporting to other people, think like bullet points for instance. Huge long paragraphs of prose, not so much. Let's see. Try to avoid using the following words, broken, problem, unusable, confusing, kind of odd, you can get more specific than that, but these are words that kind of like imply not only is there a problem, someone made a mistake in his design of the game. And truth be told, most of the time it wasn't a mistake, you just didn't have the complete play testing, you probably just had the designer.

But you don't want to make someone feel like they've made a mistake because that makes them defensive and then they're going to try to defend how what they wrote in the first place is actually correct, which is actually not correct either. You can talk about features that need

attention. With attention, this feature could be confused. I guess that's actually on the edge there.

So but just generally, pick your words carefully and one thing that Sara mentioned earlier was that testers are not designers and key designers are actually not a good testing team. So for the most part, you are immediately going to exclude yourself from your target audience because you know too much of the game. This is also covered in your book. As soon as you've played through your game once or twice, that's it. Your kind of work is finished. You can still be a good technical tester, I guess. You can still check whether some mechanics are working the way that you want to work, but you can't tell whether the dynamics in your settings are working anymore. You have to get your focus testing done.

SARA VERRILLI: You also really can't tell, you can never tell-- you can never share your usability tester. It will always make sense to you. It's amazing that what makes sense to you is completely incomprehensible to the very intelligent person you give the game to. Really, truly, I've been thinking about this one myself.

PROFESSOR: Usability testing, by the way, is one of those things that even if you only have the chance to do a pop-up partial test like, all I have is this one card, the game's not complete, but grab someone walking down the corridor on the way to the washroom and say, come in here and take a look at this thing I'm working on. Even that is better information.

STUDENT: Hey kid, come in here. Come in here, I have something to show you? I have a surprise for you. [LAUGHTER]

SARA VERRILLI: Actually, you can get going testing for [? serious. ?]

STUDENT: Free candy.

PROFESSOR: [? Mona ?] actually calls it employee kidnapping. [LAUGHTER] Of course, he's the creative director, so he can do that.

SARA VERRILLI: It works best if you're accosting people whom you have power over.

PROFESSOR: The audio thing, by the way, is one of those things that could be really useful if people are difficult [? pundits ?]. And so it might be one of those things that you want to keep in reserve, it's like, OK, there is a problem here and the team goes, I don't think there's a problem here. I don't understand why anyone's having problems here. Then you quit, because it's kind of a

bludgeon actually on the team.

On one hand, if you have a good agent team, they're going hear it and go, oh my god, we have a problem and immediately try to fix it. But it's awesome. On the other hand, it's like, well, do you need to do a recording for every single offer that we have? That's something to be known about, so keep that in reserve. I think that's pretty much it. I have an exercise, but I think [UNINTELLIGIBLE] so I can turn that off. Do you have any questions while I graph the data segments? No? OK.