MITOCW | watch?v=J_y2I09rj_I

The following content is provided under a Creative Commons license. Your support will help MIT OpenCourseWare continue to offer high quality educational resources for free. To make a donation or to view additional materials from hundreds of MIT courses, visit MIT OpenCourseWare at ocw.mit.edu.

PROFESSOR: You have to make some design decisions. And then once you've made those design decisions, then you can actually write your level 2, level 3 requirements and so forth, it's an iterative process. But when you look at the classic v model, you don't really see those iterations. But they actually exist.

So what we typically do at the system requirements review is we review and then agree on the high level requirements, level 0, level 1, maybe level 2 requirement, but not the lower level ones, because you can't, because you typically haven't done the design yet.

The other thing I want to say is a lot of the-- this is really a critical issue. And a lot of the problems we talked about last time in the schedule, cost overruns, really, a lot of these things are traceable back to the requirements. Either the requirements were over ambitious. You can set requirements that were essentially unachievable. Or you missed requirements that actually became clear later on. But they weren't written. So nobody paid attention to them.

A couple of examples. So I have sort of one example of core requirements, and then a good example. So this is a mission that you may have heard about.

A 1998 MCO. So who was born in-- I'm looking at you guys. Justin, right? Justice, when were you born? '94? OK, so you're older than MCO by four years.

So MCO was a very well-known just about not quite 20 years ago, a very well known mission failure that happened. A Mars climate orbiter, it was launched on December 1998. And it had multiple functions study the Martian climate, weather and surface changes. And it was to act as a relay satellite for the Mars polar Lander, which came after it and also failed. Both failed.

And unfortunately MCO burned up. We think contact was lost with a spacecraft after it basically entered the Martian atmosphere. And you know nothing-- there was no more communication. So we don't know for sure. But their best guess is that the spacecraft burned up in the approach to Mars, that the actual altitude at which had entered the Martian atmosphere was about 57 kilometers. And it was supposed to be about 220. So even though the Martian atmosphere is very thin, when you enter at very high velocities, that's not a good thing.

This is the very famous-- you've heard about this-- confusion of units problem, like this mission that one part of the team used SI units, specifically Newton's seconds, this is momentum, right? Force times time, that gives you momentum. So when you burn your engines for x number of seconds the sum of the product of those two is your momentum. And then another part of the team used English units, pounds of force times second. OK, and so that was fundamentally the problem that caused the burn up.

Now if you read the accident report. And I have a link here to the accident report, you will notice that the requirements were actually OK. The requirements were written correctly. So specifically, there was a document called the software interface specification, SIS, that specified everything should be in SI units on the ground segment and on the space segment. But the problem is that this was not checked and implemented.

And so here's a quote from the accident report. Items that the mission assurance manager, which didn't exist for this mission. So the mission Assurance Manager role is to make sure the requirements are actually followed. There was no mission Assurance Manager on the MCO mission. Included ensuring that the AMD file, which is one of the files put out by the propulsion system, met the requirements of the softwares in the interface specification And that did not happen. And so that was sort of the root cause there.

Here's a good example for requirements. This is a little bit older. This is the DC-3. This is an old plane. First flight in December 1935. This was this was really an airplane that kickstarted civil aviation as a commercial service. And so first of all, the DC-3 was based on an earlier evolution of the DC-2. And I first saw this when I worked in St. Louis. There was a little museum at the headquarters. And they had like a, like a nice plaque on the wall with actually a replica of the contract. You know, the original letter that left the DC-3 contract.

And the story is that this-- these requirements-- these are-- this is level 0 requirements, what's written here, was hammered out in a single quote, end quote marathon phone call between Smith and Douglas. So Smith was, at that time, the head of what was today essentially United Airlines. And Douglas, who by the way, is a graduate of this department here at MIT, Donald Douglas, was the head of the Douglas Aircraft Company. And here are the requirements: range about 1,000 miles. So this means that you don't have a trans-continental range, right? If you want transcontinental range in the US, you need about how many miles? Like Boston to San Diego, Seattle to Miami. 3,000? Yeah, reserves. Yeah, 3,000 miles.

So this means that, if you go transcontinental with the DC-3, you've got to refuel twice, basically. But you know, that was OK. Today, we wouldn't-- we probably wouldn't accept it. But that's OK back then. Cruise speed about 150 miles. 20 to 30 passengers, depending on the configuration. Twin engines. And then something that we would say today. That's a fuzzy-- that's a not a well written requirement, because it's fuzzy. But it said, it should be a rugged and economical.

And based on this sort of high level requirements that were pretty clearly defined, the airplane was then designed, built, and very, very successful. And over 10,000 copies of this airplane were built.

So what happened since then is the requirements explosion. This is a chart from a document called the technical multi-disciplinary design optimization white paper. And it's a little dated, but the basic message here is pretty stunning if you think about it. So first, heavier than air flight, 1903, the Wright brothers, right? 1G flight. What is the requirement for 1G flight? Get off the ground. Get off the ground. And stay off the ground for at least a few seconds, right? Not just ballistic. That's the requirement. Get off the ground. Heavier than air flight. Yeah, big success. Big milestone.

Well, pretty clear after not too long, it was well-- we also want to turn. We don't want to just fly straight on the beach and land again, so maneuvering, gust acceleration. There are winds buffeting. You've got to handle those gusts. So that that's important.

And then what were the first airplanes made of? Let's see an EPFL, guys. What's the first airplanes? What materials did they use?

[STATIC]

[INAUDIBLE]

AUDIENCE: Wood.

PROFESSOR: Wood, yeah. What else?

AUDIENCE: Canvas.

PROFESSOR: Canvas.

AUDIENCE: [INAUDIBLE]

PROFESSOR: Exactly. so then very soon after that, maybe in the 20s and 30s, we started using metals, right? Metallic structures. And the big issue there, especially if you're flying close to the ocean and salt water, is corrosion control. Your airplanes can't rust. And so a corrosion control became important. And pressurization, what is that all about?

Higher, faster. So at some point, you need to pressurize the cabin, right? That's a new requirement. The Wright brothers never thought about that. Well, they didn't-- maybe they thought about it. But they certainly didn't have to pressurize their Wright flyer.

So you see you see what's happening is we get greedy. You know, we were excited. We can fly now. But now we want to go higher. We want to stay longer. So we get greedy. And we go to more extreme environments and so forth. And as we do that, it get harvested do and more and more requirements start piling up. And you see a big step here, World War II, right? Handling quality, radar transparency. Radar was invented. So now you want to have airplanes that are not visible on the radar. Fatigue, rough field landing. And then we have another big step here in the 60s and 70s. This was during the Cold War, smart weapons, nuclear, fly by wire, right? Replacing cables with electronic flight controls. And then a lot of the "ilitys" in the last 20 years, reducibility, affordability, portability, et cetera.

So if you actually look at the requirements set for the new generation of airplanes, whether commercial or military, it's overwhelming. I mean, it's thousands and thousands of requirements, because we've gotten greedy. And we've gotten good at it. So we keep adding more requirements. And that's a big issue right now. And we'll talk later in the semester about complexity management. But the key message here is requirements have been growing over time. More and more requirements added as systems grow and in performance and complexity.

So here's some standards. I won't go through those in detail. But people recognize, in the system engineering community, how important it is, these requirements. So in the system engineering handbook, there's two sections, 4.2, which is about technical requirements definition, and then section 6.2, which is about requirements management. So definition means that's the initial definition of the initial requirement. And then management is the process of updating them, adding requirements, modifying them, making sure they're up to date. And there are a couple of appendices.

INCOSE, the International Council of System Engineering has, in the handbook, a whole section. There's even a requirement working group, people who really specialize in this, and then as well in the ISO standard, you have a lot written about requirements. So this is really a big deal.

So the last thing I want to do here in this section is to sort of, at a very high level, communicate to you what requirement really are about. What they are about is, like I said, don't set off in your ship without-- you don't know what port you're sailing to, right? So requirements set goals and constrain the design in the objective space.

So whatever you're designing, you're always going to have two spaces that you're dealing with. The design space, on the one hand, that's the things you can choose as a designer. These are the knobs you can turn, the decisions you can make. And then the objective space, which is the things that, essentially, your customer cares about. I'll give you a quick example here.

So when we use the word shall, which I'll get to in a minute. The English word, shall, means this is essentially a constraint. You must accomplish this. And when you use the word, should, it's more like a goal. It would be nice to do this. But it's not absolutely mandatory. So shall is a hard constraint. Should it is desirable as a goal.

So let me give you this quick example here. So let's say you're designing a house. You're about setting off to design a house. And what would be some requirements for a house. So I wrote four of them here. And I'll map them to the design space and the objective space.

First requirement, the house shall sleep between four and six people. Well, is that in the design space or in the objective space? That's in the objective space, right? Yeah, that's right. So here, we have an axis called occupance, right? So four is the minimum and six is sort of the maximum. So that's the upper bound.

The next one, the total build cost should be less than $550,000. Should be. What is that? Maybe an EPFL.

AUDIENCE: It's a goal.

PROFESSOR: It's a goal. Yup, that's right. And is it in the design space or the objective space?

AUDIENCE: The objective space.

PROFESSOR: That's correct. So I put this as a dashed line, OK, because it's a should. So what I've done here, just by writing these two sentences, these two requirements, is I've essentially carved out a space, right? And I can shape that box. Basically, what these two requirements do is they put a box around in the objective space saying that, whatever house you're going to design, it has to fit within this box.

Then the third one is the house shall have at least at least three bedrooms. What is that? In the design space or objective space? Design space, right? Because defining a bedroom is a design decision, right? And the house should have a fireplace. So you can show the lower bound here, at least three bedrooms. And then the fireplace is a-- it would be nice to have. But you don't absolutely have to have it. So you can sort of draw this-- it's more like a line here, right? Yes, fireplace. But it's a dashed line. And then at least three bedrooms.

So just these four sentences, we've now put a-- we've defined the space that we're going to be designing in. So we have constraints in the design space. And we have constraints in the objective space. And that's fundamentally the role of requirement is to constrain, to give us direction as to what we're going to design.

OK, so let's do our first concept question for today. And that's the following question, do you think there's a fundamental difference in the meaning between the words, requirements, which we've talked about so far, and then the word, specifications, which I haven't mentioned yet. So no, you think they're essentially the same. Yes, there is a difference. Requirements are like the input, and specifications are the output of the design process.

Yes, you think a difference. Specifications include the requirements as a subset. Or you're not sure. So think about this. And then submit your answers. And as far as these URLs are concerned, for the concept questions, I went to all lowercase. Hopefully, that's a little bit easier. So secc3. tiny.cc/secc3. What do you think? Requirements and specifications. The same thing or not?

Who needs more time? Anybody? OK.

OK, so we have nobody thinks they're the same. 62 thirds of you think that requirements are the input, specifications are the output. 9 of you think that requirements are a subset. And then 6% are not sure. So that's good. I agree with that.

So a lot of people use requirements and specifications as the same thing. They're really not. OK, so think of requirements and you haven't actually started to design that. This is your putting constraints in the design space, defining the direction. But we don't actually know what the design will look like. You may have some requirements may say you must use this box, or you must use this sensor, right? And you might actually find some of those in the CanSat. But basically the requirements are the input.

So any-- What are your experiences? When have you heard those terms, requirements or specifications used interchangeably or not interchangeably? Has this come up as an issue for you before? [? Marissa, ?] you're worked at-- you're worked on human spaceflight, right? Space tourism vehicles. So what-- how did you guys do it at Virgin Galactic?

AUDIENCE: Well, I think something that you generally saw was like people will look at sensors. And they would-- there's a specification for a sensor. And they would not necessarily compare the requirements to the specification. So or they would not understand how the two intersected. So I think there was a bit of a misunderstanding of the difference between what the actual requirements were, versus what the specifications were designed to be.

PROFESSOR: The sensors you're talking about would be sensors, I'm assuming, that you would purchase, right? As commercial--

AUDIENCE: Yeah, absolutely. So I think we were buying of the shelf. And so I mean, sometimes the specifications were broader than what we wanted and sometimes they were narrower. But you know, you can do additional testing on it. So there was sometimes a breakdown between understanding really whether the sensor met what we are looking for or it didn't.

PROFESSOR: OK. Good. What about an EPFL. Any experiences? [? Walker, ?] you worked on slip rings, right? Slip rings was one of the big specialties of your company. I'm assuming that you guys had to make that distinction, right? Between requirements and specifications. What are your thoughts at EPFL on this?

GUEST SPEAKER: Well, yes, from outside, we clearly had a separation also on an ether level. But I was wondering here in the class, has anybody seen the difference between specification requirements? Any comments? Maybe some of the PhD students that have been around the block?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yeah, OK, great. So I found here, I, you know, I wanted to sort of find a real example that everybody can relate to very, very easily. So the answer is there is a distinction. They are quite different. But they're-- got to be careful, because you can mix them up.

So requirements specify what the product or system should-- shall or should do, right? Functions it shall perform, how well it should perform these. Also maybe the degree of autonomy, how automated is the system? So what the operators must do, when the system is operating. And also compatibility with other devices.

And then specifications are about how the system is built and how it actually works. So the form, right? What the materials that are used, the dimensions, schematics, blueprints, the details of the user interface. Those are things-- those are all specifications. So I try to look up a very simple consumer product that we all are familiar with, a microwave oven.

Kenmore Elite Countertop 2.2 cubic foot. You go to Sears website. And what's interesting is they actually do a pretty good job. It's actually pretty consistent with what I'm saying here, except they don't talk about requirements. They call it description. So what's listed on their description. And this is, I'm pretty much quoting here, verbatim.

This microwave is large enough to accommodate the big dishes, right? So the idea is this is sort of family style, right? This is not a small microwave just for a frozen meals. This is-- you should be able to put a big casserole in it. And reheat meals and so forth. So the idea that what are the things you're going to use in the microwave. What's the use case, right? CONOPS. CONOPS for the microwave. You know, family, for kids, both parents are working, busy, you know, not time to spend two hours preparing dinner. Therefore, big dishes, a lot of people. It has to be quick, right? That's a CONOPS. And the requirements, this description, is essentially the requirement.

1,200 watts of power to reheat food quickly. So time is important. And then one touch settings for different food types, rice, pizza, frozen meals. And that, what is that? That's automation, right? Rather than having to guess how long and at what power level to reheat, there's some partial automation built in. That's essentially requirements, very user centric. Do you see that?

And then when you look under specification, it says the following things: stainless steel exterior, right? Again, you don't want it to rust. The dimensions, the weight. There's a general warranty one year. The power cord is included. It's very different. Those are things that describe how it's made, the form of it, and so forth. So make sure you keep those separate.

OK, so let me talk about the NASA requirements process. And then we'll talk about challenges.

So getting back to the system engineering engine. That's sort of the heart of the NASA system engineering process. You remember, this is the-- this happens at every level of decomposition. After stakeholder expectations. The second step is technical requirements definition. And there's some pretty strong language in the handbook that this really has to be done well. So the center directors or their designees shall establish and maintain a process to include activities, requirements, guidelines, documentation for the definition of technical requirements.

So I think I've already said this, but I just want to make it clear again, this is kind of-- I like this cartoon. This is Moses, right? Up on thee-- Moses just got the tablets, right? The stone tablets. They-- the 10 commandments. And one of them is, thou shalt not steal. And God says, out of the clouds, no, they're requirements, not goals, right? So what God is saying is that the 10 commandments are, shall, right? And not should. Even though, I think, some of us don't always succeed at that. But that's basically the idea. Shall is a very hard constraint. And that's what you will be judged against. And should is essentially a desirable goal. But the degree of attainment is somewhat flexible.

Why are we doing this? Why are we spending our time writing technical requirements? It's essentially to transform those stakeholder expectations we talked about last time, transform them into measurable technical requirements. Requirements come in different flavors. And I'll mention those flavors. And we express them in these shall statements.

They also provide a basis for agreement among the stakeholders and developers. So you will often find requirements part of contracts, right? Now it becomes serious, you know when the requirements set actually has legal implications. You're legally signing up to develop a system that will meet these requirements. That's pretty serious stuff.

By writing good technical requirements, you can reduce the development effort, because of less rework. So a lot of rework and iterations and confusion is based on missing or poorly written requirement, right? And writing those early and before the design begins is helpful.

A requirements set is also a basis for cost and schedule estimates. And so some of these missions that overran. Do you remember I showed you this chart last time. We analyzed 40 missions, earth and space science missions. Like 20% of them were responsible for 80% of the overruns. A lot of that had to do with unrealistic goals or expectations and requirements.

And then the next point here is that the requirements provide the basis for verification. In other words, the better, the more crisp, the better written your requirements are, the easier it is to test the system and to check whether these-- whether the system is in compliance with the requirements. And then, eventually, the basis for acceptance. Acceptance is also a big deal, right? It basically means, I accept your design. I take ownership of it. And I say, you, as the developer of the system, have done your job properly. And there's a legal transfer of the assets that happens. So facilitates the transfer of the product to the users.

And then even later, if you're going to do it like a version 2 or a block upgrade, a next generation product, it's much easier to do when you have a clear requirements, because it tells you what the original system or the earlier version was able to do or not do.

Graphically. So this is figure 4.0.1. I briefly talked about this last time. We start, essentially, here at the development. We have mission authority. We do the stakeholder expectations. We talked about that last time. And then here, right away, comes a high level requirement. So that means your level 0, level 1 requirement. And then as you try to get more detailed in the requirements, what do you think happens?

Ideally, you want to write all the requirements upfront, right? That would be great. But you try to do that, you hit a wall. Why is that? What's the issue? Why can't we just write all the requirements and then be done with it in one shot? Yes. Go ahead.

AUDIENCE: Well, your system changes over time. And some of the requirements clash. And you can't achieve all of them simultaneously.

PROFESSOR: Yes. so there's two things you mentioned. So one is the sort of-- and I'll talk. This is known as, requirements volatility, like the requirements are changing as you learn more about the problem. And then the other is you detect conflict between requirements. And you have to clean those up. Those are two very valid issues. But it's not quite what I was going for.

AUDIENCE: There's another one, requirements creep, in which your customer levies these new requirements on you, once you've started the process.

PROFESSOR: Yes, so new requirements get added. Hopefully, you get more budget, too. That doesn't always happen, right? So that's another issue. But it's a little different. Let's see, EPFL, you guys. Why don't we write like all four or five levels of requirements all at once? Why can't we do that, typically? Why do you think?

AUDIENCE: Usually, the other requirements aren't clear. They don't exist yet. Later on in the process, they become clear. And you start to realize the details, which at first, you don't.

PROFESSOR: That's-- I think that's pretty close to what I'm looking for. So the issue is you do your level 0, level 1 requirements. And then sort of as you get to level 2, you can't really write that level 2 requirement until you've made some key design decisions. Are we going to use electric propulsion or are we going to use chemical propulsion? You know, that's a huge decision in space system design. And unless you've made that decision, you can't really write lower level requirements, because the fundamental working principles of chemical and electrical propulsion are quite different.

So that's the key issue is that you can only do the high level requirements in a solution neutral space. And then you hit the wall, because you've got to make some key concept technology selection decision. And that's shown here by this red box called functional and logical composition. I don't like that nomenclature a lot. This should really say, system architecture or concept selection, which we'll get into in the next couple weeks.

Once you've chosen a high level architecture and concept, then you say, OK, we're going to go for ion propulsion. Well, then you can write the requirements for the ion propulsion system, which are we through the find in this yellow box here. So this is your design and product structure, derived an allocated requirements at lower levels. You see that? That's the fundament-- all the things you said, requirements creep requirements conflict. All those things are true. But the fundamental reason why we can't write all the requirements upfront, because, at some point, the lower level requirements depend on design decisions made. That's the fundamental issue.

OK, any questions about that point? Yes?

AUDIENCE: It's not totally about that point. But it's still like-- So requirements versus specification. So if you're like buying a component from a vendor, some piece of hardware, and you want identify like the specific locations where it should attach to things, it's more of like a form as opposed to really-- so does that fall into requirements or is that like a specification?

PROFESSOR: Great point. So there are-- what you're describing is a peculiar type of requirement that we call interface requirements. And it's a perfect segue to this next-- you know, if I could only include like I don't five slides in this lecture, this would be one of them. This basically is what are the different flavors of requirements or types of requirements. And there are six of them here.

So first of all, the functional requirements. I think we've sort of talked about those, right? Define the functions that need to be done to accomplish the mission objectives. There's some examples here around thrust vector control. So basically, the idea here is that you have a thrusting system. And it has or you can actually direct the thrust. And in this case, you should control the thrust. You shall provide thrust control around pitch and your axes.

So this statement is a high level functional statement. And it's written in the actor verb, object form, right? So that's sort of the classic, you know, the classic requirement and the functional requirement. And then we have performance requirements. The performance requirements are, in a sense, qualifiers on the functional requirements. So a performance requirement will specify how well the function should be performed, how fast should it fly, how much thrust. You know, this is where you have to actually put numbers in.

So in this case, this thrust vector controller shall gimbal the engine 9 degrees, right? That's the deflection angle, at least 9 degrees, plus or minus 1 degree, degree. 0.1 degree. That's the performance requirement.

Then we have constraints. OK, yeah?

STUDENT: Isn't the performance department saying that the engine has to gimbal 9 degrees, isn't that kind of pushing toward the specification. Or is that? I mean, I guess, is can also be a requirement. But isn't that?

PROFESSOR: Right, in this case, we have a set, and we're not specifying how those 9 degrees are achieved, you know, whether it's through a gimbal or the actual mechanism or how it's not described here. All that's described is the angular [? range, ?] essentially, that this. And that is a performance requirement. But you'll see, there will be some examples of things that are putting limits on the form, which looks more like a specification.

So then we have constraints. Constraints are things on like weight, mass, power, things like that. So constraints are requirements that cannot be traded off with respect that cost, schedule, or performance. So for example, the prospector controller unit module shall weigh less than 120 pounds. So that's a-- that's not, if you think about it, that is not functional, right? The weight of the thrust vector controller shall not be more than 100-- that is not performance. It's not functional requirement, but it's a constraint on the form, essentially. So this looks more like what we would call a specification constraint.

The fourth one is what [? Marissa ?] brought up. This is an interface requirement. OK, so in this case, our thrust vector controller shall interface with the J-2x The J-2x is a very famous engine. And J-2x is a kind of-- this is sort of, this was written during the constellation days of the constellation program at NASA is the idea of a new generation of the J-2 engine. So the idea is that whatever you do, however you design your thrust vector controller, it must be able to interface with the J-2x engine, according to conditions specified in this interface control document. So this is called an interface requirement.

Then the fifth category are environmental requirements. So the TVC shall use [? Biber ?] acoustic shocks and loads according, again, to some environmental document. And by the way, for the CanSat competition, we have that, right? There's two documents. There's the mission guide. And then there's this environmental testing guide.

So what this essentially says is under what conditions shall the performance, and functional-- the functions and performance that are specified in the first two type of requirements, under what conditions shall that be performed, right? And if anything you think about it, designing whether it's a sensor or anything to operate between 0 degrees Celsius and 30 degrees Celsius or between minus 60 and plus 80, is a huge difference, right? You can write that down. But what that actually means to open up the range of environmental conditions, it has a big, big impact on the design. And so this is a big deal in practice. And then, at least in Space Systems design, you know a lot of these environmental requirements are of course, driven by the space environment or the launch environment. If you're designing airplanes, whether you're flying in a small mountainous country like Switzerland or on the ocean off of an aircraft carrier, those are very different environments. And they're going to influence. You have to specify the operating requirement. The bigger the envelope that you make for the environmental requirements, the more complex the system will be.

Also medical-- anybody working on medical devices here? EPFL. Anybody? Medical devices? So in medical devices, it's the same thing. Is this medical device going to be used in a hospital setting, where everything is kind of clean. You have clean power. The nurses, everybody is very well-trained. That's one thing. Or is his medical device going to be used in the field, you know? In Africa, in India, in a rural area, where people are not, maybe not trained, medically and professionally and so forth. Very different. The function may be the same. But the environmental conditions are very different. You have to really specify those.

And then we have-- the last category is kind of the other. It's sort of a catchall. But it can be very important. So this includes a lot of the human factors, reliability and safety requirements. Those are-- and you know, they're listed here as other. And I think it's really important to say that just because they're listed here doesn't mean they're less important. These are often neglected, unfortunately. And it really can hurt you in the long term. So pay attention to those human factors, reliability, and safety requirements. So six types of requirements.

So let's talk about what makes good or acceptable requirements. And there's a distinction here between a single requirement statement and then sets of a requirements. So first of all, requirements should be written in natural language. They should be complete sentences. And each of the requirement statements should be clear and consistent, meaning that it's not a novel or it's not a poem. But it's clear. It's understandable. It's well-written. It's correct. There's no errors in it. It's feasible. Now that's a really tricky one. That's the first thing here that's really tricky, when you think about it.

Feasible means, what? It means this requirement can be satisfied within the laws of physics and state of the art technologies and other project constraints. So why is that a tricky one?

Go ahead.

AUDIENCE: I was going to ask a question regarding that, in terms of how do you deal with a program where you're working on things or you're trying to actually maybe define the state of the art or figure out what is feasible.

PROFESSOR: Right. so if you're basically designing a product or a project or a system that's a repeat of what's already been done, then you can have pretty good confidence, right, that those poles are feasible. But what if you're doing, let's say the-- I think I mentioned this last time. You're going to Europa. And you're going to drill through the ice. And you're going to explore the ocean under the ice in Europa. It's never been done before. You know, can we actually do this? So this is the tricky thing. How ambitious, how ambitious can the requirements be and you still claim feasibility?

And that's also one of the big reasons why programs get in trouble is when they're actually defining requirements that are not really feasible within-- they're definitely way beyond the state of the art. And within the time frame and budget allocated, you can't get there. And many, many programs that run into this problem are the ones where the technologies that you're going to use are not really ready yet. You know, there-- we'll talk later about technology readiness levels scale. You're not really ready yet to do this, but you're going to try anyway. So that's a tricky one, feasibility.

Flexibility is, you know, don't over specify how things should be done. So don't say how it should be satisfied. Without ambiguity. That means if 10 people read this requirement, they should have the same or very similar interpretation. Singular statement, one actor [? verb ?] object. And then the last point here is verifiability. How are you going to check whether or not this requirement will, in fact, satisfy you.

AUDIENCE: With regard to that, the feasibility again for some of these programs I had large, [? overrun ?] schedule, overruns. I mean, would it be almost better in a sense to have some of the requirements be more shalls, like-- or goal. In other words, you know, if you can do this, go for it. But if you're going to run over 10 years, maybe back off a couple of percent.

PROFESSOR: That's right. And what-- we'll talk about this a little bit. But if you-- all of them are shall statements. They're all hard constraints. The objective space that I showed you may, in fact, have 0 feasible space. So really knowing where, what is a hard constraint, what is a hard requirement, and what is flexible, that's very tricky. And that's why it's important to actually not just accept requirements. You know, if you're going to run a project, every requirement, you want to really understand it. And if you think this requirement is infeasible, you have to negotiate. That's where the upfront negotiation becomes really important. Did we lose our EPFL? OK.

GUEST SPEAKER: Hello?

PROFESSOR: Go ahead, please.

GUEST SPEAKER: [INAUDIBLE]

PROFESSOR: Please, go for it.

AUDIENCE: So in 1990, we got the requirements from NASA to build a space [? bioreactor. ?] And all the students will love this. It was the smallest brewery that ever flew in space. And the initial requirements was that all fluid containers had to be solid with double walls. And this meant that we had another solution with bladders. [INAUDIBLE] bladders.

So we had a real fight up front in the proposal phase to demonstrate the compliancy to the requirement of going the cell culture, yeast cell culture, and still being non-compliant, actually, to this requirement of hard double-walled fluid containers. And so this is exactly [? the point, finding these ?] [INAUDIBLE] since the last 15 to 20 years. And this was a breakthrough in technology. And there, you will always have these challenges with the agencies, with your customers. They are [INAUDIBLE] set in their ways. And you have to demonstrate the feasibility to them to make them change the requirement that allows you to change the specification.

PROFESSOR: No, I think that's a great example. So in the end, you were successful to argue for the bladder the bladder solution.

GUEST SPEAKER: Well, the bladder was just how to keep the fresh medium and the used medium, meaning the beer. The technology was to put in [INAUDIBLE] And actually, it needed to have flexibility on the pressure in the reservoir. That's why the argument to put [INAUDIBLE] in space finally won the day, and allowed to relax the other requirements.

PROFESSOR: Yeah, thank you, thank you. A great example there. Any other comments?

AUDIENCE: [COUGHING]

PROFESSOR: OK, so so all this is a pretty long list. And this applies to single requirements. And then there's a set of-- this is really important. Then there's a set of characteristics that we want to see when you look at sets of requirements, groups that require.

Absence of redundancy. This means that each requirement is specified only once, right? You don't want to have redundancy. Redundancy can be good in system design, but not in the requirements. Consistency, using terms. Completeness, this basically means not missing key requirements. And then this idea of absence of conflict. And this is also similar to the feasibility. This is a tricky one.

Requirements can be in tension with each other, particularly their should goals. But they shouldn't be in direct conflict with each other, like you know you shall use aluminum for this unit. And then another requirements says, no metals are allowed to be used in this unit. You can't. That conflict is not solvable. That's a direct contradiction. That's different from having competing requirements or requirements in tension.

So let's do a quick exercise. And then we'll actually take a break, as well. So this is a turn to your partner exercise. And then we'll have a break. And we'll restart in like about seven minutes.

So here's what I'd like you to do. I have three systems here. They're very different in scale and complexity. So A is sticky tape. And I'll tell you this quick story here. I grew up in Switzerland surrounded by farms, you know. And I spent all my time at the farms and you know, big tables, big families. And this is-- they have these things here. This is basically-- these are flies sticking on tape. This is to keep the flies off the dinner table. It's called Mr. sticky tape for trapping flies. Really kind of gross. But I remember it.

B is I just a couple of weeks ago had a test drive and the new I3. This is the BMW small electric city car. Very cool. And then C is something that at EPFL, you guys know very well. This is the Rolex center. It looks kind of like Swiss cheese, when you look from the top. This is the equivalent of W-20 here at MIT. This is the student center. So there's a library there. There's a cafeteria there and so forth.

So what I'd like you to do is turn to your partner, pick one of those three, and come up with one single statement, one good requirement that you think was possibly used in the development of that solution, whichever one you pick, OK? So pick one of those three. And then jointly discuss and write a requirement that you think led to this design, OK?

So take about five minutes. Take a break. And then we'll sample what you came up with.

So let's hear from EPFL first. did anybody do A?

AUDIENCE: We did A.

PROFESSOR: OK, go for it. Speak up.

AUDIENCE: We had four requirements for.

PROFESSOR: OK, go for it.

AUDIENCE: All right. So The tape shows [INAUDIBLE] of the tape should be less than 4 grand. The tape should be able to catch up to 60 insects. And the sticky tape should not be toxic for humans.

PROFESSOR: OK, I like, yeah, the toxicity. That's good. Excellent. Very good. ow who did A here at MIT. Anybody? A? Go for it.

AUDIENCE: We said if greater than 10% of the surface area of the fly contacts the paper, it shall not be able to release itself.

PROFESSOR: Ah, I see. So this is a trapping requirement.

AUDIENCE: Yeah, how effective should it--

PROFESSOR: There are different, very different flies, right? There like these little day flies. And then there's huge horseflies. So which type of fly did this apply to.

AUDIENCE: We said 10% of the surface area of the fly.

PROFESSOR: Oh, any type of fly. Any different fly species.

AUDIENCE: Yeah.

PROFESSOR: OK.

All right, cool. So go ahead. Well, you know the point of this. You get the point of this is, right? A is like the sort of you think trivial system. But once you really start thinking about it, it's pretty tricky, right? Who else wants to, A-- something that hasn't been mentioned yet on A. Sam, go for it. Make sure you push the button.

AUDIENCE: Yeah, for the sticky trap we said that the product shall fit on a store shelf when packaged.

PROFESSOR: OK, so that's kind of packaging, logistics, distribution requirement. Very good. Mike?

AUDIENCE: They kind of already touched on it, but the sticky tape shall not be toxic and allow for-- allow for removal of human skin without bodily harm.

PROFESSOR: OK, great. So that's in the same-- you know, human factors requirement, basically. Yeah. Very good. Anything else on A at EPFL?

AUDIENCE: [INAUDIBLE]

PROFESSOR: OK, so installation requirement. OK. Nobody mentioned the ones I remember as a child are bigger than this. They were really long. So capacity, right? You could sort of have a length or capacity requirement, as well. Good. All right. I think we want to move on from the sticky flies to BMW i3.

So we had a little discussion here during the break. [? Lucy, ?] go for it.

AUDIENCE: So the car shell meet environmental regulations through mechanical or software tweaking with a proposal.

PROFESSOR: Could you guys hear this at EPFL? You've been following the news, right? With Volkswagon.

So what happened? Why did Volks-- the CEO just step down, right? It's kind of a big deal. Trust in the company has eroded. What happened there?

AUDIENCE: So the requirements that were set at the start. So the car shell meet environmental regulations on emissions. This requirement was probably not met. But the realization that it wasn't met was probably too late in the design process, such that it cannot be changed. And so the easy or cheap way to meet environmental regulations, in that case, was to tweak the software, I guess.

PROFESSOR: Right. So by the way this is kind of interesting. But how do you measure environmental compliance of emissions for cars? Do you know how that's done in practice? Do you guys know?

AUDIENCE: So it's having [? the car ?] run on a-- having the engine run, but the car is not rolling. And there's, behind the [FRENCH] the exhaust. Behind the exhaust, you measure whatever is emitted.

PROFESSOR: And it's like another rolling carpet, right? It's a dynamo, basically. And the way they do it is they have so-called drive cycles, like the [? FT6 ?] drive cycle. You know, the highway, the city cycle, which was a lot of on and off. And those drive cycles are actually different in Europe and the US and other countries. Every country has different drive cycles. So that's-- and what is a drive cycle, essentially? What word would we, as systems engineers, what word would we put on it?

AUDIENCE: CONOPS.

PROFESSOR: A drive cycle is just an, over time, accelerations velocity of the car, right?

STUDENT: It's a CONOPS.

PROFESSOR: It's a CONOPS A drive cycle is a CONOPS. And so for diesel engines, this issue that happened with Volkswagen, it's only an issue with the diesel, the TDI engines, right, not the gasoline engine.

So the issue is that the CONOPS, the drive cycle, in the US with TDI engines, they couldn't meet it. So they came up with this trick, right? Whereas in Europe, they didn't have to do that, because the European drive cycle, there's a lot more diesel engines use in European cars. They didn't have to do that in Europe, because the CONOPS, the drive cycle it's used for checking the environmental compliance is different in Europe. So great.

Anybody else on the I-3. Now the I3 is interesting, because that's actually an electric car, right? And so it doesn't have an engine, a diesel engine or a combustion engine, except if you get the range extender. So it's all electric, but you can actually, as an option get the range extender, which does use fuel. OK, EPFL, the i3, any other requirements there?

AUDIENCE: So we have four [INAUDIBLE]

PROFESSOR: Yeah.

AUDIENCE: As a third one, the car shall carry [INAUDIBLE]

PROFESSOR: OK, so recharging. The third requirement is interesting. You said of average build, right? That's the word you use.

AUDIENCE: Yes.

PROFESSOR: Now that's great. That's a human factors requirement. Now of average build is a little fuzzy, right? How would you make that more verifiable?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Yeah, so there's actually people that, you know, the size distribution, male, female, you know, weight.

AUDIENCE: [INAUDIBLE]

PROFESSOR: And [INAUDIBLE] the metric measurements are actually being recorded and updated. People are getting taller around the world. People are getting heavier. We know that, right? So the way you would write that third requirement to be verifiable is that the car shall accommodate for passengers in a, say, P10 male or P5 female and a P90 male, right? And if you write it that way, then you know the actual weight and dimensions of the human body can actually be traced to a database that's pretty well known. So then instead of saying average build, you say it's a PT50, P50 female. And that's something that's very verifiable. Does that make sense?

AUDIENCE: Yes.

PROFESSOR: OK, great. So what about the last one? So by the way, here at MIT, anybody who's been to EPFL? Been to the-- [INAUDIBLE] He has a Master's from there. [INAUDIBLE] been there? This is a very cool building. It's pretty unusual. I mean, we have some very unusual building on campus, too, here. But if you get a chance, it's pretty. Unusual so let's hear from you guys. So what do you think was a requirement for the [? Rolex ?] center design?

AUDIENCE: The building should be a recognizable structure, that would be memorable for EPSL.

PROFESSOR: Ah, interesting. A recognizable structure that should be memorable. So that's-- I think I know where you're going with this. But it's a little-- you know, recognizable. Every building is recognizable in a sense, right? I think-- I like it. I know where you're going with this. But it's a little fuzzy still, right?

So why do you think it has holes in it? Why do you think the holes?

AUDIENCE: [INAUDIBLE] unique might be a better word. [INAUDIBLE]

PROFESSOR: Yeah. So that's really. The holes are, in some sense, inefficient, right? Because you're putting holes in the middle of a building. But they provide natural lighting. And there's a symbolism there, right? There's a symbolism, the Swiss cheese symbolism. So if the holes, in a sense, have at least two functions, right? That's good. I like that. That's very good. What else? Another example of a requirement for the [? Rolex ?] center?

AUDIENCE: You're taking and making [INAUDIBLE] category. [INAUDIBLE]

PROFESSOR: Good. No, that was great. So you have functional requirement, you had interface requirements. You had a lot of those six categories we talked about, right? Good. Excellent. Yeah? Go ahead.

AUDIENCE: So how do you write a requirement about something aesthetic in a way that's not fuzzy or is there a way to do that?

PROFESSOR: That's a great question. I'm not sure I'm the best person to answer that. What I can't tell you is, in the automotive world, the aesthetics of automobiles and how people judge whether a car is beautiful or appealing, that's actually moved from being kind of just a very fuzzy thing to quite measurable. You know, there's different shapes and then the building blocks of shapes and streamlines. Those things are-- that's really a science today.

And eventually, you know, there's ratings. These are people rate vehicles for aesthetics and so forth. And usually, it's a five point scale, like a Likert scale, like JD Power and associates is a very well-known marketing firm. And so they'll say, this looks-- this will prob-- they can actually, at this point, you can show them a-- not a sketch, but a model. And based on past data and information, they'll tell you, this car will probably score between a 4.2 and 4.4 on the 5 point aesthetic scale, JD Power scale. It's really pretty remarkable.

But I guess the bigger point is there's some things that really delight us, that have an artistic quality and surprise us in aesthetic quality. And it's true. That is one of the tensions is yet-- system engineering should-- you know, this very precise. Write it down. Make sure there is no fuzziness there or as little as possible. And then on the other hand, we want delightful, surprising things that have an artistic nature to them. And that is absolutely a tension. And we just acknowledge that.

OK, any questions? Comments before we move on?

AUDIENCE: I had a question about--?

PROFESSOR: Hang on. Just one second. OK, go ahead. At EPFL?

AUDIENCE: And to which extent can we refer [? to norms ?] and [? loads ?] in the requirements?

PROFESSOR: Yeah, you should do that. So compliance, being compliant with standards and just to make clear, standards and regulations are not the same, right? A standard, like an IEEE standard or an ISO standard, it's not a law. It's not a legal thing. It's-- a standard it is something that maybe a group of companies or a group of organizations have agreed to. This is how we will do it, right? The IEEE Wi-Fi standard, what is it? 802.11g. And that's not a law. That's a standard. And if you're going to be 802.11g compliant, you write that in the requirements.

The environmental emissions standards we talked about, those are actually laws. If you are going to sell a vehicle in country x, it has to comply legally with these regulations. But you're absolutely right, if you need to comply with these things, it needs to be part of the requirements set, because otherwise, it will not happen just automatically. [? Weston? ?]

AUDIENCE: Yeah, it's kind of along that vein. If you have to comply, say, with ADA, instead you have to be called out, specifically, or do you say must comply with building codes or building laws? And it's sort of a blanket requirement?

PROFESSOR: No, you have to be specific, because building codes are, first of all, a lot of these are local, as well. And it's very chaotic. So and some of these might be conflicting. So you should be as specific as possible.

OK, let's move on here. My sense is you're getting it just for what requirements are. And why they're important.

So let me talk briefly about requirements, decomposition, allocation, and validation. This is a figure from the NASA handbook. And basically, what it talks about is the high level system functional requirements are broken down into the performance requirements. And then as you make design decisions, as you decompose your system into different subsystems, each subsystem will have its own functional and performance requirements. And then the important thing is the difference between allocated and derived requirements.

So allocated requirements are requirements that you choose to allocate. And then derived requirements are calculated based on the dependent requirements, based on the allocated requirements. And I'll get into this in a minute. So requirements are hierarchical. We talked about this. Functional performance requirements are allocated. And then from these, we can further decompose and derive requirements. And then the total set of these requirements needs to be verified and then validated against the stakeholder expectations.

So let me briefly talk about requirements margins management. So because you don't know everything upfront, there's uncertainty. We typically build in reserves into our requirements. And those are called margins. So we put in margins for mass, power, maybe memory in computer systems. So margins are essentially reserves that are not allocated to particular subsystems, but are controlled by the project managers or at a higher level.

So the idea is that you write the requirement in a way that is a little bit more stringent than it really needs to be. And then by being more stringent, you've built some reserve into the system to handle unexpected things.

So I'll just give you the example with mass growth. You know, this is very typical in aerospace vehicles. Mass growth can range between, here, 10 to 60%. I'll show you some examples. And a lot of it depends on the novelty of the project. So a typical guideline, specifically for mass margins, is about 30% reserve at the SRR, 20% PDR, 10% CPR, and keep about 5% right before you operate the system. IOC is initial operating capability.

So here's some historical data. This is for manned or crude vehicles, starting with Mercury, Gemini, Apollo, Skylab, and then the shuttle. You can see the mass growth from the concept stage, which is phase A or prephase A, all the way to operations, you know, between 10 and 60%. The new Orion spacecraft is not yet included on this chart.

So what you do is you essentially write the requirement, knowing that this growth will happen during the design process. That's the basic idea of margins. And then the next thing you get is that you monitor the satisfaction of the requirements during the design process.

And so let's say you have, in a performance requirement, you say that the system shall not emit more than x number of [? NOX ?] emissions, or the systems shall not be heavier than such and such. That's on the y-axis here. That's your technical performance measure. And then you move through time and, you know, as the design gets more detailed, usually it gets heavier. You add more things. And you monitor that. And then you have your final current estimate of where you will be at the end of the project. And as you do that, your limits, your reserves, your margins, and here there is an upper and lower margin shown, is going to be narrowing down.

I'll give you a quick example. When I was at McDonnell Douglas, the F18 EF version, the Super Hornet, was being developed. And the key thing there was GTOW, GTOW, gross take off weight. It's basically the weight of the plane with crew, fuel, any payloads all in. The gross take off weight of the airplane, very important, because it determines the range. It determines a lot of things. And the number, not to exceed number, which is shown here on this chart as the maximum contract or allocated requirements threshold, was contractually specified. Not only that, but there were penalties, financial penalties, associated with every kilogram that you would be over.

So this was a big deal. And they had a huge the wall chart in the hallway, the main hallway, of the engineering building. And every day somebody would actually manually update that day's best estimate of what the gross take off weight of the airplane would be. And it's like a Brownian motion thing. And as soon as you hit some critical threshold, you would see there'd be people with would come together and say, we've got to take weight out of the airplane somehow, again, right? And then basically try to get the design to comply with that requirement as you move through the design process. You can't do that with too many of the requirements. But the most important technical performance measures, that's what you do. It's a big deal.

So here's the flow chart, basically, for requirements. I'm not going to go through this in detail. But the basic idea is the inputs come from the stakeholder expectations, the stakeholder work we talked about a lot last time. You go through the requirements definition process and outcomes invalidated set of technical requirements, measures of performance that you can measure, and then these technical performance measures that you can then track and validate against later.

Here's a question that I often get asked. Well, OK, so we write these requirements. You guys write them on paper or your tablet. But how do these requirements actually get recorded, right? And managed. So I would say there's sort of two-- there's the low cost, and then there's sort of the professional version of doing this.

The easy way to do it is you just write them, you capture them in a document. So that means Microsoft Word, Excel, Google Docs, just a document, a well written, organized document. And then you capture and revise your requirements. And the one thing I strongly recommend is using hyperlinks to link requirements. And this is the idea that every requirement has to be linked to some other requirement. So if it's a low level requirement, you want to ask, well, where does that come from? Why did we write that requirement? Well, it has a parent requirement. You want to have a hyperlink there to get you from one to the other. And I have a little example for that.

And I think this is OK for smaller projects, where you have dozens or a few hundred requirements, right? And so here's my rule of thumb for this. Do you remember the magic number 7? So we talked last time, where does the world of really complex systems start. And the argument was, well, if you need more than three levels of the decomposition. What does that mean? Well, 7 plus, minus 2 to the third power is somewhere between 125 and 729, right? So if you're sort of in that world or fewer, right? If a few hundred, a few dozen, or few hundred requirements kind of what we have in CanSat. It's OK to do it this way. It's still going to be a lot of requirements. And you have to do a good job. But you can do it that way.

If you have more than that, and that typically means more than 1,000 requirements to handle, and there are big projects that have 5,000, 10,000 requirements, there is no way you can manage that effectively in a kind of document based way. So what you need then is a database. You basically capture the requirements in a relational database where that allows you to link each requirement to other requirements.

And so this is not meant as an advertisement, but one of the most heavily used requirements tools out there is called, DOORS. And this was relatively recently bought by IBM. This was a separate company. It was bought by IBM and included it in a suite of software tools called IBM Rational for System Development. And so DOORS allows you. It's a database, relational database, that allows you to write requirements, share requirements. And the latest version of DOORS is actually web based, so you can have people in India, in Europe, in the US, you're co-developing a system. You can all write requirements and manage them on this common database, right? Because if you have a document, very quickly it's going to be confusing, as what's the latest version, who has the latest update. Version management becomes a nightmare.

So just so you're aware of this. We're not going to be using DOORS in this class. We'll just do it document based. But you know the rule of thumb here is, more than 1,000 requirements, you've got to go to some professional solution.

So here's a very sort of trivial example of hierarchical requirements with links. Requirement one, the systems shall fit into a volume not exceeding one cubic meter. And then we have sub-requirements. The system's width shall be between 0.5 and 1 meter. The height, the depth. The system shall be made entirely from aluminum 60/60 alloy. A sub-requirement here is the system shall not contain any internal voids or cavities. Requirement three, the shape of the system must be a cube. And then a sub-requirement, the angles between the sides shall be 90 degrees plus or minus 1 degree.

And so I did this here just in the slides. But if you click on this, it'll actually transport you into another requirement. This is the requirement four. The mass of the system shall not exceed 2,700 kilograms, right? Aluminum has a density of about 2,700 kilograms per cubic meter. So I click back, it transports me back to the earlier requirement. So the fact that it's made of aluminum and the volume cannot exceed one cubic meter, then this requirement four, the mass shall not exceed. It's not an independent requirement. It's an dependent or derived requirement based on the first two. And therefore, they're linked. Do you see how this works?

And to really manage requirements well and then link them, use these hyperlinks. It's very, very effective.

So what would be an object that satisfies these requirements? So here's our one cubic meter envelope. So an aluminum cube with a side length of 60 centimeters this volume and this mass will satisfy the requirements. It's not the only thing. There's a lot of other geometries that would satisfy it. But this particular instantiation would satisfy these requirements. So that's the idea hierarchical requirements, linking them to hyperlinks.

OK, so let me talk briefly about the challenges now of requirements definition. And there's-- this is not easy. There's a lot of challenges. The first one is requirements allocation. You know, there's composition and flowing requirements to the lower levels. And then the idea is that, whatever requirements you derive at a lower level, if you satisfy the lower level requirements, that should guarantee that you then automatically satisfy a higher level requirements from which the lower level requirements were derived. That's the basic idea of requirements allocation.

And so the way you can think of this, graphically, is we start out at the high level. Here's our stakeholders. Stakeholder needs requirements. This is level 0. You set the system boundary. What's the lifecycle? And then we apply it. That's our first application. And then we apply it by decomposing the system into its function.

And so that's the second application. And then you say, well, how do these-- how will these companies be carried out? So we define subsystems. And essentially, then, the subsystem requirements are derived from the functional requirements. And you put numbers, key system performance parameters, behind these. And depending on the complexity of the system, you may have to go multiple layers down. But that's the basic requirements allocation process.

And I think I said this already. It's difficult to do and stay solution neutral the deeper you go into this.

Let me briefly mention, this will not be, we will not sort of test you on this. But I want to make you aware of this. A methodology called ISO performance. And that was actually the topic of my dissertation. How do you allocate requirements to lower level parameters in systems, when a higher level requirement is defined, is clear.

So the idea is that you have a vector of desired performance requirements. And you want to understand, first of all, you want to understand are those requirements feasible. And if they are feasible, it usually means there is more than one way that these requirements could be satisfied, that the higher level requirements could be satisfied. So find different non-unique feasible combinations to satisfy the high level requirements. And so this is one of the readings. Very quickly, we have our design space. We have our objective space. And then we have this cost risk objective space.

And so the idea is that, in the performance space, you have the shall statements. You shall perform at that level. And that's this point here. That's your performance target. And if it's in this gray area, it's actually feasible. And it usually means there's more than one way to do it. So if you can then map backwards to the design space, these points here are all ISO performance, meaning they all satisfy and provide this level of performance. But they do it in different ways. And then in order to select the final design or final requirements set, you want to look at other objectives. This is where the should statements come in. And these are typically cost and risk related objectives.

So let me give you, this sounds pretty abstract and theoretical. Let me give you a very specific example. This was my case study, the Space Telescope. This is the actually this has now become the James Webb Space Telescope. Hasn't launched yet. This is 20 years ago. We've been working on the James Webb for a long time. But if it works, it will be exciting, because it's going to get us very, very close to the Big Bang, right? That's what James Webb will look in the highly redshifted infrared to really see the formation of the earliest proto-galaxies in the universe. So if this works, it will be very exciting.

So here's, essentially, a model of the spacecraft. This is a precursor to James Webb. And the big thing you need to do is you need to point and be very, very stable for a long time to take in these very, very faint images. So we have wavefront error phasing requirements. And then we have these pointing requirements. You've got to a point very stable and in order to achieve this optical pointing performance, we have the structure of the spacecraft. We have reaction wheels. We have controllers. And there is noise, different kinds of noise sources that are trying to basically prevent us from pointing with this precision.

So very quickly, this is what this Nexus precursor spacecraft looks like in the deploy configuration, here on the upper right in the stowed configuration. Initial performance assessment. So this is really trying to define the requirements for deriving the requirements for the structure, for the optics, for the controller, knowing that this is the point. If you want to have this science happen, you've got a point with that precision. That is known. So let's flow that down.

So if we look at the pointing, for example, here on the right, this is kind of fuzzy furball. What does that represent? That's a time domain simulation of the centroid of the image, right? The telescope is now observing a part of the sky, trying to get these early proto-galaxies. And it's flexible. It's a very lightweight telescope. It has these reaction wheels are turning, trying to keep it stable. You have electronic sources of noise.

So the blue furball here predicts that we would not meet the requirements. This is some initial design, some initial set of requirements. We need to find a way to get down to the-- this is about 15 microns-- [? root mean ?] square error. So it's about three times worse than what we need to achieve. And the requirement is we need to get down to 5 microns of pointing or jitter precision.

So how do we do this? So what ISO performance does is it looks at the sensitivities in the system. What are all the things that influence the pointing performance of the telescope. And you can see here. These are the two key performance measures on the left, wavefront error, and then line of sight. And the bars essentially tell you what are the really sensitive parameters that drive performance. So disturbance parameters, planned structural parameters, optics, and then controls. So it's really multi-disciplinary.

And what's neat about it is you can, if you know these sensitivities, you can essentially calculate the Jacobian a matrix. The Jacobian matrix is essentially the partial derivatives of your performance, which is your higher level requirement, with respect to these lower level parameters or requirements. And you can then find, using this essentially the null space of the Jacobian matrix, that will tell you how do you move in that space to keep the performance fixed at the requirement level.

Let me explain this just using two parameters. And this was a big deal in the Hubble Space Telescope. So the two parameters I'm going to use are KR ISO and UD, the dynamic wheel and balance. So these spacecrafts have reaction wheels that are turning to point the spacecraft and counteract any momentum that you get from, for example, from solar pressure. So UD is the amount of imbalance that you have in these wheels. If you have imbalance, that will cause chatter. It will cost torques, which will cause that jitter. And then KR ISO is the vibration isolation of the reaction wheel assembly. How stiff or soft is the vibration isolation.

And what you can see in this clock is that, our initial design-- this is based on a simulation-- does not satisfy a higher level requirement. In order to satisfy that, we need to go down to the blue line, labelled with 5 microns appointing precision. And what you can see is that there are different ways to achieve that 5 microns.

We could go over here. This is labeled as HST. That's what Hubble Space Telescope is. The Hubble Space Telescope basically went for ultra, ultra, ultra quiet reaction wheels, very, very low dynamic imbalance. So [? Marissa, ?] you talked about buying sensors from a supplier and putting them in. These reactors, there's companies that really specialize in making reaction wheels and things like that.

So if you're going to go for this point here. Let me point that out again, so you guys can see it at EPFL. If you're going to go, if you're going to derive and allocate this requirement, it means you put a lot of pressure on your supplier to achieve that level of dynamic wheel imbalance. If they can do it, that's great. But it could be very expensive. But it makes your job easier as the spacecraft integrator.

Or you can go straight down to this point here called spec. So you're essentially tolerating a noisier reaction wheel. But then you need a very, very soft, very capable isolation stage, very soft isolator. What's the disadvantage of having very soft isolation? What do you think is the big issue?

AUDIENCE: [INAUDIBLE]

PROFESSOR: So the launch. You probably have to lock it down during launch. You have a big displacements. You need, if you have a very soft isolation stage, you need volume, because the isolator is going to move a lot, right? And you may not have that volume. Or you could do a little bit of each.

So this test point here. We're going to have quieter wheels, but not ultra quiet. And we're going to have a soft isolator. But we have the right combination of the two. And we're going to go for that point there. So now if you go for that point, you're going to satisfy your pointing requirement, bu you've allocate at the lower level requirement, in this case, to the isolation stage and the dynamic wheel imbalance in a kind of balanced way. So everybody's job is roughly equally difficult. That's the sort of idea here. Yes?

AUDIENCE: Does this model assume that all of these variables are independent and, I guess, kind of continuous in terms of you can incrementally work better?

PROFESSOR: So they're not independent in the sense that they're coupled through the performance of the system. But you can choose-- the assumption is you can choose these independently. But they're coupled, because you need to achieve that requirement. And yes, in this case, they're continuous. But you can think of this as a discrete, as well.

So for example, the reaction wheels could be, you're picking from a catalog, in which case now you're really picking specifications for components that are off the shelf.

OK, so let me-- I'm going to skip here for time. So basically, following this ISO performance process, you can go from some initial requirements, derived requirements that are infeasible, to, in blue here I'm showing you the results of this. It's very close to 5. There's some numerical tolerance here. But by but essentially rebalancing the requirements within the system, we're achieving a higher level requirement. But in a way that this [INAUDIBLE] sort of the challenge equitably across sub-systems.

Does that make sense? At EPFL, did you guys follow this? I know this was a bit fast. But that's the basic idea of ISO performance. Any questions on your side?

AUDIENCE: [INAUDIBLE]

AUDIENCE: Maybe I have one. So given that you have [? distance ?] requirements and you probably have [INAUDIBLE] for each one, is there a way that we would integrate all those [? lines, ?] that you could see the [INAUDIBLE] for those? If you don't mind, maybe it's a little bit [? softer, ?] but maybe there's another requirement where [INAUDIBLE]

PROFESSOR: [INAUDIBLE]

AUDIENCE: We don't hear you.

PROFESSOR: Sorry, guys, ran out of battery here. Can you hear me again? Can you hear me? Yeah? So the question was, well if you have multiple of these performance, ISO performance lines, that's exactly right. If you have a vector of higher level requirements, you're going to have ISO performance surfaces. And so it gets, you know, the more of these you have to satisfy at the same time, the more complex this gets. But we can handle that computationally. So you have to do some simulation and computation upfront to make sure you pick. These requirements are not just wild guesses. They're actually based on some calculation and simulation. OK?

All right. So let's do this. Let's do this in a kind of simplified way. This is going to be our last concept question for today. Here's the higher level requirement is a balloon. Those of you that have done unified engineering here, you've done this, right? We did this in unified. A balloon shall lift the payload of 1,000 kilograms, which includes its own mass.

You can use either helium, which has a density of 0.2 kilograms per cubic meters. This is in standard atmosphere conditions. Or hydrogen, 0.1. I've rounded these numbers, as a lift gas. The standard density of air is 1.3. Which of the following requirements is infeasible? A, the balloon shall have a radius of 6.1 meters and the balloon shall use 99.9 percent pure helium. B, a radius of 5.9 and the balloon shall use 99.99 point percent hydrogen as a lift gas. And then 5.9 meters helium, 6.1 meters hydrogen, all these requirements actually are OK. Or none of these requirements are feasible.

OK, so I'll give you I'll give you three minutes to try and figure this out. And then I'll show you the solution. Don't cheat. Don't go to the next chart.

So think about this the high level requirement is the balloon shall lift 1,000 kilograms, including its own mass. And we're trying to allocate lower level requirements for the size of the balloon and the gas that we're going to use. So try to figure this out.

All right, I think we probably need-- who needs more time for this?

OK, so we're going to leave this as a cliffhanger, OK? We'll do the solution next time. And take a little time to figure this out. Try not to look at the next slide, OK?

So I'm going to skip this just for time. And I want to talk briefly about the SRR, what it is, and then kick off assignment 2. So SRR. So the idea is you've really been working on these requirements hard, you know, as a team, with your customers. And this takes-- this is not a quick thing. This is not just two or three days. Typically, writing your level 0, level 1 requirements is a process that takes weeks, months, sometimes at least a year. But typically, it's on the order of three to six months in many projects, roughly.

And once you have your high level requirements, SRR is the milestone where you review those, right? So here's an example. So it's a social. It's a peer review process. And the main goal of SBAR is to vet the requirements as they were written to see if you have any missing, misstated, redundant, or otherwise unsatisfactory requirements.

This is a picture from JPL, from MSL, actually, mission. And I want to just point out a good friend of mine who is a graduate here of our department, Richard Kornfeld. [? Voelker ?] knows him, met him last summer. He's been working on three different Mars missions over the last decade. And we're going to hear from Richard later this semester about verification of requirements, because he verified the entry descent and landing requirements for MSL.

So here's essentially what it says in the NASA handbook. SRR happens during phase A, before you go into phase B. And you need at least your top two level requirements written before you can do the SRR.

I do want to mention the second reading post reading. This is about requirements volatility. So just because you past the SRR doesn't mean that the requirements are completely frozen. First of all, you don't have a lot of lower level requirements yet, right? Those are going to be added post SRR. and even some of the-- you pointed out, some of you, the requirements creep and these things. So this second reading is about requirements volatility, which you can actually quantify. What are the sources of requirements volatility? That's what this figure shows you. And then what is the impact of requirements volatility?

What is the impact of requirements volatility on a project? So specifically, on the number of system requirements, on rework, on project schedule, and so forth. Really, it's a very, very recent paper on requirements volatility.

So let me just kick off assignment 2. And then we'll be done. So assignment 2 goes out today and is due in two weeks. So that's October 9. And it's very focused on requirements. Basically, what I want you-- and it's the same teams that you're in for all the assignments. Is essentially, first of all, first task is find a project or program where poorly written or managed requirements were a major problem. So as a team, discuss an example and discuss that example as a team.

The second task is look at those CanSat 2016 requirements, those 47 base requirements and analyze those critically. So that means are they feasible, are they well-written. Classify them. What type of requirement is it? Is it an interface requirement? Is it a performance requirement? And then figure out whether there's a hierarchy there. And basically, then, from that, generate your own set of requirements for the CanSat competition. So you're going to either rewrite or organize these requirements in a way that's better and that works for you as a team.

And then the fourth requirement is to figure out where you want your margins, where do you think you need to have reserves and margins in these requirements. So it's very much looking and reorganizing and critically analyzing the CanSat 2016 requirements set.

So let me summarize. Good requirements are really essential. It's the starting point for system design, system engineering. It is a challenging thing, especially the flow down is challenging. There are some methods and commercial tools for doing formal requirements management. So I mentioned ISO performance. And then I mentioned DOORS, these commercial tools.

And then the last point here is just because you passed SRR, and you have high level requirements approved, doesn't mean that's the end of the story. The requirements will continue to be updated, refined. But you have to be you have to be very disciplined to keep that in check. If your requirements volatility is too high, then bad things can happen to your project. And I really recommend the reading on requirements volatility, some very, very recent data on that, OK?

So that's the end of the session for today. I'm going to be online on the WebEx in about 10 minutes. If any of you have questions, or any comments, or you want to dive into this further. I know this is not a great time for you at EPFL, because it's Friday night for you guys. So I think what we're going to do, we'll figure this out with [? Leighs ?] and [? Johana. ?] We'll see if we want to do a separate time, a different time during the week for the office hours. I'm happy to do this at a better time, if this doesn't work for you guys, because it's happy hour time for you.

So anyway. OK, so great. Have a great week. And we'll see you next Friday.