**OLIVIER DE WECK:** OK so let's start on the material. We're following the V-model and the empty boxes are getting fewer and fewer every week here. So we're on the right side moving up toward lifecycle management and operations. And today's topic is via V&V, verification validation. And I also want to discuss the FRR as one of the key milestones, and that's the flight readiness review. If you're not building something that flies, this is your launch to market, right launch to market review. Are you ready to launch your project or product to market?

The outline is-- we have quite a few things to cover. So first of all, I want to drill into verification and validation. What's the difference? What is their role? What's their position in the lifecycle? Then we'll spend quite a bit of time on the issue of testing. What kind of testing is done? Why is it done? We'll talk about aircraft testing, flight testing, we'll talk about spacecraft testing, but also some caveats. Testing is not always the-- it's not free of challenges and difficulties.

Then I want to talk about technical risk management, which is often covered in classes on project management, but I think it's essential here as well as a system engineer to have a good grasp on technical risk management. So I'll cover the risk matrix, the Iron Triangle, cost, schedule, scope, and risk, and then I added a small section on system safety, which is a very important topic as well. And we'll finish up with discussing the FRR, the flight readiness review.

So the readings related to this lecture are sections 5.3 and 5.4 in the handbook, the *System Engineering Handbook*, and then there's a couple of appendices there, Appendix E And I that are very, very helpful. And at least one of these I'll mention in the lecture. Plus one of the papers, and this is a paper about a decade old-- and I know a lot of work has been done on this since then, but this is a paper by Professor Leveson. Nancy Leveson was a colleague of mine here at MIT who's really an expert in systems safety. So the system safety model that she's developed is the subject of that paper.

OK, so let's talk about V&V, verification and validation, and how they fit together. You've seen this diagram before, but I just want to talk through it in some detail again. So the idea is you start your project, you're undertaking, in the upper left. You do your stakeholder analysis. Who are the stakeholders, the customers, beneficiaries, the regulators, the suppliers, the partners on the project? So you really have to do a good job doing your stakeholder analysis then in order to write your requirements.

And I do have to say I've been very pleased, especially with your assignment A-2. You really dug into those 47 requirements for canned set. You grouped them, you scrubbed them, you did a great job. And the idea is that for each of these requirements, you also have target values. There are certain thresholds or target values that have to be achieved. And then you actually do the development. You do the conceptual design, the detailed design, and that's written here as functional deployment. In other words, especially for the functional and the performance requirements, how will you actually implement those, embody those, in technology, hardware, software, and so forth. So that's your intended function, your concept, and then you're implemented design solution.

Now the question is, do you actually satisfy A, the requirements, and do you satisfy your stakeholders? And it is possible that you satisfy the requirements, but not the stakeholders. So the way to think about this is we're going to close the loop. In fact, we're going to close two loops. An inner loop-- and I put testing here. Testing is really one of the ways to verify whether you meet requirements. There's other ways too, but testing is often the most important. So we close this inner loop. So what we ask is-- we test our implemented solution, our implemented design, and ask the question, did we deliver? Did we actually satisfy the requirements as they were written? Do we satisfy the requirements as they were written? And are these attainable? Were these requirements attainable?

So this loop here, this inner loop, is the verification loop. You verify whether you design as implemented satisfies the requirements as written. That's what verification is. And then there's an outer loop where you take your implemented design solution, and you essentially bring it all the way back to the stakeholders. And usually that also means you're employing it in a realistic environment, like in the environment that the stakeholders will actually use the system, not in a pristine lab environment. And you have the stakeholders try out your system and see whether they're satisfied, whether this meets their original intent.

You remember the CONOPS, concept of operations? Can you actually do the CONOPS the way you had envisioned it in a realistic environment? And that's what we call validation. And that's the outer loop. You see the difference? So a lot of people who don't know system engineering who have never been exposed to this when they hear verification and validation, they think it's basically two different words for the same thing. It is it is different, it's not the same thing. And then if you successfully verify and validate, you end the SE process and you deliver, which is good.

So this is something I pulled out from the handbook, which is the differences between verification and validation. And I'm just summarizing this here. So one way to ask is, was the end product realized right? Meaning, did you do the right thing? Or did you implement it correctly? So verification is often done during development, so you verify components, subsystems, you check if the requirements are met. Typically, verification is done in a laboratory environment, or on a test stand, or some environment that allows you to very carefully control the test conditions. And verification tends to be component and subsystem centric.

OK, and then validation is the question was the right end product realized? Did you actually build the right thing? Did you deploy the right solution? This is often done-- so validation focuses more on during or after system integration. It's typically done in a real or simulated mission environment. You check if your stakeholder intent is met. And it's often done using the full-up system. It's difficult to do validation on a subsystem basis alone. Typically, validation implies you've got to use the whole system to do it. Or, you basically use dummy subsystems. You basically replace the actual subsystems you're going to have with something temporary so that you can go back to the stakeholder and give them as close to the real experience as they'll have with the actual system.

OK, so that's essentially the distinction here. So I want to do a quick concept question on this to see whether this point, this distinction, came across. So here's a link, SE9VV, these are all caps. And what I'm listing here is different test activities or different type of activities. And I'd like you to check the box here whether you think this is verification, whether this is validation, or you're not sure.

All right, testing and handling of a new car in snow conditions in Alaska. 90% of you said this is validation, and I would agree with this. So many car companies, I think all car companies, once the vehicle has been finished essentially, the design, it doesn't go to market right away. There's a very extensive-- usually it's at least six months of field testing of a new vehicle. And you go to the desert where it's very sandy and hot, test your air conditioning systems right at the limit. And then you go to really cold climates. In Europe they go up to Sweden and Norway.

And here we tend to go up to Michigan, Minnesota. And so the idea is that you really utilize the vehicle in a extreme environment, but that's realistic of actual operation. So I agree with this.

Frontal crash test in the lab. Most of you said it's verification, and I would agree with. So those very standardized crash tests that you've seen in some of the commercials. The vehicle is prepared, instrumented, you have the crash test dummies. And then you have, typically, there's at least three different kinds of crash tests. There's frontal, there's side impact-- like the T crash-- and then there's a rollover tests, that are more and more standardized. And the test conditions in these are there they're highly stylized tests. They're very prescribed exactly the speeds, the angles, everything is prescribed. And real accidents, the variability of conditions is much, much bigger than in these tests. So I agree because the test conditions are so tightly defined and constrained, this is verification, not validation.

Testing of a new toy in a kindergarten. OK, so here you're-- and this is the toy companies, and they essentially-- before, again, they make a big million or billion dollar decision to mass manufacture toys, they will actually have kids play with them in realistic environments. And so I agree with that this is validation. Vehicle emission testing. Obviously, this was the big Volkswagen scandal that we had. So basically, what this cheating that happened is essentially software that was embedded in the vehicle, such that when the vehicle experienced exactly the test conditions of these drive cycles that are very well known, very well defined, the vehicle would internally switch or reconfigure to a verification mode and really emphasize low emissions at the expense of fuel economy. And as soon as the vehicle would detect that it's in a more general driving conditions, it would essentially switch that mode off.

So verification, again, is on a dynamo in the lab. Satellite vibration testing on a shake table. We'll talk about this. This is often-- we refer to this as shake and bake in the spacecraft business. The spectra, the load spectra, that are put into these shaking tables are, again, very stylized and different for each launch vehicle. So here we can debate a little bit whether-- is it closer to validation, because the actual test conditions are so much adapted to each launch vehicle. But I agree, this is primarily a verification activity.

And then the field testing of the Google glasses. So you basically produce an initial batch of your product and then you give it to like lead users. You have them try it and give you feedback. This is much closer to validation. So I think by and large your answers here are very good. And the real distinguishing factor is whether this activity happens in a lab, in a very controlled environment, under stylized conditions, or whether you're actually going out in the field, in a realistic mission environment with real users or real potential users that are not especially knowledgeable or not specially trained about the system. So, very good job. I think most of you really understand that distinction.

OK. So let me talk briefly about the-- yes, please, Veronica?

**AUDIENCE:** Got it today. OK, are there any has that really bridged the gap that really could be seen as both. So I'm thinking about products in particular that are to be used in a lab setting where you have a very specific kind of user, where meeting the requirements is more about how the tool is employed, and I see the user in that sense is part of the system. So I'm wondering if in a more clinical sense there's an action that is both validation and verification.

**OLIVIER DE WECK:** Yeah, and so you said clinical. So I think there are situations where the distinction is not as sharp, not as clear. You said clinical, so I think in hospital, you know for medical equipment, like if you think about surgical equipment and things like this, where it's very hard to really-- it's very hard to do verification in a stylized way. The only way to really check it is to have the equipment embedded and used in a pilot study, for example, in a hospital. And in that case, because the human is so involved, and it's not a general consumer product, but it's really a tool for specialists, the only way to really check your requirements is to actually embed it in a realistic environment to begin with.

So whenever it's very difficult to design, very specific, isolated tests where you can check for each of these requirements one by one, you almost have to move straight into validation. And I think in medical equipment that's often the case. Yeah, go ahead.

**AUDIENCE:** Would you say that for a spacecraft really true validation isn't possible? You have to recreate the conditions in some way, in some kind of a laboratory.

**OLIVIER DE WECK:** I think it depends on the novelty of the spacecraft. If you're launching something like a standard communications satellite where you've launched dozens before, and you know the actual pitfalls and the operating conditions, you've experienced failure modes in the past and eradicated most of them, I do think that you can do a lot in verification. But then I'll show you one example of a spacecraft we've actually talked about before, where there's going to be a lot of residual risk. And the first time it's deployed, people are going to sweat, because there's still a lot of unknowns to be resolved.

OK, so let's look at the product verification process in particular. This is from the-- so this is the product verification process from the NASA System Engineering Handbook. So what are the inputs? The end product to be verified. So you have to have the artifact that you're going to verify. The specified requirements baseline, you need this as a reference. What are you going to check against? The product verification plan which is essentially your test plan. What test cases are you going to run? How long are you going to run them? How many repetitions will you do?

And then product verification enabling products, which would be test software, test equipment, things that are not part of the product itself, but our enabling of the verification process. You then, essentially, go through this process, and what are the outputs? The verified end product, the product verification results, so these would be test protocols, things like this, product verification report, and then product verification, any other work products that come out of it. So you could have, for example, discrepancy reports. You failed some tests. Well, that would be an important output. And then the question is, is this significant enough that you have to go redesign or retest? Or is it a minor issue that you can waive essentially to move to the next stage?

So let me just give you a quick example here from my own experience about this verified end product. One of the things on the Swiss F18 program that we did is not just by airplanes and equipment, but also models of the plane itself. In particular, finite element models, very detailed finite element models of the structure. And these models were very expensive. Like, some of these models were millions of dollars. And so I got a phone call, I was a liaison engineer at the time in St. Louis. I got a phone call from Switzerland saying, this is crazy. How can we be charged millions of dollars for this particular set of models?

And I said, yeah, that seems pretty expensive, so I'm going to go negotiate this. And so I started negotiating, and I guess either I'm a bad negotiator or it was really, really clear why these were so expensive. The reason these models were so expensive because they were on the right side not on the left side. So every one of these models that we were purchasing had been verified using actual physical tests. So every location was guaranteed under the load conditions to produce a stress and strain prediction at that location that was guaranteed to be correct within plus or minus 5%.

So the model had been very carefully calibrated and tuned against physical reality as opposed to a finite element model that's just anybody can make a model and put some load cases and boundary conditions on. And you don't know how closely does this mean. So there's a huge difference in value between a product, a model that has gone through verification where at the end of it there's actually a report, there's a protocol, there are data that says all these features, all these requirements that you had against it have actually been checked. This is a certified product. And that's the main reason for the price, because the actual process of verification is very, very resource intensive.

So even though when you look at it physically, you might say, I can't tell the difference between pre-verification and post-verification because physically it's the same. But in actuality, there's a huge difference because once it's been verified and certified against a set of requirements, it's a much more valuable asset. Does that make sense? So keep that in mind when you think about these products on the right side.

Now what are the types of verification? So tests we'll talk about, so you're physically testing. But there's other ways to do it through analysis, through demonstration, and through inspection. So analysis essentially means you're doing a calculation with a mathematical calculation or a simulation that satisfies you that this requirement is met. And you're doing this with the input parameters into the simulation are as accurate as possible based on the physical reality of the system you have. But for whatever reason, either because you don't have the funds for it, or you can't simulate the operating conditions well enough, you have to do it through analysis.

Demonstration essentially means you're operating the system, you're demonstrating the functions that you need, but you don't necessarily have a lot of instrumentation on the system. And you don't certainly do destructive testing. In other words, a demonstration simply means you're operating the system as intended and demonstrating physically that it performs its purpose. Inspection essentially means you are physically inspecting the artifact either visually inspecting-- there's also a lot of techniques called NDI, nondestructive inspection through with X-rays or eddy current sensors. You're checking for the lack of manufacturing flaws or [INAUDIBLE], whatever it is. But inspection essentially is you're not physically operating the system, but you're inspecting the artifact to make sure that it satisfies a certain set of requirements.

And then testing typically means that you're putting a stimulus into the system. You're operating the system under some test conditions. You're recording data, which you then analyze in terms of comparing that to your prediction or expected behaviors. So these are analysis, demonstration, inspection, and tests. They are all different ways of verification.

Yeah?

**AUDIENCE:** How do I know when I'm supposed to use more than one type at the same time? I mean in and or or.

**OLIVIER DE WECK:** Yeah, that's a good point. There's no real general rule of this, but in general, I would say the more crucial, the more critical a particular requirement is to the operation of the system, the more intense the verification will be. Whether that's just using one of these types, you know you just run more tests or more different tests, or you doing a combination of inspection and testing. There's no there's no general rule in terms of two out of three, or two out of four, but the purpose of the V&V plan, the verification and validation plan is-- and you did a little bit of this in A2. You did a little bit of thinking into how would we actually verify this requirement.

The purpose of a V&V plan is to say for each requirement which of these four methods are we going to use for verification, and then actually write down each test that you're going to perform, what kind of equipment you'll use, what kind of test conditions. It's a lot of work. In fact, I think it's fair to say that the people that do this kind of work, verification and validation, are typically different people than the people that do the writing of the requirements or that do the actual design work. This is a pretty specialized activity and the people are a little different.

If you've met people who would do testing or quality inspection, they're quite different. It's a different mind set up. Go ahead.

**AUDIENCE:** According to what's happening in ESA, actually this ADIT will be imposed in the specification prior to your proposal. And it will give you to a minimum requirements to just test against. And they will give you a rough matrix for every requirement line, whether it's [? ABIT, ?] and then you have to answer with a validation intense plan, usually, unless your agency and you are defining the specification and you have to do it, and it's mostly based on experience. And it's some people that have really lots of knowledge that then make these specifications. But I think for all of you engineers here in the next 10 years, you'll be just hoping that not so many of these ADITS in the specs you will get, because you will have to answer as part of the specification actually.

**OLIVIER DE WECK:** Yeah, and what--

**AUDIENCE:** I'll just demonstrate it.

**OLIVIER DE WECK:** Well, and the point you're making, Voelcker, is that this is a contractual requirement. This is not optional.

**AUDIENCE:** [INAUDIBLE]

**OLIVIER DE WECK:** Yeah.

**AUDIENCE:** It's not optional and it has to be followed, the pricing, right in front of [INAUDIBLE]

**OLIVIER DE WECK:** Yes, very good points. So the outputs of all of this are discrepancy reports, if there's any discrepancy reports, waivers, the verified product itself, and then the compliance documentation, which is essentially your test protocols, et cetera, et cetera. And as you can imagine now if you had 47 requirements with can set, and then by the end-- [? Uonna ?], what would you say the number of requirements that we ended up with at the end in A2 were closer to 100, right? Most people were around 80, 90. OK, now imagine-- the good thing is if you can do tests that actually check multiple requirements at once, that's a good thing.

If you can do tests that help you verify multiple requirements through the same tests, you can save some money. But the whole testing strategy, the contractual requirements that Volcker was mentioning, it's a big, big, big deal. It's a really critical part of system engineering. OK. So in terms of the lifecycle phases where this fits in, most of this activity happens during phase D.

So you remember this was the NASA lifecycle model, and so phase D is system assembly, integration, test, and launch. So much of the testing that we talked about happens during phase D. So the system has been fully designed, it's been assembled, it's been integrated the way we talked about last week, and now you're really putting this system through its paces. And so this phase D is intense, it's expensive. And if something goes wrong, it sends you back to the drawing board often. And you have to do-- you have to figure out whether a failed test, a failed verification, is a showstopper. If it is, then you have to redesign the system, you have to retest.

But if it's a minor thing, then you might be able to request a waiver and you can say, OK, we didn't achieve this requirement, or we failed this test, but we think it's a minor issue. And instead of holding up the program, we're going to get a waiver for it, which means you get an exemption essentially, and you can move on. And whether or not a waiver can or cannot be granted is a big deal and that goes under risk management, which we'll talk about in a few minutes. Yes?

**AUDIENCE:** I had a question on-- so if you're a system integrator, and you created statements of work for other people to procure a large optic, or something, they go and build that [INAUDIBLE] requirements, and then they do all the verification and testing, and then they provide all that documentation. And then like from my experience with anything like procurement out stuff, that comes back in your in house, and you go to assemble it, and you essentially do a lot of that verification and testing again to double check that supplier. There is a little bit of a conflict of interest, obviously, with that supplier doing the work and also verifying their own work. Is there any good way to get around that? So it seems like a very expensive process.

**OLIVIER DE WECK:** So my experience-- basically, you're talking about separation of powers. The good suppliers, they will have internally separation of powers. In other words, the people that are doing the testing and the Q&A, they're usually people who really enjoy finding mistakes and faults. And that's why when I'm saying-- I'm trying to be diplomatic when I said it, but people even in software, people who do software verification, they love to find bugs. They love to find problems, because that's their job. And so the good suppliers, it's in their own self-interest not to do shortcuts.

Now if you don't trust that, and you do all your same testing and Q&A again, that's a duplication of effort. The way I've seen that done effectively is that you, as a customer, say you're going to buy the subsystem or the engine for, example, from it. What you do is you send liasion people, you send representatives, who are knowledgeable people to the supplier while the testing is being done. And so they're present when the testing has been done. They're very involved with it. And therefore, you don't have to do it twice. So there are ways around this.

OK, so what I did here is just search for the word test and the list of milestones. And where does it come up? So the first time it really comes up in a major fashion is at the CDR. OK, so let me just read this to you. So the CDR demonstrates the maturity of the design and is appropriate to support proceeding with full scale fabrication, assembly, integration, and tests. So in other words, even at the CDR you're blessing the final design, you should say something at the CDR about how the testing will be done. In fact, test planning often is way before the CDR. But at the CDR, you should you should really talk about the testing.

Then we have so-called TRR, which is a test readiness review. And so for each major test, you would have a separate TRR. The TRR ensures that the test article, hardware software, the facilities, the support personnel, the test procedures are ready for testing data acquisition reduction, meaning data post-processing and control. And then at the system acceptance review, the SAR, that's when you essentially transfer the ownership of the asset. And that's at the SAR, at the system acceptance review, that you're going to review, not just the product and its documentation, but all the test data, the analyses that support verification.

So at CDR, you say this is the testing we'll do. At the test review itself, you say everything is ready for the tests to happen, and then you do them. And at the system requirements review you look backwards and you say, what tests actually happen? What's the documentation? What were the results? Are we ready to own the asset now? Does that does that make sense?

OK, so with that in mind, let's talk about testing itself. What kind of testing there are, and so testing is one of the four methods of verification. And it's the one that we often spend the most money on. So this is from the handbook, section 5.3. This is basically an alphabetic list of the testing that we typically do. And I'll just highlight a few here, and then I have a group exercise for you. So aerodynamic testing, burn-in testing, which is often done with electronics. Make sure that you use you burn-in your electronics, you get them running at the right conditions, drop testing, pressure testing, pressure limits, thermal testing, G-loading, human factors testing, thermal testing, manufacturing random defects-- that's when you do nondestructive inspection-- thermal cycling, vibration testing, and so forth. So this is 20 or 30 types of testing. And then within each there's even subtypes, so there's a lot of different-- and there's a whole industry actually that is primarily focused on providing test equipment, sensors, data logging equipment. It's a big industry not just in aerospace, but throughout.

OK, so I'd like to do a little turn to your partner exercise. And the question is I want to ask you what kind of testing have you been involved in the past. And if this was like in product design, product development, that's fine. If it was for an internship, but even at the University itself, if you did some experimental work and experimental testing as part of research, that's fine too. You can talk about that too. So what kind of testing have you been involved in the past? What was the purpose of the testing? What were the challenges? What went well? What were the results? Maybe if it didn't go well, talk about that too.

All right, good. So let's see, we're going to go back and forth. So who wants to start here at MIT? Who has a good story to tell? Go ahead.

**AUDIENCE:** So I worked on the ground station side of the Lunar Laser Communication Demonstration program that recently flew. So I was involved in assembling an integration, but also doing verification testing in the lab and at our field site, but then we did validation when we be moved out to the field site in New Mexico. So I was involved in the whole process and it was neat to see. And we had to go into the clean room a few times to adjust the optics, because we saw that there weren't meeting requirements.

**OLIVIER DE WECK:** The reflector that was left by the astronauts, are you using the reflector that was left on the surface?

**AUDIENCE:** No, this was-- we were using just like [INAUDIBLE] like optical alignment stuff in the lab. And then when we were out at the field site, we were utilizing guidestars to align optics.

**OLIVIER DE WECK:** So what was-- what went well? Was there a big difference between indoor and outdoor? What surprised you in these tests?

**AUDIENCE:** Yeah, so once you can do the alignment of the individual telescopes which were 20 inches in diameter, you can do that well in the laboratory, but then every time you assemble and disassemble the system, you change the alignment of them relative to each other. So there was a lot of attention paid to making sure that we could replicate the alignment to a certain extent. So that was very difficult in a laboratory setting to get done, but once we did that, we were able to have fair confidence that when we were out in the field that we could match that.

**OLIVIER DE WECK:** Very good, very good. What about EPFL?

**AUDIENCE:** Well, I did an internship in an aluminum-roll product factory. So basically, I was doing natural science there. And there was a whole bunch of tests to do. And well, all the tests was about heat treatments and different tempering.

And actually, the alloy that was already produced in the factory was not the best of what we can have of it. And it [? applied ?] to change the [INAUDIBLE] with the heat treatment for a few seconds, naturally, on the line of production. Adding this amount of time was totally critical because it was continuous.

And the rolled aluminum, if it spends a bit more time in the oven, it would melt. And that's a bit like for the Swiss plane, actually. Like I discussed with my boss, saying that we should maybe change the original power meter. But at the end, it was really critical to change something on the line because it could have cost a lot, like in the modification of the oven or the general machine.

**PROFESSOR:** So were those tests successful? Were these heat-treatment changes eventually implemented on the line? Or did the tests reveal that it would be too difficult?

**AUDIENCE:** Unfortunately, I don't know because I finished my internship before.

**PROFESSOR:** OK. Well, you should find out whether it worked out in the end. Very good. Back to MIT, any other examples people want to mention? Test experiences? Yes please, go ahead.

**AUDIENCE:** We bought the CASA-295. It's a small cargo aircraft. And we get we got involved in the development of its simulator.

It was pretty different, the simulator. Because as the aircraft has no fly-by wiring, it is pretty light. So lots of hydraulics to de-motion.

And they brought the flight model from the factory. And we applied the flight model to the simulator. But it was not real enough. So we had to go for flying, like 60 test flying points. And we have to go back to the simulator to apply these points to tailor the simulator to meet reality.

**PROFESSOR:** I see. So the purpose of this testing-- because the plane itself had already been certified, it sounds like.

**AUDIENCE:** Yes.

**PROFESSOR:** It's Spanish, right? Spanish airplane?

**AUDIENCE:** Yes.

**PROFESSOR:** You tested it specifically to get flight dynamics and other data to then tune the simulator to be more reflective of reality.

**AUDIENCE:** Exactly. Because the flight model from the factory was not close to reality at all,

**PROFESSOR:** Very, very cool, very interesting. So different purpose, of not testing for certification of the first airplane, because it had already been certified, but to get the simulator to be matching more closely.

**AUDIENCE:** It was development for the simulator. Because it was sold afterwards as a type delta simulator.

So it was the development of the simulator.

**PROFESSOR:** OK. Great, thank you for that example. This all sounds pretty good. Anybody involved in test failures? You know, things that didn't go well? Yes, [? Narik? ?]

**AUDIENCE:** Well, it was an interesting experience. We were designing a wind turbine that we were 3-D printing in undergrad. And we had certain requirements on the wind turbine. And we were supposed to test in the wind tunnel afterwards.

What happened was that the wind turbine matched our performance prediction fairly closely. But the generator and the electrical power system that the test operators consisted of wasn't designed to handle the current that we were outputting. So we caused a small fire.

**PROFESSOR:** OK.

[LAUGHS]

So this was the test equipment itself?

**AUDIENCE:** The test equipment.

**PROFESSOR:** Not the artifact you were testing failed--

**AUDIENCE:** Yeah.

**PROFESSOR:** --but the test equipment around it, because overload.

**AUDIENCE:** The interesting point was that we had no control over the test equipment. It was managed by the university. So within the requirements that they gave us, the power output possible didn't match what they had.

**PROFESSOR:** OK, great. Great example. So the test equipment and the test artifact need to be matched to the test conditions. Excellent, good.

I do hope that you those of you that have not had a lot of test experience, that you get to experience it. It's a lot of work, slow, tedious. But in many cases, despite modeling and simulation, there's still a big role to play for actual testing.

OK, so let's talk about aircraft testing. Typically we distinguish between ground testing and flight testing. Weights and balance, I had some experience with this on the F-18 program. You think this is the most trivial testing there could possibly be, you just put an airplane on a scale and that's it.

Well, it turns out it's actually more involved than you think. First of all, airplanes are very big. They're heavy, multi-tons. And typically it's not just one scale. You have several scales you put on the landing gear.

So the scales need to be properly calibrated. If you have differences in calibration of the different scales, you have an issue. You need to determine the mass. Not just the mass, but the CG.

And then the most difficult thing to experimentally determine, at least in a 1G field, is the inertia matrix, if you need to experimentally get the inertia matrix. Do you remember your Ixx, Ixy, Iyy? The inertia matrix is tricky because you typically then have to suspend the airplane.

And just the presence of the cables and the suspension will pollute the real inertia matrix. And you have to subtract out the effect of the suspension system. So something that seems super trivial, weights and balances-- you just stand on the scale in the morning, there it is-- is actually very tricky. And there are people, that's all they do. They do weights and balance testing for spacecraft, aircraft. And it's basically a science.

Engine testing, I'll show you some pictures. This is done in what's called the Hush House. So Hush House is heavily insulated. You run an engine through all of its test conditions, its operational conditions. And then you integrate it into the airplane and you run it outdoors.

Fatigue testing, this has been a big issue on the Swiss F-18. But in general, making sure that the airplane can satisfy all the static and dynamic structural load conditions. Avionics checkout, this is very, very involved.

As we get more and more displays, mission-control computers, all of the avionic suite needs to be checked out. Essentially every function, every button, every menu item needs to be tested. And the tricky thing is interactions among different pieces of avionics. So you can't just test each box in isolation.

You also have to look at the interactions of different pieces of avionics, the flight control software, or the flight software that is loaded in each of these avionics boxes needs to be in the right configuration. It's a very big combinatorial challenge to do avionics checkout these days.

And then finally, pre-flight testing. So this is everything you can do on the ground, run the engines, taxi with the airplanes, basically turn all the equipment on, turn it off, do the cycling. You could do a lot of testing before you actually fly.

Flight testing itself falls into different categories. So flight performance testing, rate of climb, range, can you meet each point in your prescribed performance envelope? Stability and control, this is where test pilots typically earn their living putting airplanes into stall conditions, recovering from stalls, trimming.

Flutter testing is a big deal. So flutter is a phenomenon whereby at high speeds you have a coupling between the structural deformations of the airplane and the actual excitation of, for example, the wings. Flutter can be very dangerous. If you hit a resonance at high speed you can actually destroy the airplane because of an instability.

So flutter testing is also very, very interesting and very tricky. And then finally, this is primarily for military airplanes, weapons testing, both guns, missiles, bombs, live fire testing-- sometimes also using airplanes that are towed-- simulated targets, and then LO stands for a Low Observability.

So this is essentially all the new generation of military airplanes have measures to reduce their radar signature, or even make them invisible or quasi-invisible to radar. And you know, a lot of this stuff is classified. But actually checking that an airplane is invisible on radar or has truly low observability, there's a lot of testing involved in that. And that's also quite expensive and very involved.

So let me show you just some pictures that I've collected over the years. This is a wind tunnel test model. This is a model that was developed as part of the F-18 program. This is about 1995, vintage.

This model, it's a subsonic wind tunnel model. And you can see in yellow, you have all these probes and radomes and things like this. So it's basically to check whether any modifications you make to the airplane will affect its performance in airflow.

This is for wind tunnel testing. This model, by the way, just building this model is about half a million dollars. It's very accurate. It's very precise. This is a picture-- yes? Go ahead?

**AUDIENCE:** Is that model full scale or half scale?

**PROFESSOR:** No it's, I want to say, like 1/8 scale, something like this. Yeah. OK, here's the Hush House that I was talking about.

This is in St. Louis. So you can see that the airplane is not painted yet. And only one engine at a time. So the engine is being, in this case, with full afterburner, you can see the airplane itself is secured with these chokes here. And there's load cells in these chokes.

So as you fire up the engine, you can measure the thrust by the load cells that are attached to these chokes here. You also see that there's these cables running in and out of the airplane. So all the sensors, everything, and the engine is put through its full different operating profiles. And a lot of sensor data is recorded to make sure that the engine responds appropriately, it has the right thrust for the right throttle setting, the fuel consumption, all the temperatures in the engine, that everything is nominal, essentially.

Live Fire Testing, this is a Maverick missile being fired, an air-to-ground missile. As you can imagine, there are special ranges and test sites for doing this kind of work. So in the US, one of the most well known as China Lake, out in California.

You have to reserve months and sometimes years ahead. So if you want to do like a live fire test campaign, you have to reserve the range at least 18 months to 2 years ahead of time. Because a lot of other services, a lot of other programs are using the same facilities.

In Europe, it's a little harder. Definitely in Switzerland, because the country is so small and dense and highly populated, you can't test live missiles in Switzerland. You can do guns, air to ground. But in order to do missile testing, typically that's done here in the US, or in a more limited fashion, in Scandinavia, like in Sweden, in northern Sweden, there are some test ranges up there.

This is the most expensive kind of testing you can do. So a single test like the one shown here, a single test like this will probably cost several million dollars. Not just the airplane and the weapon itself, but all the test procedures, the protocols, airplanes that observe it from all kinds of angles. It's very, very involved. And because it's so expensive, you will typically only do it for something new that you haven't done before. Either a new weapon, or a new weapon integrated on a new platform and so forth.

And obviously, it's very interesting. But it's very involved. Yes?

**AUDIENCE:** Are they just testing accuracy, Or that the two things work together? What are they looking for, really?

**PROFESSOR:** The first thing you look for is does the weapon fire? So do you have all the electronics? All the signals? The wire bundles? Did you get it right?

Is there an end-to-end functionality? That's number one. Number two, safety. Does the weapon separate properly from the aircraft? The worst thing that can happen to you is if you release the weapon and it collides with the airplane.

And so you can see, the various angles and release conditions are very tightly prescribed. And so there's a separation, has to be proper. And then the third, of course, is accuracy.

So within each of these tests, there are multiple sub-objectives that you would test for. But safety always comes first. OK, any questions? This was a little bit military-aviation heavy.

If you're testing, whether it's a CASA airplane or a new Airbus or Boeing commercial airplane, many, many months of testing. They actually fly the routes. You'll fly New York to Singapore, to London.

You would actually fly the real routes. You would record fuel consumption, a lot of parameters. Some of this testing is not very exciting. It's many, many, many, many hours in the air.

But the key is that you have a lot of instruments and sensors during these tests that you may not have during regular flight operations, to really make sure there's no surprises. The airplane flies at least as good as the requirements that you promised your customers. And even then, when you think about what happened to the Dreamliner, the 787 had a lot of battery problems. Because they used a lot of lithium ion batteries. There were overheating issues.

Some of these problems didn't show up in testing. They only showed up in early operations, once you had a fleet going. So it's not a guarantee because you're doing a lot of testing that you're going to catch all the problems. But you want to catch as many as you can. Yes?

**AUDIENCE:** So my question was about the risk posture for larger airliners, for Boeing, the Airbuses. So for military aircraft, there is an escape method for the pilot. But for these larger aircraft, how much analysis do they do before they decide to go ahead and put a person inside? They do fly by wire beforehand? Is that possible. for such large aircraft?

**PROFESSOR:** So that's where the ground testing, pre-flight testing becomes very important. So you basically taxi for many, many hours. All the flight control surfaces, all the engine, you have the Hush House testing. So you essentially try to do as much as you can on the ground before you do the maiden flight.

All right, let's move to spacecraft. And it's kind of a similar thing. You can distinguish the ground testing versus on-orbit testing.

So the ground testing is really not that different, weights and balance. The biggest thing is if your satellite is heavier than the launch capability of the launcher, you have a real problem. So the mass constraint is even tighter in spacecraft.

Then, a lot of testing on antenna and communications. This is typically done in anechoic chambers in the near field, and then later in the far field. Vibration testing, that's the shake part. Thermal and vacuum-chamber testing, that's the bake part.

And then you also have pre-launch testing, so off-pad and on-pad. Off-pad testing is the satellite or the spacecraft has already been shipped to the launch site, and it's hooked up. Like a patient in the hospital, it's hooked up to a lot of cables and power and cooling and so forth.

And then when it's on the pad, it's pretty limited. So on the pad means the satellite or the spacecraft is already integrated into the launch vehicle. It's on the launchpad. And then that amount of testing you can do is very limited.

So that's when we say off-pad, on-pad is, is the spacecraft already been integrated on the launcher or not? Once you launch to orbit, you got your eight minutes of terror. And hopefully the launch goes well and the spacecraft is released into its initial target orbit.

And then you do a lot of other tests, like thruster testing. Can you do station keeping? Can you turn on and off the thrusters? You deploy all of your mechanisms, your antennas, your scientific instruments, and then your communications, communication and instruments.

And we'll talk more next week, but this typically is called commissioning. You're commissioning a spacecraft before you actually turn it over to the users. And that commissioning phase could be anywhere from a few days to several weeks, or even a couple months.

And again, you don't want to randomly put commands into your spacecraft. These test sequences, deployment sequences, are very, very, very carefully worked out. Every command, the order in which you send the commands have been worked out ahead of time. They've been simulated. And all you want to see here is confirmation that the spacecraft behaves as planned.

Some pictures, so this is what typical spacecraft integration testing looks like. So this is in a cleanroom environment. You have people in bunny suits. And the idea is to not damage the spacecraft while you're doing the testing.

This is a picture of the Clementine spacecraft. This is a radio frequency anechoic chamber testing. So you see these funny cones here. These are essentially foam cones. And the idea is to prevent multipath to prevent echoes in the test chamber, to test all of the antennas, EMI, electromagnetic interference and compatibility, charging and discharging of the spacecraft.

This is one of the failure modes, is that you have high electrostatic charges that build up on a spacecraft, create a lot large voltage potential across the spacecraft. Some spacecraft have failed because of that. So all of these things you want to test in a very controlled environment.

James Webb Space Telescope, I'll send you a link. I'll send you a link through email. There's a simulation of this on-orbit deployment. It truly is amazing. This spacecraft will be launched in a box, essentially. And the deployment sequence is very carefully choreographed.

First, typically, you deploy your solar panels because you need power. Because you're only running on battery initially. So if your battery runs out before you've had a chance to deploy your solar panels and get fresh power into it, you're in big trouble.

So typically, solar panels first. Then communications. And then you start deploying the other subsystems. So for James Webb, also very tricky, is this-- this is called the Sunshield. It's essentially thin layers of insulation. And the geometry is very important. The primary mirror, the secondary mirror, all these things need to be deployed with very, very high precision.

And it's even to the point where this particular spacecraft is so lightweight that it cannot support its own weight in a 1G gravity field. So there is no way to test, end-to-end, the full deployment sequence on Earth. The first time it will happen is in orbit.

Now, they've tested sub sequences or scaled models. For example, the Sunshield has actually been deployed at a smaller scale in a 1G field, but never the full thing. So this will be kind of scary, after an $8 billion investment. So let's hope for the best, 2018.

So testing is good, testing, testing, testing. But testing also has its caveats. So caveat means limitations, essentially. So testing is critical, but it's very expensive.

Think about test rigs, test chambers, sensors, DAQ is Data Acquisition Equipment. All this stuff is very expensive. And if you can reuse things between different programs, that helps. But still, how much testing should you do of components?

So one of the comments, who mentioned the vendor, the supplier? One of you. You talked about it. And this is a key question. Do you trust the parts that come from your vendors? Or do you retest everything yourself?

Calibration of sensors and equipment, if you've done some testing and you forgot to calibrate your displacement sensors, your thrust sensors, you didn't calibrate them or they're out of calibration, that's a big problem. That's a big problem. So before you start your tests, make sure that all your sensors are properly calibrated, or you can get the wrong conclusions.

This is a mantra that's well-known. Test as you fly, fly as you test. Fundamentally, this means that the configuration of your item-- a spacecraft, aircraft, medical device-- the one that you test should be the same configuration as what you're actually going to fly.

And it's often failures occur when the test went well, but then somebody tinkered with it and modified it before it actually flew. And that change actually caused a big problem. So make sure that your test conditions reflect the actual operations as closely as possible.

Simulated tests, what do we mean by this? So simulated tests use dummy components. Maybe your full spacecraft or aircraft isn't ready yet, you don't have all the pieces. So you can still start testing, but you have to replace the missing pieces with dummy components. At least, they should reflect the right mass distribution. But maybe you can do more.

Simulated operations, so the 0G versus 1G, is it representative? And then, what's often true is that you pass all your tests and then you still have failures in practice. And the failures often happen outside of the test scenarios that you had tested. So you have to be ready for that. But try to avoid that.

So here's from Appendix E. This is called a Validation Requirements Matrix. Essentially what this is, is an organized way to organize your V&V activities, in terms of what's the activity?

What's the objective? Which facility or lab will you do it in? What phase? Who's in charge? And what are the expected results?

It's pretty straightforward. It's just a table to organize these activities. And then appendix I is your more formal V&V plan. This is a suggested outline for it.

And I'll just say this. The degree to which you take Verification and Validation seriously and the resource you make available for it are critical for success. So how many dedicated Q&A personnel?

What is the interaction in working with suppliers? Are you planning ahead for these tests? How close are you getting to actual end-to-end functional testing?

Can you piggyback on existing facilities and equipment? How well do you document all the outcomes and follow up with discrepancies? And my last comment here is this work is often not glamorous, except for some of the very cool flight testing that I showed you.

Most of this work is really hard work. It's very detail-oriented. It's not glamorous. But it's essential. If you cut corners, you often pay the price for it.

So any comments or questions? We'll take a short break, like a five-minute break. But any questions about testing, verification, validation? Yes, go ahead.

**AUDIENCE:** Not really a question, but I wanted to say that also, flight testing is not so fancy. I mean, many people think it is. Well, if they actually fly, maybe it's fun. But there are not that many. And you need to prepare them weeks and weeks ahead. And it's actually very, very boring. Because you need to make sure that you don't waste time at all. Well, at least you can waste a little bit more time in the lab. So it's very stressing and not so fun to fly.

**PROFESSOR:** Do have experience with this?

**AUDIENCE:** I wasn't flying, because you need to be certified. But I was preparing. And I think it's even worse than testing in the lab.

**PROFESSOR:** Yeah, yeah. Which airplane, or which system were you involved with?

**AUDIENCE:** I was in with the power plane system for [? Airbus, ?] particularly. And the aircraft was an A330.

**PROFESSOR:** A330, OK. Great. But I think it's healthy to have this experience. It really makes you humble.

And you also see, for the things in design, did you design well? Did you design for testability? Really, I highly recommend for every one of you to try to get on some kind of test campaign, at least once in your career, because it's eye opening. So thank you for that comment.

Any comments at EPFL? Any of the students? [? Voelker? ?] Did you want to add something? Katya? Go ahead.

**AUDIENCE:** Yeah, I guess the comment is, it seems like sometimes you can meet all of the requirements in the verification process. But when you get to the validation part, for example, maybe the customer has some expectation that the range for the time of flight would have been on the maximum edge and you were on the minimum edge. But actually, it seems like sometimes you can meet all the requirements but they're still not going to be happy.

When do you find that middle ground? And it's going to be constant, to continue iterating this over and over again. Do you try to involve them earlier on, during the testing process too? How do you handle that, that difference?

**PROFESSOR:** Yeah, it's tricky, you know? So that's when you do need contractual agreements in place. You need to have the requirements, baseline. You need to have the contractual agreements. And hopefully, any problems that occur will not lead to some kind of legal dispute. But sometimes that's unavoidable.

But as a designer, as a manufacturer, unless you have agreements in place and clear baseline, how do you decide in the end, is it successful or is it not successful? And if there's problems, try to isolate these problems and say, OK here, by and large, the testing went well. But we have like, three, four, five issues that need to be addressed. And you can tackle these issues one by one.

But if you don't have a contract in place that's really a good contract, if you don't have a clear requirements baseline, and then if you don't have a good relationship with your customer, you're setting yourself up for big, big problems. [? Voelker? ?] Go ahead.

**AUDIENCE:** Yeah. There is also the one big difference between commercial operators or commercial customers that are becoming more and more frequent compared to the institutional ones. And often I remember, for some of the [? Global ?] [? Store ?] [? Iridium ?] series, the customer was only accepting the hardware six months after they had been commissioned in orbit.

So there you have a validation that is still your responsibility in orbit. You can't go and fix it, but it still has to work. And he was retaining up to 10% of the full contact value, even up to the N minus-2-tier-level suppliers, until he was satisfied it was working in orbit. So these considerations, it's really the proof of the pudding when you have to test this up there and can't fix it. So there's no recall of a satellite constellation, like, sorry we messed up with the software. It's not possible there.

**PROFESSOR:** Yeah. And of course, those terms and conditions you've probably negotiated years before. So you've got to be careful. That's where risk management, which is actually-- thank you, [? Voelker, ?] for that comment. Let's take a short break. And then we'll talk about risk management, which is really what this ends up being.

So let me talk about risk management. And this is actually quite prominent in the System Engineering Handbook. This is right in the middle here of your System Engineering Engine, Technical Risk Management, Section 13.

Why is it important? So first of all, what is risk? So risk is the probability that a program or project will experience some undesired effect or event. And then the consequences or impact or severity of that undesired event should occur.

And so think of risk as the product of probability times impact. And the undesired events could come from a number of things, technical, programmatic. So cost overruns, schedules slippage, safety mishaps, health problems, malicious activities-- cybersecurity is a big thing these days-- environmental impact, failure to achieve the scientific or technological objectives or success criteria.

And so technical risk management is, therefore, an organized systematic risk-informed activity centered around decision making to proactively identify, analyze, plan, track, control, communicate risks to increase the likelihood of success of a program. And so what risk really does is measure the future uncertainties of achieving your program goals-- technical, cost, schedule goals-- and think of risks in a holistic way, all aspects of the technical effort, technology maturity, supplier capabilities, performing against plan, and so forth.

And so the idea of risks is that risks have some root cause. There's something that gives rise to risks. And then the actual quantification of risks happens in terms of likelihood and consequences, which are kept separate, separate dimensions.

So the first thing to think about is where do risks come from? Where is the source of risks? And I want to show you a couple of models for thinking about this.

The first one is this idea of layers of risk, that there are layers of risk. And I want to credit one of my colleagues here at MIT, Don Lessard from the Sloan School, who really developed this Layer of Risk Model and applied it to different industries. So there's a version of this for the oil and gas industry. You could make a version for medical, medical technologies. So this is the version for Mars missions, so if you're designing a new Mars mission, a new Mars Rover.

So you have, in the bullseye here, the narrow interpretation is technical or project risk. So the airbag technology. If you're using airbags for deployment, will it work? The rover/motor performance, are you going to have software bugs? Those are the risks we typically think about.

And the idea is you have high influence over these risks as a system engineer, as a project manager. Then you have a layer around it, which we call industry or competitive risks. Will your contractors perform? Will you have budget stability?

And then there's sort of more country and fiscal risk. So in the US, we have a budget cycle. We have four-year administrations. Will you get your budget? What is the priorities between human and robotic space exploration? And then working with international partners.

And then there's another layer of risk, which are called market risks. So if you think in Mars missions, who's your market? Well, the science community and maybe the public. So will these missions hold their attention?

Are there new science requirements? We discovered there's water, probably flowing water on Mars, maybe with a lot of perchlorates in it. It's not pristine water. But that could change the priorities for your mission.

And then finally, the most outer is what we call natural risk. So this would be things like cosmic radiation, micrometeorites, uncertainties in the atmospheric density of Mars as you're doing entry descent and landing. And you have very low influence.

That doesn't mean you can't protect yourself or take measures to deal with these risks. But fundamentally, the occurrence or the probability is something you can't really do much about. So that's one way to think about risks. And the seeds of risks is in these layers. I know this is very high level, but I find this to be a pretty useful model. Yeah, go ahead.

**AUDIENCE:** So this references the influence you have, not necessarily the amount that each of these are a risk to the program?

**PROFESSOR:** That's correct. And That will be program specific. Just the stuff that's in the bullseye here, you can do a lot about it, and perhaps both in terms of probability and impact. And then as you move further out, there's less and less influence you have as a system engineer, as a project manager.

Here's another way to organize your thinking around risks. And this is around the "Iron" Triangle and project management. We talk about the "Iron" Triangle of cost, schedule, and risk. And we call it "Iron" because the idea is that if you constrain all three too tightly, it can be very difficult.

And it's also referred to as the triple constraint in project management. So the three dimensions here are technical risks, cost risks, and schedule risk. And in the center we have programmatic risk, which means it's kind of the combination of all three.

And the idea that even if you do a great job on keeping your budget under control, schedule, and you're meeting your technical objectives, you can still fail because the program as a whole isn't doing the right thing. Or the market that you had been targeting is no longer really attractive by the time you launch. The key idea here is that these risk categories are not independent of each other.

So let me mention a couple of examples. So cost risk might limit your funds. And that could, in itself, induce technical problems which cause you further cost risk.

So one of the big initiatives at NASA in the '90s was the faster, better, cheaper program. We're going to launch more missions, cheaper. And out of 10 missions, maybe 2 or 3 will fail.

And then seven will succeed. But we'll get more value out of this as a portfolio. Unfortunately, it didn't work very well. Because when the one, two, or three missions fail out of your portfolio, the media and the public focuses on the failures rather than the aggregate value of the whole portfolio. And eventually, that's probably the main reason why faster, better, cheaper was abandoned.

So for example, we just talked about testing. If you have a very limited budget, what is the first thing people typically cut out? What's the first thing to go? Testing. Testing is very important.

Sam asked me during the break, what's your typical budget for testing in V&V activities? And in many programs, it's very substantial, you know, 40% of the budget, maybe 30%, 40% of the budget easily. And so you start cutting out tests.

Well, what you do is you introduce technical risk. And if you have failures because you didn't test, that could cause you additional rework and more cost. Similar, schedule slips can induce cost risk.

So as you slow down, you have what's known as the standing army cost, right? People are going to charge to your program, even if it's at a reduced level. And that will also increase your cost. So lots of coupling here between risk categories.

This is a very useful Risk Management Framework. It's essentially a controls framework. And the idea is you start in the upper right.

You anticipate what can go wrong in your program. So that's risk identification. You then analyze these risks, in terms of prioritizing them, which of these are important?

You plan to take action. This is often called risk mitigation. You track these actions. And then you correct any deviations from your plan and you communicate throughout, and you cycle through this. So typically, risk management will happen on a weekly basis, a monthly basis, at least quarterly basis for big programs.

Now, how do you actually do this? First of all, the risk ID and the assessment. So the risks are typically brainstormed.

So you think about risks. You have to imagine all the bad stuff that could happen to you and your program, the probability that these things will happen, and then the impact or consequence if they do happen based on the requirements, the cost, the schedule, the product and its environment.

And this is where actually having a mix of younger engineers and more experienced engineers really comes in handy. The experienced engineers, they will have been through several programs. They will have seen failures in the past.

They will really be able to point to potential risks that less-experienced people may ignore or just not understand how important they could be. So the next step, then, is to aggregate these into categories, typically not more than 20-or-so categories or risk items. Projects often keep so-called risk registers, which is just a database or a list of risks.

If you have hundreds and hundreds of risks in the risk register, it's too much. It's just a long list and really it's just a check-the-box exercise. To really take risk management seriously, you have to focus on few of the risks that you think are important.

You score them based on a combination of opinions and data. And you try to involve all the stakeholders in the risk management. And eventually, risks are placed on this matrix of uncertainty and consequence. So let me zoom in on the matrix.

There are many, many different versions of the risk matrix. This is one that NASA typically uses. And I like this particular version for several reasons that I'll explain. But basically, the way it works is you have the two dimensions, impact and probability.

So probability is how likely is this to occur? And then impact, if it does occur, what will happen? What's the consequence of that?

So of the two things that I really like about it, the first one is that each of these levels, there's actually some definition behind it. Not just guessing at the level, but there's some criteria. So for probability, a level 3 means it's about equally likely that it will happen and not happen. So a level 3 means it's about 50/50, whether this will happen in your program. And then 4 is very likely.

So maybe that's, I don't know, 75%. And then near-certainty is like 90% or more. Improbable is like 10%. Unlikely is 20% to 30%, something like this.

And then more importantly, the impact so a level 1 impact is negligible. It has almost no impact. A level 2 means your mission performance margins are reduced on the technical side.

Do you remember margins? I asked you to assign margins in assignment A2? So it means you're eating into your reserves. But you should still be able to meet all your performance. There should be no visible impact. Your safety cushion is less. That's what level 2 means.

Number 3 means your mission is degraded. So you can still do the mission. But you're not going to hit all your targets.

4 is you lose the mission, but the asset is still recoverable. So maybe you could try again in the future. And then level 5 is a catastrophic failure that involves a loss of mission and/or loss of crew.

On the cost side, you have some thresholds for cost. Obviously these numbers have to be adjusted for different programs. A $10-million loss is a huge thing in some program and almost like pocket change in other programs.

And then schedule, so a level 1 milestone would, for example, be launch. Good example of this was the Mars Science Laboratory, their Curiosity mission. Originally it was supposed to launch in 2009.

They missed that deadline, mainly due to problems with cryogenic actuators. They took a lot of the blames, the actuators. But there were a lot of problems across the board.

They missed that launch window. And they had to launch in 2011. So that was considered, from a programmatic standpoint, a level 5 failure. Because you missed your main launch window and you had to wait 26 months for the next one.

So that's good. Because now when you assign a probability and impact, you can really look at these criteria. And it's easier to do that in a repeatable fashion. The other thing is if you look at the colors on the matrix, you can see it goes from 1, blue, which means low risk, to 12, which is the highest risk.

So there's 12 risk levels. But when you look at the matrix, there's something peculiar about it. So look closely at the colors. And you'll see something special about this matrix. Anybody notice what I'm talking about?

Let's see at EPFL, do you guys, when you look at those colors, at the matrix, do you notice something? Go ahead.

**AUDIENCE:** It goes from blue to red, which is a light spectrum, with the blue the lowest radio waves, waves, and red the highest ones.

**PROFESSOR:** Right.

**AUDIENCE:** [INAUDIBLE].

**PROFESSOR:** Go ahead.

**AUDIENCE:** It's due to impact is more serious than probability.

**PROFESSOR:** Right. So it's asymmetric. You see that? It's asymmetric. So the high-impact, low-probability corner is weighted more heavily than the low-impact, high-probability.

And that's intentional because it's been shown in the past that things that are not likely to happen but if they happen, they're really bad, in the past, people have sort of pushed that away and ignored those. So the purpose of this asymmetry in this matrix is to elevate the low-probability, high-impact events to be higher in the risk level so that people pay more attention to it. OK? So most risk matrices don't have that asymmetry in them, but this one does.

[SNEEZES]

And I like it because of that. Bless you. So the question then is, what do you do with this?

The idea is you do your risk management. You identify your risks. You place them on this matrix. And then you track each of these risk items over time.

So here's your 12 risk levels, between 1 and 12. That's the y-axis. And then on your x-axis there's Time.

And for each of your risk items, people might disagree. Some people on your team might say, hey, look, this is not a big deal. We've seen this before.

The impact is not big. We have a quick fix for this. We know how to deal with this. And other people disagree and say no, this is very, very serious. You have to take it seriously.

So the idea is that for each risk item, you have this optimistic, expected, and pessimistic estimate of what is the true level of risk. That's what these bars are. And then you track it over time.

And depending on whether you're before PDR or CDR, you could have very substantial risks. And that's, I guess, OK still, as long as you find ways to reduce the level of risk. For example, by doing extra testing, by changing your design to put in extra power margins, bigger solar panels, redundancy, for example. There's a lot of things you can do to affect both the probability and the impact, radiation, extra shielding.

And so the idea is that over time you're going to reduce these risks gradually below some threshold. So this red line here is like the acceptable threshold of risks at that point in the program. And then as you get closer to launch, things should be below the threshold. And if it's below this watch domain, then you even don't track it. You don't pay much attention to it.

If it's above this red line, you have a big problem. You might have to stop the program or do a major redesign, or repeat a major milestone. And some programs have been canceled because they just couldn't get these risks under control. So the idea is that gradually you transition. And you do this by actually doing risk mitigation around that risk management cycle.

Now, the last thing I will say here is that every mission that is worthwhile doing is still going to have some residual risk at the end. The requirement is not that all the risks are at 0 in the lower-left corner. You will launch, and you're going to have residual risks. And you just have to accept those. But you have to be cognisant of this. And this is really no different in the automotive industry, for example.

When you're developing a new car or a medical device and you're going to launch it to the market, if your requirement is 0 risk, you will never sell anything. You will never launch anything. Because you will always think of something bad that could happen. And there will always be people saying it's too risky. We can't do it.

So knowing how much residual risk you should be willing to carry is a big part of being a leader, being a system engineer, really understanding things. And in the automotive industry, there are people whose primary job they have is to do this work.

And they're called Quality Engineers or Warranty Engineers. So the Warranty Engineers, their job is twofold. Before you launch a vehicle to market, it's actually ranking on this particular vehicle or program, what are the top 10 things that could cause warranty planes and problems in the future? We don't know that they will, but they might. Right?

And then once a vehicle goes to market and reports are coming back from users and from the fleet, actually tracking what these issues are and then knowing when has it hit a threshold where you do need to do a recall, you do need to do a retrofit, this is a big part of it. And it's really a big deal. I mean, the amount of money that automotive companies spend on recalls every year is about the same as what their profits are.

So if you could eliminate recalls and warranty claims altogether, you'd basically double your profit. And so depending on what industry you're in, whether it's automotive, medical, spacecraft, how much risk and safety is involved, this is more or less emphasized in the industry. But it's a big part, I think, of system engineering job, is to understand this.

This is, again, a flow diagram for how to do this risk management properly. You have a Risk Management Plan, your technical risk issues that are placed on the matrix, any measurements or data you have. And then how do you report this?

And then out of it comes a mitigation plan, a set of actions, technical risk reports, and then any work product from the technical risk management. And the idea is that you repeat this process on a regular basis. And it's a big part of your milestone reviews as well.

OK, so I'd like to spend a few minutes on system safety. I am not going to do this justice because there's a whole class here at MIT on this, taught by Professor Nancy Leveson. By the way, who's taken that class? Or who's been thinking about taking it? About three, four of you.

So I'm just going to give you a very quick exposure to this. So this is a book that Professor Leveson wrote several years ago. And she basically distinguishes two kinds of failures. Component failures, which most people think about, an axle broke or there was a battery caught fire. And clearly, component failures are real and they happen, single or multiple component failures. And usually there's some randomness to them.

So most of the classic accident investigation techniques and safety techniques focus on component failures. But there's also component interaction accidents or failures which are trickier, in a sense, because you could have a system that has no single component that's failed. And yet, you had a system failure.

And so it's the interactions among components. And this could be related to interactive complexity in coupling, more and more computers and software, and then the role of humans and systems. And this is really what a lot of this is about.

So the traditional safety thinking is that component failures, you need to worry about the component failures only. So here's a classic example of a sequence of events. This is for a tank failure.

So in this case, we have a tank. And there's moisture that builds up in the tank. And then corrosion, essentially, as a result.

The metal gets weakened. And under the operating pressure of the tank, the tank itself has been weakened due to corrosion. The operating pressure causes a tank rupture.

And the tank rupture then causes, essentially, an explosion. And fragments or shrapnel from the tank will be projected and then cause equipment damage or personnel injury. And so in this linear chain-of-events model, the way you think about safety is putting in barriers.

This is often also referred as the Swiss cheese model. if you take layers of Swiss cheese-- and I guess it has to be Emmentaler, right? You guys know the Emmentaler at EPFL? Emmentaler is the one, it's got the big holes.

**AUDIENCE:** Yes.

**PROFESSOR:** So you take these slices of cheese. And if you can look at the cheese, and there's actually a hole right through, then the accident can happen. But if you put another barrier in between, you can't see through the cheese and the accident is prevented.

That's the classical thinking around system safety is chain of events. And then put barriers between these. And this is, I think, valid for a very particular kind of accidents, which are these component accidents.

What Professor Leveson says in her STAMP and STPA Framework is a little different. This is based on essentially thinking about safety as a lack of control of a system, lack of controlability of a system. And so if you think about it this way, I'm showing you here a control loop.

In this case, you have the actual system, the actual process that you're executing. The controlled process is here. You're sensing things about that process, so temperature, pressure, proper alignment.

And so one problem could be your sensors are inadequate. You're sensing the wrong information. And then here's your controller model, how should you control the system?

So you have the wrong controller or the wrong process model. And then here's your actuators. Are you issuing commands at the right time? Are you issuing the right commands? Or are you not issuing commands when you should be to the system? And that feeds back into the control process itself.

And so process inputs could be wrong or missing. You could have disturbances into the process that are unidentified or out of range. And then eventually, if this process goes unstable, then you have a failure or an accident.

So it's quite different. It's essentially thinking of this as a control problem instead of a chain-of-events problem. And the argument here is that for safety or failures that involve a combination of hardware, software and humans, often this model is able to be more complete, in terms of identifying hazards and potential mitigation actions.

So I think we're out of time. But I want I want to give this to you as a homework for thinking about this. This is an accident that happened earlier this year, I guess, in July. And this is the Virgin Galactic crash that happened.

Virgin Galactic is one of the space tourism companies. And during a test flight, the airplane crashed because the copilot unlocked the brake system. So it has a kind of feathering mechanism. And the pilot unlocked it too early during a high-speed flight phase.

So what I'd like you to do, the link is here. Just read the story. There's a more lengthy accident report that's come out. I'd like you to just read this quickly and then think about how does this relate to risks? How does it relate to this particular model of system safety?

So the System's Theoretic View of Safety is then that safety is an emergent system property. Accidents arise from interactions among system components-- physical, human, social-- constraint violation. Losses are the result of complex processes, not simple chain of events.

And that most accidents arise from-- you could have a system that's quite safe when you start operating it. But over time it migrates to an unsafe state. Because sensors fail. People start bypassing safety procedures. And gradually, it migrates to high risk.

OK. So the last thing I want to talk about-- just for a minute or two-- is the FRR, the Flight Readiness Review, which is one of the later milestones. And what happens at the flight FRR? Essentially, this is your last chance to raise a red flag.

This is the last milestone before launch. Have all the V&V activities been passed successfully? Are there any waivers that need to be granted?

What are the residual risks we just talked about? And then after the FRR has passed, you actually start the countdown-- T minus X days, Y hours, Z seconds-- to an actual launch, or a product launch, whatever it is. And then here's from the handbook, the entrance and success criteria for FRR.

Everything should have been done at this point. Your design, your integration, your testing, your operating procedure, your people should be trained. This is your last chance to raise a red flag. After the FRR, you're essentially go for launch. So the stakes are high.

OK, so a quick summary. Verification and validation are critical. There's a distinction between the two. Verification is against the requirements as written. Validation is you go back to your customer and you test in a real environment.

Testing, many different kinds of testing. It's a fundamentally Q&A activity, and it's really expensive, but it needs to be done right. Risk management, we have different tools like the risk matrix, risk identification, mitigation.

And that's really where the rubber meets the road, in terms of the tension between cost, scope, schedule, and risk in projects. System safety, think about not just the chain of events model, but this control's view as well. STAMP/STPA is a particular framework for this. And if you're interested, there's a whole class. There's a whole set of things you can learn about just safety.

And then finally, FRR is your last chance to raise the red flag. It's sort of the big milestone before you go live with your system. OK? So any last questions or comments? EPFL? Yes, please go ahead.

**AUDIENCE:** So after the FRR you should have finished your tests, right? It's the limit?

**PROFESSOR:** Yes, you should have finished your tests except for the ones that you're going to do, say, on orbit, right?

**AUDIENCE:** Because I was wondering, in the list there is the go, no-go test. And I'm wondering if it's not related to the launch, actually?

**PROFESSOR:** Yeah, so the actual launch itself, of course, has the actual launch countdown. And you can stop the launch. So this is a review. The FRR is more like a CDR.

So the countdown hasn't actually started yet. But if you successfully pass the FRR, that's when you begin the countdown. The official countdown starts. And then you still have a possibility, of course, of stopping the launch. But in terms of a formal programmatic review, this is your last chance.

**AUDIENCE:** I think the point here is more about the what is the system boundary? If you consider the system boundary being the whole mission, including the launch, then obviously the system will only be finished when the mission has finished phase EF.

**PROFESSOR:** Right.

**AUDIENCE:** And this FRR, Flight Readiness Review, is linked to the panel or to the whole satellite, specifically that you are allowed to go forward and now start the countdown issues. And then all the other things. Like, you'd extend the mission boundaries to the whole space program or to the whole colonization of Mars, it will be, obviously, later.