Defining Scope

Table of Contents

- Activity: Add Use Cases from Stakeholders and Internal Sources
- Watch: Delving to Expand Use Cases
- Watch: Recognizing When You Have a "Good Enough" Set of Use Cases
- Activity: Refine Your Use Cases
- Discussion: Evaluate Your Use Case List
- Module Wrap-up: Identifying Key Scenarios

Module Introduction: Aligning Your Scope to Stakeholder Expectations

- Watch: Organizing and Communicating Use Cases with Use Case Diagrams
- Tool: Use Case Diagram Template
- Watch: Adding Uses Cases into a Use Case Diagram
- Activity: Populate Your Use Case Diagram
- Watch: Understanding Use Case Connection Types
- Watch: Selecting Use Case Connection Types
- Activity: Make Connections Between Use Cases
- Watch: Adding Secondary Actors to Your Use Case Diagram
- Watch: Formatting Your Use Case Diagram
- Activity: Completing Your Use Case Diagram
- Discussion: Review and Refine Scope
- Watch: Using Scope Trees
- Tool: Scope Tree Template
- Watch: Moving Forward with Your System Design
- Assignment: Course Project, Part Two - Create a Use Case List, Use Case Diagram, and Scope Tree
- Module Wrap-up: Aligning Your Scope to Stakeholder Expectations
- Course Tools and Miscellaneous Documents

Welcome to Your Course

Video Transcript

You just got the opportunity to design for a brand new project. It's really exciting. It's something that you always wanted to make a real difference in. But how do you start? There are so many different aspects to figure out before you really begin to design your system.

Well, that's whats this course is all about. We're trying to help you to really determine what is it that your system is responsible for, and what are all the other things that your system may need to interact with in order to make that system really be a success. That's part of the first part of defining your scope, and we're going to introduce you to a tool called the context diagram to help you figure out what those things are, and what are the interactions between these things that you need to be able to figure out.

Then we're gonna take it a step further to really think about all the different kinds of scenarios that your system might need to do in order to meet that project's needs. This is going to get into some use case analysis overall, and use case diagrams. But overall between these two tools, it'll give you a great start for being able to define the scope of your project and really define what your system has to be.

I can't wait to see what we're gonna be able to create together using these tools.

Course Description

How do you take a fresh new idea for a product, service, or whatever your system might be, and define that idea so others understand? To begin this first stage of design, you need to define the scope of your system.

In this course, as you explore and define your system's scope, you will also begin to identify what your system is responsible for. And you will also explain what your system must interact with in order to be a success.

For the first part of defining your scope, you will use a graphical tool called a context diagram to identify what elements interact with your system. For the second part, you will consider all the different kinds of scenarios that your system might need to

handle in order to meet the project's needs. In determining these scenarios, you will conduct some initial use case analysis, create a use case diagram, and design a scope tree. Once you have worked with these tools, you will have a solid start to your system design because you will understand the scope of your project, and you will have a beginning definition of your system.

What you'll do

- Define interrelationships between your system and other factors by creating a context diagram
- Capture in use cases the set of functionalities that any system must have to meet the needs of the challenge you are addressing
- Create a shareable use case diagram that organizes the relationships between use cases
- Document all tasks and sub-deliverables necessary to produce a project end deliverable

Faculty Author

David R. Schneider

Director of MEng Studies in Systems Engineering

David R. Schneider earned his Masters and PhD from Cornell University in Mechanical Engineering with a concentration in Controls & Dynamics in 2007. David has taught at both Cornell and Columbia University. David Schneider's research has traditionally focused on the realm of NP-Hard Computer Science Problems and Controls for Robotic Systems in centralized and decentralized as well as autonomous and semi-autonomous systems. His most prominent research is his creation of the G*TA (G-Star-T-A) task allocation algorithm and his work as Program Manager of the Cornell RoboFlag program, with notable applications including AFRL UAV controls and NASA/NOAA unmanned boat designs.

With a strong focus on education, David's endeavors have included the creation of the Cornell Cup, Innovative Embedded Design National Competition; leading Cornell

Cornell Engineering Cornell University	University Sustainable Design (CUSD); and the broader impacts video game creation for the NSF Expeditions in Computing Grant on Computational Sustainability. David has led the efforts to make Cornell the first university to officially partner with Make: and is a leader in the Higher Education Maker Alliance working with the White House Office of Science and Technology Policy.

Back to Table of Contents

Read: Preview Your Course Project

Throughout this course, you will practice applying the teachings of each module in a multi-part course project. You can review all relevant information on the **Grades** page of this course.

Since each module builds on the previous ones, it is essential that you work through the course in the order it appears. At the end of the course, you will submit your completed project for facilitator review and grading.

☆ Key Points

You will apply the content from this course in a multi-part project.

You must submit all prior assignments to access the final project part.

Course Project Parts

Preview the course project below. As you work through the course, reflect on how this relates to the course content and to your experience.

Note: Though your work will only be seen by eCornell, you should take care to obscure any information you feel might be of a sensitive or confidential nature.

— Course Project, Part One — Create a Context Diagram

In this part of the course project, you will create a context diagram for a system you want to develop using the Context Diagram Template.

It is strongly recommended that you complete the assignment based on a system of your own choice because you are more likely to benefit personally and professionally by working on such a project. However, as an option, a project setup is available for you to use as a basis of your project work.

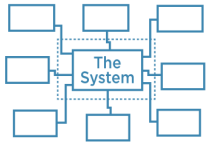
— Course Project, Part Two — Create a Use Case List, Use Case Diagram, and Scope Tree

In this part of the course project, you will use your previous work on the context diagram and use case list to build a use case diagram. You will also

begin to build out a scope tree.

Back to Table of Contents

Module Introduction: Describing Your System in Terms of Interrelationships



You have identified an important need. How do you discover what any solution would have to do to address that need? And how can you present a functional scope for your solution where you indicate how it will operate and how it must interact with other factors outside of your control?

In this module, you will develop a context diagram which begins the process of exploring the interrelationships between your solution and other factors. From your diagram, you will begin to define the functions performed by your system.

Back to Table of Contents

Read: Designing the Right Solution

Amazing! You've just come up the with most exciting and brilliant idea for a new invention. You immediately rush to write down all your ideas and then begin designing it right away. Your passion drives you and you work to make your first components. Weeks or even months later you discover some problems with the idea; somehow those first components aren't working quite the way you thought. But you know it's going to be awesome. So you focus on solving these components' problems and push forward until you're able to make it work. Finally you're able to present your invention. Maybe it isn't quite what you envisioned, but you're still very proud of your accomplishment and ready to show it off to potential users and stakeholders.

☆ Key Points

During the design process it's important to ask question such as:

- Is this really the best way to solve this problem?
- Is this really the best way to meet the needs?

Here's how an exchange like that might go:

Inventor: "Check out my awesome new creation. This is better (cheaper or faster) than anything else out there!"

Stakeholder: "Well how is it better? And is it more important that it's better (or faster) this way than another way?"

Inventor: "Well there's nothing that does this new feature, or that does it this well..."

Stakeholder: "Okay so it's new. But how do you know that putting all this effort/resources into that new feature is actually the best way to meet your customer's/project's needs?"

Inventor: "But it does it in a really cool new way..."

Stakeholder: "But the user doesn't always care how you did it. They just want something that meets their needs, and does so well and efficiently. So how do you objectively measure the performance of your idea for meeting all of the customer's needs against other ideas or other products/approaches that are already out there?"

Inventor: “Uh…”

We as designers can become very passionate about our work, which is fantastic and we should never lose that passion. However in doing so, we sometimes get so focused on making our vision happen, or getting that “one thing” to work, we forget to ask the questions throughout the design process: Is this really the best way to solve this problem? Is this really the best way to meet the needs? Being a professional designer isn’t just about getting the design right but rather designing the right things in the first place.

This course provides an introduction to a number of tools and techniques you can use to help discover the needs of the challenge you’re trying to solve. With an understanding of the *need*, you can thereby prove the value of your solution to your users, boss, stakeholders, and the world!

Back to Table of Contents

Tool: A Checklist for Defining Scope

Download the Tool

Defining Scope Checklist

You want a clear vision of what your system's purpose is. In order to do this, you need to explore the scope of your system's interactions. Download a checklist that will help you define that scope through the use of a context diagram and a use case list. Keep this checklist handy and use it whenever you need to explore a new system. Once you have completed the steps in the checklist, you will have a more comprehensive view of your system's purposes.

It's recommended you read through all of the steps provided and then decide the best order for you. This course will take you through a typical sequence when working with the tools presented. But every design process is different, and as you explore your design process you may find that some steps can overlap and happen concurrently. By breaking the process into steps, you have a guide to help you recognize all that you should be doing. This helps you avoid accidentally leaving out a key part of the design process.

Note: The steps have been taking from the guide **Defining Your System's Scope** however the steps numbered in the checklist don't necessarily align with the guide's numbered steps. Still, all the necessary steps are clearly provided within the checklist.

Back to Table of Contents

Watch: Defining a Nameless System

At this early stage of defining your system, you don't want to give it a name. Instead, you want to begin documenting the solution space of your system using a tool called a context diagram.

In this video, Dr. Schneider describes how to begin building your context diagram and provides an explanation of why your system should be shown as **The System** rather than being given a more descriptive name.

Video Transcript

One of the most important things when starting a project is being able to determine what is the scope of your project, and by this, a great way to think about scope is: what is that you have control over, and what is it that you don't have the control over? Luckily, there's a great systems engineering tool to help with this. That's a context diagram is what it's referred to as, and it could really help to flesh out what are all the things that your system needs to interact with, and what are the ways that all those things interact with your system, and force those interactions to occur.

Now luckily this is one of those tools that you can really learn while you build them, and we're going to provide some really great examples for you that you're going to be able to copy directly some of the elements from those examples in order to make sure that you get the formatting just right, because formatting is extremely, extremely important in order to make sure that your work is being seen as professional, in many cases, even be accepted in the first place by your client. So again, formatting, extremely important but we have examples that'll help you out.

Now, in order to start this off, we're gonna start off really simple. We're gonna start out by putting your system in a box. Just a simple, singular box right there in the middle of the screen that says, The System. That's it. Then we're gonna place a dash box around that system box to basically indicate your system boundary. Anything inside that dash box is what you have control over. Anything outside is what you don't have control over. We'll be adding a lot more later, but this dash box and your system box is a great place to start.

Now you might already be saying, "Well I already know what I'm going to be designing, so instead of running the system, can I just say well I'm going to be designing a refrigerator, so can I say the refrigerator or the power plant or whatever it is?" And to this, I'm going to make a very clear distinction to say no. Do not do this yet. There's a very important reason. Let me give you an example.

So let's imagine we're being asked to design a car. Someone came to you and said, "Hey, we want you to design a brand new car." And so, to me, at least, this might start me thinking, well, maybe a sedan, mid-sized economy, something of this nature. But you have to take a step back and recognize, even if you're being asked for a car, that's not what they're really talking about. When they're asking for a car, they're really saying the best representation for a solution to their needs. What they're really saying is not that I need a car; I have a transportation need. I have something that basically needs to take me from point A to point B, that it can do so with great fuel economy comfortably, safely, reliably, that's got good cargo space, maybe looks cool, right? And anything that meets that set of needs is in fact a solution.

So although they might be asking for a car, it might turn out that a truck, a van, an airplane, a bicycle, those are all possible other transportation needs solutions, right? But the thing is, if they've never heard of an airplane before, they don't know enough to ask for it, so you really wanna try step back and say, "I'm not designing this specific physical solution, I'm designing a system in order to meet a need". So you don't want to name what that system is too quickly because otherwise you'll narrow down the solution space far too quickly because basically if you say it's a car, anything that isn't a car is an invalid solution and you couldn't possibly come up with any other kind of possible creative solution because you've already said it has to be a car.

Now, for the purposes of the examples in these videos we're going to assume the customer only accepts some kind of personal family vehicle. The one maybe that has worked with roads. So maybe the airplane is out. Maybe. What if you could actually design a vehicle, a car of some kind, that actually had some sort of fold-up wings that allowed it to be able to fly and be able to drive? Might that be something that could add additional value to your user? Maybe I need something that could interact with water. Maybe it has, you know, a boat-like feature to it. It might sound crazy at first but you want to keep your solution space open as much as possible because it's a lot harder to get to that boat-car idea if you started with a mid-sized sedan as being what it is that you have to create. And you're going to stay with that car idea until somebody

else figures out, "Yeah, actually the boat-car would have been a better idea." Then you'll be asking yourself, "Why didn't we think of this?" It's because you narrowed your solution space too quickly.

So that's why again it's important just to start at the beginning with The System, and don't name your system until you absolutely need to.

Back to Table of Contents

Tool: Context Diagram Template

Download the Tool

Context Diagram Template

You use a context diagram to help you begin exploring the various interactions of your system. Through the connections shown in your context diagram, you create a visual of elements that influence and interact with your system and how those interactions occur.

Download this template and keep it handy when you need to create context diagram. You will use this tool to help you create your own context diagram as part of a required assignment in this course. As a starting point, this template already provides the label **The System** in the center box. Then the outside boxes are for you to identify and label the stakeholders and elements that influence your system.

Note: This is a PowerPoint template for your context diagram. Examples, both basic and detailed, are provided as additional slides to help guide you through the process of populating your template.

You may also find it useful to review the more detailed, step-by-step guide to building context diagrams that can be found in the Course Resources module at the end of this course.

Back to Table of Contents

Watch: Identifying Elements that Interact with the System

Now let's consider what elements will influence the design of your system.

In this video, Dr. Schneider continues building out a sample context diagram. You will see how to add elements that will interact with your system.

Video Transcript

Okay, now we have our system in a box, and the next step is to start to add in all those elements that our system is going to have to interact with, or at least influence our system. We want to try to focus on having at least eight major ones that we're going to focus on, but we don't have probably more than 20 for the purposes of starting this out.

Now, what could be things that you might want to put around your system? Let's go with that vehicle example. Roads, for one, might be an example of something that you don't have control over. You can't actually change what the roads are, but your system still has to work with it, so roads are a great example of something that might be outside of your system. Now in each one of these elements, what we're going to do is we're going to place them in their own box, but we're going to place these boxes outside of that dash box, remembering that that dash box is your system boundaries. So things you don't have control over go outside of the box.

Now we're going to go into some important formating rules and guidelines, and this is again to really make sure that your work is recognized by the professional community. So it's important that you adhere to these rules very, very strictly. Now, for a context diagram, each box should be roughly the same size, if possible. Boxes should have square, never rounded, corners. Color is strictly prohibited; it's just black and white. Names inside of the boxes should be capitalized and all names should use the same font and the same font size. Short hand and abbreviated names are acceptable if the abbreviations are well understood and defined elsewhere.

All right, that's it for now, and we won't even worry about really arranging the boxes nicely at this point. Again, we're just going to try to brainstorm what might be a lot of different kinds of boxes that we want to put around this, the things we want to interact with or may influence our system. So for the vehicle example, it might be roads, right? It could be passengers, cargo, weather, wildlife, gas stations, parking spaces, other

vehicles, and we could continue on. I could add a lot more later, but that's a good start for now. I've got a bunch of different boxes there, but keep brainstorming on this and add more if you want. But again, have at least eight is what we're shooting for.

Now I may not even interact with all these things in the final design that we come up with here, maybe not even roads, right? We might even come up with a way to avoid that, but again, it's okay to start to brainstorm. Just add them in for now; we'll eliminate them later. And we can even get more specific in a lot of the examples that we've given here too. For example, there could be lots of different types of passengers, and we'll go into in a later video how to actually delve into more detail on the kinds of things that you want to do and how to split these things up, but again for right now you can keep it a little high level.

As a general rule of thumb, though, when you're thinking about trying to add these items in there, try to think about the things that will force your system to have to do something differently or make your system have to act in a special way. So, because of this interaction, your system will have to do something in a unique way, will have additional requirements, and that's what we're really trying to get at: What are the things that are going to influence the design of our system?

Now, you want to make sure you include lots of uncommon and undesired interactions as well, not just the way we anticipate things to work but all those uncommon, undesired interactions such as wildlife. It's one of the reasons we mentioned that one. Again if you're uncertain, you can decide to eliminate it later, but for right now add it in. Just keep on adding more and more ideas.

Now, before we break to give you your 8 to 20 boxes I want to point out a really common mistake when people first do this. Now they start getting excited about all of the kinds of things that they want their system interact with and so they start to list properties and characteristics as well. So in our vehicle example someone might write comfort or safety or good fuel economy. Well yeah those are things that you want your system to have but they're not just things that your system interact with. So we do not include those kinds of things in a context diagram. Again, we're trying to focus on nouns of things that it has to interact with, not properties that we want it to possess.

So, actually, more formally, those things like comfort, safety, good fuel economy are things referred to as performance criteria, performance measures, and it's an extremely important topic that we are going to cover in an entire video later on. But for now,

again, we're going to focus on just those things that the system has to interact with. So stay tuned.

Back to Table of Contents

Activity: Add Elements to Your Diagram

In this activity, you will add **8-20** elements to your **context diagram**. Each element will appear in its own box.

When adding element boxes, keep in mind these formatting rules:

- Each box should be roughly the same size.
- Boxes should have square, never rounded corners.
- Color is strictly prohibited. It's just black and white.
- Names inside the boxes should be capitalized.
- All names inside the boxes should use the same font and font size.
- Shortened and abbreviated names are acceptable if the abbreviations are well understood and defined elsewhere.

Note: As you work through the activities in this course, you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Back to Table of Contents

Watch: Differentiating Primary and Secondary Stakeholders

When designing your system, it's critical to be aware of your stakeholders and how their needs influence your work.

In this video, Dr. Schneider describes stakeholders and how they affect the design of your system.

Video Transcript

Just as giving a presentation isn't about what you want to say but rather what your audience needs to hear, creating a solution isn't about designing the solution that you want, it's about designing something that meets the stakeholder's needs. Stakeholders are really important to think about whenever you're brainstorming about a design overall, really trying to understand their needs is fundamental. And brainstorming is actually a really fundamental part of developing a context diagram.

There's a saying that "All mistakes are made on the very first day," and that's usually due to the fact that we didn't spend enough time brainstorming, and hence we miss part of the problem, part of the requirements of what our system actually had to do: What did it have to interact with? What was most important for that system? And as a result, you wind up designing the wrong system overall. So let's focus on thinking about our stakeholders for awhile, and use to help us brainstorm different things to add into our context diagram.

Now there's a lot of different ways that people talk about stakeholders, and they're really isn't one formal definition for what makes for a primary stakeholder versus a secondary stakeholder overall, but we're going to think about a primary stakeholder as anyone or anything that has influence - direct influence - on your design. Now this doesn't mean, necessarily, your boss, but more about who is your main user, and what are their needs, and what might they need your system to interact with. Hence, this is a great way to think about brainstorming for your context diagram.

Now your user, however, doesn't have to be just an operator or a person, per se. It could be lots of other things. Let's say for example you're working on part of a system that's got lots of different aspects to it, so you're just working on, say, a subsystem. For a vehicle example let's say you're working on the vehicle engine. So there's a lot of

different kinds of things that your vehicle engine may have to interact with, such as, let's say, the vehicle chassis could be another subsystem that it has to interact with, the cooling system, the fuel system, etc. Those could all also be thought of as stakeholders, things that might influence the way that your system has to be designed, compose different kinds of needs in the way that they interact. So, putting the vehicle chassis, cooling system are all great things to include outside and into your system boundary and into your context diagram.

Other common elements might be elements of infrastructure that you don't have control over: say, existing manufacturer lines that you know you have to work with. Those could have a very strong influence in the way they interact with your system design, and potentially have a very strong effect on what are the costs of your system, which is something that pretty much all stakeholders care about. Now, in general though, you want to think about stakeholders in terms of anything that your design effects of your design is affected by. By, so those are the kind of things we want to brainstorm around for coming up with other elements that might be interacting with your system.

There's also this general notion of secondary stakeholders which gets the idea of almost being one step removed from the system. So things that might not interact with your system, but are strongly influenced by it. A good example might be your company, which might have their own policies, maybe branding requirements, all which might influence your design, and outside of your control could be location which you might be operating with. You can't necessarily change that, but again that might have a strong influence or it just might have a strong influence on it, and with that, the environment is also a classic secondary stakeholder you might want to think about overall. It's important to recognize these because they're often, in times, a source of constraints as well.

Again, another example of a secondary stakeholder might be something like the Department of Motor Vehicles for a vehicle example, and all of the government regulations that they impose are ways that your system's going to have to interact with the Department of Motor Vehicles. They're not a direct user, but still that's a really important stakeholder that you need to be thinking about for brainstorming in your context diagram, defining your scope. Now these are elements that often create influences on your design, and by thinking about all these influences, you may

recognize early on that there are better design choices that can be made, or help prevent you from making poor design choices in the long run.

For example, you may notice that there's an opportunity to gain a little bit of extra profit here, but you could have a large cost to the environment. And if you didn't think about that interaction early on, you could have designed yourself into a corner to make it harder to make the adjustment later on. Similarly, you might design something that looks really cool, but it could be really hard for people with disabilities then to use your system; if you didn't think about that stakeholder and that user at the beginning, hard to make those adjustments slit later on as well. So again it's really essential that you think about these elements for your brainstorming process here.

Now what we want you to do right now is to think about your own list of your stakeholders. No need to distinguish between a primary and secondary stakeholders at this point, because again, definition isn't really a formal one but it's important to think about who all these stakeholders are, what they are. Then think about what needs they have for your system and what interactions they might cause.

Back to Table of Contents

Activity: Add Stakeholders to Your Context Diagram

In this activity, you will create a list of your stakeholders and add them to your context diagram. Review the worked example provided to ensure you understand the relationship between your system and your primary and secondary stakeholders.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Add Stakeholders to the Context DiagramTask: Add Stakeholders to the Context Diagram
This is Complete When...This is Complete When...
Worked ExampleWorked Example
Back to Table of Contents

Watch: Making Connections between the System and its Context

Once you have these content boxes in your diagram, it's time to create connections between these elements and your system. Doing this will provide interesting realizations about your system and what it has to do.

In this video, Dr. Schneider describes how to make connections between your system and external elements.

Video Transcript

Now we're focused on the interactions between our system and all the boxes around our system that our system's going to have to interact with. So those connections are really important to help make realizations about what our system actually has to do. For this part we're going to focus on trying to pick our eight most critical boxes. By critical we mean what is it that is most important or what are some of the most demanding functions that are going to be required as a result to the interaction between your system and these exterior boxes, these exterior things that your system is going to have to interact with overall.

Now we're going to keep it just eight at this point, and as a result you may want to focus on things that are a little bit higher level. So although you could have lots of different kinds of passages for example, maybe we'll just start with just a singular passenger box so we get a good mix of different kinds of boxes around our system overall. Again, don't worry so much about picking the right eight at this point, you can always add more boxes later or choose a different eight boxes as a part of it. Iteration is very natural. But again, for now we're just going to focus on picking eight that we think we can work with, and will require a wide variety of different interactions or requirements of our system.

And once again, formatting is extremely important. So let’s look at the example pretty closely as we talk about this. All connecting lines must be drawn as rectilinear lines. Curved chrono lines, for whatever reason, are considered very unprofessional. To signify that the outer box item is going to do something to or with the system, write what that will do on the connecting line, but on the end closest to the outer box item. To signify what the system's going to do to, with, or for an outer box item, write what the system will do on the connecting line, but on the end closest to the system box. Notice that is

its also common to place this text within your dash system boundary box. See, this reads as, "Your system drives through weather." Great. Not "Weather drives through system." That would be bad. Interactions should also be written in lowercase only. Only one line is allowed to connect between any two boxes. So if you have multiple interactions, they should be placed on the same line separated by commas. Not semicolons, dashes, slashes, has to be commas.

Now we can begin to rearrange the boxes to keep the lines clear and assure that any text is clearly associated with one line and one line only, and we don't want our lines to cross either. And the little silly line jumps, they have no place in a context diagram. Now occasionally, you may have more text that can neatly fit on a line. So you sometimes see people use unique capital letters and circles to make references. This is actually acceptable often, but you may want to check with whom you're going to submit the diagram to, just to be safe. But if you are going to do it, just place the reference definitions together, particularly at the bottom of the diagram.

Now, connections are sometimes often desired to be made between the outside context boxes. That means between not your system and outside box, but just between the outside boxes themselves. There's a strong desire to do this, but there's a big caution in this, to do this very, very sparingly. We only want to indicate this kind of connections on our context diagram if we feel that this interaction is going to have a strong effect on our system design. Let me give you an example. Here on the streets, weather affects the visibility of passengers. Weather and passengers are both things that are outside of our context, right, of our scope. So they go outside of our main dash box, outside of our system there. But they still might be worthwhile to include as a system that could help to overcome this interactions effects could be potentially more valuable for the stakeholders. So we'll include this outside interaction in our context diagram.

Now, there could be a lot of other interactions possible between outside boxes but, again, we want to keep the focus on interactions between our system and these exterior items, minimizing the number of outer connections that there are overall. I also want to offer at this time a very common mistake warning: it can be really, really tempting to split up your system box, again that's that one, the system box in the dash box, into several different boxes. For example, you might want to add something that says gas tank interacts with gas station in some way, right? So the gas station is being part of your system. This is a big violation. Do not do this. I repeat, do not do this. This is

very strongly against the professional rules of creating a context diagram and can be grounds for immediate rejection of your diagram in any kind of form of submission.

More importantly, putting the formatting aside, this is actually goes against why we don't name our system in the first place because basically if you start to break up your system into subsystems you are forcing that these subsystems have to exist, and you're preventing a better way to handle these interactions in the future. For example if you're putting gas tank in there, you might prevent the possibility of actually being able to create an electric vehicle that might not need a gas tank at all. So again, make sure you keep it as just this system, as much as you may want to split up those interactions.

Right now you actually have all the information you need to correctly create a context diagram but I am going to encourage you to hold off from making your connections quite yet. We are going to go onto some more details on some videos that are going to help you do it a lot more valuably and a lot more efficiently as you go forward.

Back to Table of Contents

Activity: Determine Outer Box Interactions with Your System

In this activity, use your context diagram to show the interactions your system has with outer box elements. Review the worked example to ensure you know which details to include and how to correctly format these components in your context diagram.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Determine Outer Box InteractionsTask: Determine Outer Box Interactions
This is Complete When...This is Complete When...
Worked ExampleWorked Example
Back to Table of Contents

Watch: Iterating the Context Diagram

The more you began to understand the needs of your system, the more you will want to provide further detail to your context diagram.

In this video, Dr. Schneider describes how to delve deeper and how to accommodate your growing list of needs.

Video Transcript

Now I want to focus on how to iterate your context diagram by either breaking up your existing exterior boxes further, figuring out ways of combining them to be more efficient, and when to actually justify removing some of those outer box items. So one of the ways that you may have already recognized that you can break up boxes is by thinking about more specific examples. Our passenger box, for example, was one that we can think about breaking up in terms of drivers, maybe your passengers, your children passengers, maybe you have elderly or disabled passengers as a part of it, or elderly or disabled drivers, maybe even a drunk driver. Those are all different kinds of examples of passengers that you might have to break up in your context diagram further.

Weather could also be one because you could have snow. You get a heavy rain, you get fog, any other kind of inclement weather, and all these different kinds of things are going to require your system to do something special or act in a different way. Hence, it is worthwhile to break those boxes up in those cases. Again, anytime that you feel that your system would do something special as a result of a new kind of interaction, it's worthwhile to include that into the context diagram. So those are good reasons to break up different ones.

Now on the other side of things, you may want to try to combine them as well, because you can imagine this could grow very quickly. So let's think about our vehicle again, and let's think about all the different kinds of interior messes that you might wind up with. You might have water. You could have spilled dirt, soda, fast food, juice, gas, milk, animal excrement. Sure it's gross, but you have to think about all the different kinds of ways and things that your system may have to deal with, especially those that you may not necessarily want to interact with. But again this is a really long list, and we don't

want to have to just think about every single possible one that there is, we'd rather be able to focus on a couple that are representative of the entire group.

So one of the things that we do is we look for overlapping needs, and by this we mean say, "Can I pick just a few of these that if my system was able to handle these, they would be able to handle all of the rest of the ones as well?" So sometimes it is also thought about as looking for what are representative of the worst case. More formally, this is referred to as an argument of dominance. If I can do this, I can do all these other ones, because this one is more dominant, it's a harder case to deal with overall. And so for our interior mess example, you might pick something like maybe just one or two of those such as oil, as an example, something that's greasy, flammable, and a liquid, right? So I can cover that whole set of different things. And maybe I'll pick baby vomit as another one: yes, it's organic, so it's also a potential biohazard. It's smelly, and it's a combination of liquids and solids. So I'm covering a lot of different kinds of things all at once. So in this case, baby vomit is actually a very good thing to include in your context diagram.

Now, in order to help make these diagrams be more digestible and readable, quite often another thing to do is to split up the context diagram into several, and there's a lot of ways to do this overall. One way that we've kind of hinted at is you could have one that might be high level, such as one that you might have just passengers on and then you might have some lower level ones that maybe deal with all the different kinds of passengers or all the different kinds of weather, separate context diagrams just for those lower level focuses overall. Another way you might think about it is maybe you see a subset as being the core problem, as these being the set of interactions that have the strongest influence on your design decisions overall, and hence you want to put those all into a single context diagram. And then you have other context diagrams and maybe focuses on special cases, or maybe one that just focuses on your secondary stakeholders.

There's really no one particular rule to do this and there can be lots of different ways. In our vehicle example, we actually split it up in terms of interaction that occur on the interior on the vehicle, and interactions that occur with the outside of the vehicle. Again, there's no wrong way of doing this. The important thing to remember is whenever you're doing any systems engineering tool, primarily, you are doing this for your benefit. It's not to meet some government requirement or even to make your boss happy or customer happy. These systems engineering tools are here to help you in your

design decisions. So in that way, there really isn't a wrong way to split it up. In the end, you may need to massage it in order to meet your boss's more stringent requirements or customer's stringent requirements, but it doesn't mean you can't still hold onto these versions that helped you the most along the way.

Back to Table of Contents

Activity: Iterate on your Context Diagram

In this activity, you will use your context diagram to continue to show the interactions between your system and outer box elements. Review the worked example to ensure you know which details to include and how to correctly format these components in your context diagram.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: IterateTask: Iterate
This is Complete When...This is Complete When...
Worked ExampleWorked Example
Back to Table of Contents

Assignment: Course Project, Part One—Create a Context Diagram

In this part of the course project, you will create a context diagram for a system you want to develop using the **Context Diagram Template**.

It is strongly recommended that you complete the assignment based on a system of your own choice because you are more likely to benefit personally and professionally by working on such a project. However, as an option, a project setup is available for you to use as a basis of your project work. In this hypothetical situation, you are a design consultant that is working to meet a Request for Proposal (RFP) for a New Toy design. You can download the **New Toy RFP** to review this project setup.

Completion of all parts of this project is a course requirement.

Instructions:

1. Download the **course project document.**
2. Complete Part One.
3. Save your work.
4. You will submit your completed project at the end of the course for grading and credit.

Do not hesitate to contact your instructor if you have any questions about the project. You will add to this document as the course proceeds and will submit it to the course instructor at the end of the course.

Before you begin:

Before starting your work, please review the **rubric** (a list of evaluative criteria) for this assignment. Also review eCornell's policy regarding **plagiarism** (the presentation of someone else's work as your own without source credit).

Back to Table of Contents

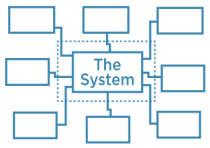
Module Wrap-up: Describing Your System in Terms of Interrelationships

In this module, you began the process of exploring your system. You identified your stakeholder and other entities, differentiated primary and secondary stakeholders, and made connections between your system and its context.

At this stage, you have developed a context diagram to help you define these interrelationships between your system and other factors. You have completed the first part of understanding these interactions, placing you closer to your overall goal of defining the scope of your system.

Back to Table of Contents

Module Introduction: Identifying Key Scenarios



What does your system need to do? What functions must it perform? These are important questions to address during the early design stages of your system. To help answer these questions, you need to identify and build a list of all the ways in which your system will be used.

In this module, you will develop a use case list which will help capture the set of functionalities that your system must have to meet the needs of the challenge you are addressing.

Back to Table of Contents

Watch: Identifying Use Cases for Your System

Consider the situations in which your system will be used. You will want to create a list of these situations, or use cases.

In this video, Dr. Schneider describes ways to identify your use cases.

Video Transcript

Now we're going to try and focus on creating a list of the situations in which your system will be used. These scenarios, if you will, or again, the ways in which the system may be used are often referred to as "use cases," and use case analysis is a really important initial step of the design process. Essentially, we're trying to create a complete view of all the system needs, everything the system needs to do overall. This is really important because quite often if you don't you find out later on in this process, "Man, if I only recognized earlier on this is what my system needs to do, I would've decided this in an entirely different way." And it causes a large amount of redo in as a result of that.

Coming up with these use cases can also help you to prioritize quite a bit. You might get really excited on working on one part of the solution, as we engineers typically do, but then we may find out later on that we don't have enough resources to handle all of the other things our system needed to do. So again, both for gaining the complete idea and prioritizing, use cases are very important.

Now I'm going to walk you through an iterative process in creating these use cases. And this iterative process is not just good for the first time that you do it, but this is a process that I still follow, and many of my other professional colleagues do too. A good place to actually start this is with your context diagram. Now, on your context diagram you've already written words on your connections. For our vehicle we had things such as driving the system, fueling the system, getting the system washed, maintaining the system, manufacturing the system, et cetera. These are all great examples of ways that your system will have to do various things, so these are the good bases for a lot of use cases.

Notice that in this first pass, a lot of them are pretty high level. There's a lot of things that go on in driving the system. But that's okay to begin with that; it's an iterative

process. We also want to start with very short names and we want to try to focus on a single action or verb, hence "Driving the system," "Fueling the system," that's the single action verb that we're looking for.

Too often though the focus is placed on just the intended uses, that is the main ways of expected use, but it is very important to include in your use cases all the unintentional or even undesired use cases. For example, running over pot holes is an important use case for a vehicle, and perhaps an even more important, but highly undesirable one is surviving an accident. So these undesired ones can be very crucial in determining the way you actually design your solution.

Now, each of these will later help to generate specific list of needs and requirements that might have otherwise been overlooked. But so again, there's an important element and in the same way of helping to generate additional needs, you may go back to your context diagram and discover that there are additional interactions that you want to add, new boxes that you want to add. Actually that's quite common and when you think about it, really a good thing. Because in this iterative nature, we're making not only our use cases better, but we're going to make our context diagram better at the same time. So you may want to update your context diagram as you go through your use case process as well.

Now, keep in mind, what we've talked about so far aren't actually officially use cases. There's no main operator action noun that we've put into this yet. For example, who is fueling the system or what is fueling the system or what is driving the system, etc. So we'll be going over those parts later, but for right now, jot down as many ideas for use cases as you can, and in the end, we're going to try to aim for a list of about 20 to 30 good ones overall. But again if you can come up with at least one for each of your context diagram's eight main exterior boxes, you're at a good place right now.

Back to Table of Contents

Activity: Begin to Develop Use Cases

In this activity, you will use your context diagram take a first step toward creating a use case list. Review the worked example to ensure you know which details to include and how to use your context diagram as a resource in building this list.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Create a List of ScenariosTask: Create a List of Scenarios
This is Complete When...This is Complete When...
Worked ExampleWorked Example
Back to Table of Contents

Watch: Using Actors to Expand Your Use Cases

To expand your list of use cases, consider the different array of operators or actors who will use your system.

In this video, Dr. Schneider describes actors and provides examples of how different actors impact the use of your system.

Video Transcript

When you're creating your use cases, it's very important in order to make sure that you recognize that your system will function differently depending upon the needs of different users. And to help give an example of this, let's go back to that "driving the system" idea for a use case that we started out with before. Now, when you first think of that, you may think of an able-bodied person who's probably passed her driving test and was able to make good judgments. But unfortunately as we know, that's not always the case.

So I'm going to take that "drives the system" use case and actually expand it out into many use cases in order to capture the needs of different users. So we're still going to have one use case that we're going to call "Driver drives the system" in order to handle that more general use case that we're thinking about, a general situation, that able-bodied person. But let's also change up now, let's say "Student driver drives the system." Suddenly I'm beginning to recognize that my system has to do things in order to help train somebody to know how to use that system properly. Now, the way I design my system is going to be different in order to also meet the needs of that person who's learning about it.

Let's do it again: let's imagine we have "Elderly driver drives the system." All right, for this user we may recognize our system has to do the things in order to help improve the reaction time or the situation awareness of the user, deal with the constrains our user might have in those areas, again causing me to design differently than I might have if I didn't think about that user. Take another step, "Maintenance worker drives the system." In this case, it's more from a perspective of someone who wants to do diagnostics on the system overall. I have to think about "All right, how can I help that user be able to perform good diagnostics on the system?" So again, thinking about all

those different users' needs is going to focus our design of the system in different ways overall.

Now, so far we've focused on users as just being people, but there can be lots of other examples of users out there too. Common examples can be other pieces of equipment, maybe other subsystems of a larger system that you're designing. Software programs as well is another common example that might be a user that you have. Overall when we've been talking about users we're really talking about what is the main stimulus that is causing the system to have to do something, that is driving that use case overall. And so for example you could have even something that's like "Automated assembly line constructs the system," and here, the automated assembly line is driving the action that it's causing. So now we have stimulus and that's our user. Even though again, it's not a person or anything close to a person in that regard.

Now, when we add in this main noun as a user, if you will, we have now actually made these things become official use cases. And as such, we often try to refer to users with a more generic term. One of the most common ones that's widely accepted is the term "actor." So instead of recalling the users we'll use the term "actor" and we're gonna keep with that term throughout our videos. "Operator" is another common term. But again we're going to stick with actors.

Now there are even a few situations though that you might not have an exterior actor. Instead, it's more of an internal focus. As an example, you might have the system that is doing something just completely internal to itself, or maybe the system is monitoring something else. It's not that something else is directly interacting with our system or doing something to the system, the system is taking a more active stimulus role in that case overall. So a good example with our vehicle might be "The system monitors the weather." So even though we don't have an exterior actor, it's still considered an official use case in that situation because again it's the system that's driving the action overall.

Now another thing with regards to consider with actors is we have a general tendency to focus on the ones that are most common, the ones that we think about that we want to use our system. But more often than not, it's those other actors that actually cause more demanding needs. And we want to make sure that we seriously consider those, including those actors that we don't want to use our system. Now let me give you an example, what if you had "Drunk driver drives the system"? All right, now that completely changes the situation, right? Because now we're recognizing our system has to do many different kinds of things in order to reduce the likelihood or severity of

the potential negative effects of that use case that might happen, and causing us to design our system again in a very different way than if we didn't recognize that. And it's very critical that our system did recognize the important needs that came from that situation.

So, as you noticed in developing these use cases it's very common to have multiple variations of the same idea but with different actors overall. So what I like to do is basically two things: I want you to write official use cases now. So out of your past use case ideas, add in an actor wherever it's appropriate, maybe just driver or operator, some representative actor for it. And again its okay if some use cases don't have an actor again, just saying that the system does something there, they're driving the main action, and, again, we want to be writing official use cases now.

The other thing I want you to do is write out a list of potential actors including all of those undesired actors or any ones that might cause your system to do something special. We'll have some neat ways of doing some copying and replacing later, in order to be more efficient in representing all those different ways, but again, for right now, official use cases and your list of actors.

Back to Table of Contents

Watch: Stakeholder Input and Derived Use Cases

It's important to pay close attention to functionality that has been requested by stakeholders as you continue to build out your list of use cases. There may also be detailed functionality that is not specifically requested but is necessary in order to meet the broader stakeholder need.

In this video, Dr. Schneider describes how to take functional requests from your stakeholders and put meaning behind them to derive additional use cases.

Video Transcript

So far we've focused our development of use cases on basically the interactions between exterior items, as using our context diagram, as well as thinking about the actors and their needs and what that's going to cause our system to do. But there's a couple other sources we want to make sure we include as well when we're thinking about use cases.

Sometimes, one of the most important things to think about is actually, what did the stakeholder directly tell you that they want, that they need, and this in terms of functionality, not specifically what it is, but what is the functionality that they are looking for. For example, you might have that the user wants the system to alert the driver to potential driver drowsiness, as an example. You might have the system automatically parallel park, as an example for those. Those are both kinds of specific functions that we want the system to be able to do that might come directly from a stakeholder request. Now again, it doesn't tell us how we're going to do that overall, but those are specific functions, and we want to make sure that we capture those, even though they might not show up directly in any of our previous brainstorming for use cases.

Other things that often show up are things such as "System completes the test course successfully, meets some kind of criteria overall." Now, we have to be very careful on writing those because what actually do we mean by "success" in regards to that, so normally you wanna try to drill down a little bit deeper than that, and think about this as a general form: "The system performs task x" or "The system performs task x to meet criteria y." That's kind of a key way you want to think about all the ways you are writing

those direct stakeholder request overall, and we will talk more about how to to actually define criteria more specifically.

As you are going through your design process as well, you may also recognize that although we've been talking about largely external interactions, there are a lot of kind of things that your system may have to do internally, and you want to be able to capture those use cases as well. An example of these for our vehicle might be that the system converts fuel into motion, that it measures its speed, that it protects against internal wear and corrosion. There's lots of other ones that can happen in this regard.

Now the thing though with these is as we try to keep our system fairly generic or undefined at the beginning, we may not be able to make as many of those use cases at the start. And those use cases will evolve as we go through our design process, and it's important that we continue to add in that those uses cases as they are identified to make sure that we don't miss any of those needs as we go on, and we check ourselves against our use cases.

Now whenever you're developing your list of use cases, again, we get to this notion of always going back and checking, making sure that you are iterating and that you are checking with all of the stakeholders or sources that the original promise statement came from. This could be your boss, this can be re-reading the original request for proposal or talking with whoever is sponsoring that request for proposal. And even if it was your own idea, you want to make sure, are you really still addressing all the problems that you originally intended to, because we tend to get excited about a particular aspect of our system, and we can use these use cases as a check, to make sure that we are still meeting all those needs.

But again, whatever the source of that problem statement, you want to make sure that you discuss your list of use cases with them the best that you can, to make sure that you really are covering everything that they're looking for.

Back to Table of Contents

Activity: Add Use Cases from Stakeholders and Internal Sources

In this activity, you will use sources other than context diagrams to determine additional use cases for your system. Review the worked example to ensure you know what to consider and include in your use case list.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Add from Other SourcesTask: Add from Other Sources
This is Complete When...This is Complete When...
Worked ExampleWorked Example
Back to Table of Contents

Watch: Delving to Expand Use Cases

After several rounds of expanding your use cases, you should have a substantial list. You can now use the list itself as the basis of further expansion of your use cases. Keep in mind that the idea is to identify all use cases that may require unique functionality. This can lead you to examine variants of existing use cases. For example, by modifying a use case slightly you may discover a special case that requires you to address a new need.

In this video, Dr. Schneider outlines techniques to help you identify included use cases as well as special cases that extend previously identified functionality.

Video Transcript

You've probably already noticed that many of the use cases you've generated have been more at a high level, and as such you've naturally started to want to delve down into more details as you've thought about all the different perspectives to think about these use cases. So in this video we're going to talk about delving down, and how do we represent the relationship between those high level ones and those more specific ones?

And, again, let's start out with examples. Let's take driving the system, as we've been using that one quite a bit. Now, you might recognize that in driving the system there's a lot of other sub use cases, or it includes a lot of use cases such as being able to stop at an intersection, changing lanes. Each one of these might be special use cases that you might think about that are included in driving the system. In fact you might even go even further say well changing lanes, actually, there's special versions of that too. There's merging onto the highway. There's exiting the highway. Those are special versions of changing lanes, as well. And, so, you want to be able to show that all these things really relate back to that original use case of driving the system.

And, so, if it's a common occurrence that this sub use case is part of a large use case. We use the term "include" in order to basically say that driving the vehicle includes these other use cases. That's an important part of terminology, again, one that we use to represent the relationship there.

Now there's other situations too, that might happen only under special conditions. These ones we refer to as extending a use case overall. Let me give you some examples.

So driving a system might be, a special case might be driving through snow. It might do many of the other use cases, but again in a unique manner. So we basically say that driving through snow "extends" driving the system. And it's important to recognize as many of these special cases as possible, whether they be from different operating conditions that are causes, different user conditions as well that might be happening. You want to do as many of those as really are realistic, so again, you get as complete a picture as possible. Again, for these special cases, we use the word "extends," typically.

Now you may find that you want to repeat many of these use cases with all of your relevant actors. Luckily, in order to flesh out your list, you can use simple copy and then find and replace actions in order to replace, you know, the driver with the different kinds of drivers, perhaps, that we talked about before. However, you want to make sure that you don't just do this in a blanket format. You only want to repeat use cases with a different actor, if you believe the system will have to do something very differently for that actor, overall. So again, that's the key difference in making sure that you have both a good in depth look at the use cases, but also still staying focused on the use cases that are going to make the biggest difference to your system.

Back to Table of Contents

Watch: Recognizing When You Have a "Good Enough" Set of Use Cases

At some point you will have identified enough use cases, but how do you know when you have reached that point?

In this video, Dr. Schneider describes some guidelines to help you decide when you have enough quality use cases.

Video Transcript

When you're refining your list of use cases, you often come across a question of how deep should I go, what is really a reasonable level? Actually that's something you hear quite often. You were asked to develop these things to a reasonable level. How do you know you've reached that? Well a great indicator often with a lot of these tools is to see are you prepared well enough in order to take the next steps?

And myself, when I'm looking at use cases, I try to answer these basic three ideas. One, can I clearly state the initial and ending conditions of each use case? Do I even understand what that use case really entails overall, is it well defined? Two, am I able to basically think about what is the list of step by step functions that might have to occur throughout that use case in order to get from that initial state to the end state without feeling too long or running over several other use cases that I have? Otherwise I may need to delve down deeper, and include or extend more use cases from them.

The third thing I ask as well is do I feel that there is at least a significant subset of functions that performed in this use case that are unique? Said another way, does this use case capture various needs that otherwise might have been missed if I didn't consider this use case? When I feel like I've covered those series of things, I've defined my use cases well. I wanted to make sure that I've got a good breadth of those across all the different kinds of needs that I need to consider.

Time is always the most valuable commodity. And so we want to make sure that we are being as efficient as possible, and I use these three guides to help me make sure that I have thought about my use cases well enough, in their level of detail.

Now, you're doing this, perhaps for the very first time, and it's important to recognize that in the same way you might recognize a pattern or an opportunity when you're

learning to play a sport, or musical instruments, or even a video game, you're training yourself to think and act in a new way. You have to recognize that, as such, the first time you go through it, it probably won't go that perfectly. It probably won't go that perfectly the second time, nor the third time. It's going to take time to develop these skills and really think about these problems.

As you continue, you'll be developing your intuition. By the time you've done this a number of times, you'll look back and even the first time that you thought you did a great job and be like, "Wow, I can do even such a better job at this right now," because you develop that intuition as you're going with it. So allow yourself the opportunity to learn and experiment and iterate with this. And bounce your ideas off of as many other engineering system people that you can in order to help develop that intuition.

Back to Table of Contents

Activity: Refine Your Use Cases

In this activity, use your use case list to refine what the use cases will be for your system. Review the worked examples to ensure you know what to consider and include in your use case list.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Refine Your Use CasesTask: Refine Your Use Cases
This is Complete When...This is Complete When...
Worked ExampleWorked Example
Back to Table of Contents

Discussion: Evaluate Your Use Case List

If you've been completing the activities as you have progressed through this module, you should have at least 20-30 use cases for your system. Now you're going to share your list with your fellow students, and as a group you will evaluate one another's use case lists.

Instructions:

You are required to participate meaningfully in all discussions in this course.

Discussion topic:

Part I

1. If you have not already done so, generate a list of at least 20-30 use cases with at least half from the same actor.
2. Rank them high, medium, or low by how critical they are to the functionality or performance of your system.
3. Evaluate your list by using the following rules of thumb.
    a. Can you clearly state the initial and ending conditions of your use cases?
    b. Could you describe what occurs during the use case as a series of step-by-step functions that your system must perform? Could you do so without it feeling too long or running across several use cases?
    c. Does each use case capture functions that might otherwise have been missed if this use case wasn't considered?
4. Post your use case list along with a brief description of your project. You should be able to copy the description from your Course Project Part I.

Part II

Choose another student's use case list. Read their brief description and review their list of use cases. Try to identify one or more additional use cases and include these in a reply to their post. As a starting point, consider what cases might be included in existing use cases, or how special cases might extend existing use cases.

To participate in this discussion:

Click **Reply** to post a comment or reply to another comment. Please consider that this is a professional forum; courtesy and professional language and tone are expected. Before posting, please review eCornell's policy regarding plagiarism (the presentation of someone else's work as your own without source credit).

Please note:

While discussion board postings will be accepted through the end of the course, we strongly encourage everyone to move through this course as a group. As such, postings that are made to this board after 5pm ET on the due date will be read and graded but may not receive a response on the board from your facilitator and/or your peers. Please let your facilitator know if you have any questions.

Back to Table of Contents

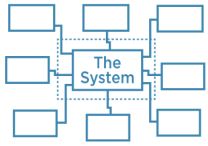
Module Wrap-up: Identifying Key Scenarios

In this module, you captured the set of scenarios that your system must handle to meet the needs of the challenge you are addressing. You identified and documented use cases for your system, used actors and problem statements to expand your use cases, and accounted for special cases.

At this stage, you have developed a use case list to help you define these scenarios. You have completed the next part of understanding your system, placing you closer to your overall goal of defining the scope of your system.

Back to Table of Contents

Module Introduction: Aligning Your Scope to Stakeholder Expectations



Now that you know what your system will be used for, it is important to make sure the use cases and vision you have for you system aligns with the stakeholders' expectations. It is also important to look at the level of work, or scope, required to build your system to fit within those expectations.

In this module, your main emphasis will be creating a shareable use case diagram that organizes the relationships between use cases. You will also begin to develop a scope tree that will help you recognize the effort needed to meet stakeholders' expectations.

Back to Table of Contents

Watch: Organizing and Communicating Use Cases with Use Case Diagrams

With your lengthy list of quality use cases, you're ready to explore how to provide this information to others.

In this video, Dr. Schneider describes how to organize your use cases into a use case diagram.

Video Transcript

Delving into your use cases, you definitely notice some relationships. Some are high level. Some are more generic. Some are more detailed in parts that focus on special cases, and you probably notice that your list is becoming fairly long in dealing with all these different kinds of use cases. It'd be really nice to have some way to organize this list. Luckily, there is a tool called use case diagram that is very helpful for this purpose. Not only for you personally, but being able to communicate across an entire team because right now, you might be able to say, "Yeah, I understand what driving the vehicles entails, I've got a good understanding of that." But can you effectively communicate that to others? Effectively communicate to others five years from now, or even five weeks from now with all the different kind of things that you might also have to be dealing with?

So to make sure that we're able to record these relationships, organize these use cases, we use this use case diagram as a standard way to communicate them. There's only so much detail and definition you can provide in any one diagram for it but at the very least this diagram serves as a good table of contents for some of the future work that we could be doing.

Now the use case diagram is one that is actually based off of UML and SysML formatting. For those aren't familiar with those terms they are standard governing bodies, if you will, on how to create different kinds of diagrams. And again the reason we have these standards is that someone from your team is able to communicate with someone else across the world and you can both look at the same diagram and interpret it in the same exact way. It's also very important to stick to the formatting for that reason and to make sure that your work is being seen as professional.

So, to get ourselves started, we're just going to do a couple of very first initial steps here in creating our very first use case diagram, in order to set ourselves up to, again, fill ourselves up in future videos. Now, I also encourage you guys to feel free to copy directly from any of the examples then modify it to your project in order to help to make sure that you maintain this proper forming, which is very, very stringent.

So to start, the first thing we are going to do is very similar to our context diagram, we are going to make a system boundary box and add a label to your box as the system, very simple there. Again, noticing that these are things that our system has to do and there's going to be outside forces once again. There is a formal diagram border and title that is normally added at the beginning, but I'm gonna recommend that we add that at the very end. You'll see that's going to be easier and it'll prevent rework if we leave that to the end as well.

We also want you to select one actor at this time and I'm going to suggest that it be that actor that you thought about for being at least in half of your use cases in the past. Maybe it's driver for a vehicle, maybe if it was a toy product that you're thinking of, a child could be a good representative actor. But what we're going to do is we're going to represent the actor in our diagram as a stick figure outside of our boundary. Why a stick figure? Well, again, it's the common formatting rule for this, but I will offer one small caveat to this, despite the formalness of this diagram, how that stick figure's actually drawn may vary slightly, depending on the software packaging that you use. It's the one diagram that people can be a little bit more forgiving.

Back to Table of Contents

Tool: Use Case Diagram Template

Download the Tool

Use Case Diagram Template

In order to share the use cases of your system, you need a tool to record and organize these use case interactions. Such organization and communication can be shared through a use case diagram.

Download this template and keep it handy when you need to create a use case diagram. In this diagram, you first have a box with the label **The System**. Then you will want to reference your context diagram and use case list to build a diagram representing the use cases for your system.

Use this template as a starting point any time you build out a diagram for a specific use case.

Note: This tool is a Microsoft PowerPoint document. One slide provides the template for your use case diagram. Another slide provides the use case diagram resources for you to copy and paste the different components as you build out your diagram. The final slide offers additional guidance through an example diagram.

You may also find it useful to review the more detailed, step-by-step guide to building use case diagrams that can be found in the Course Resources module at the end of this course.

Back to Table of Contents

Watch: Adding Uses Cases into a Use Case Diagram

Now it's time to start populating your use case diagram.

In this video, Dr. Schneider describes formatting requirements when entering key elements into your diagram.

Video Transcript

The goal for this next step is to add actual use cases into our diagram. For our purposes, let's focus first and try to come up with a good ten use cases to add into this diagram overall, help us to give us a good taste of what this diagram is really all about, and get a good use of some tools that are used to describe the connections between the use cases.

So, what I would like you to do, in thinking about what use cases to pick, is I'd like you to think about which ones are focused most closely with the actor that you've already added into your use case diagram. I also would like you to think about the use cases that you've already rated as being high, again, being most important in terms of performance or functionality. Then from there I'd like you to think about adding a few of the use cases that you might be included in that high level, high rated use case. And maybe one or two special cases as well, those extended use cases.

And let me give you an example. So I might have started out with the use case driver drives the system, and that being one I'd like to add to the use case diagram. I might also want to add some included use case such as driver accelerates the system, or driver brakes the system. Those are a couple good sub use cases there. As a special one, I might include one like driver starts the system. Often initialization or shut down procedures are good special case use cases to include as well.

Now, when making your use case diagram, it's of course important to pay attention to the formatting. And so for use cases, they're added as ovals or sometimes known, or more commonly referred to, as bubbles. So we're going to add the use cases as these ovals or bubbles into the actual diagram boundary that we have come up with so far. Again, label that as The System. It's going into our system because these are our use cases.

Now in writing each one of these bubbles, you want to make sure that the text is centered, is the same font size, and you want to try and keep all the bubbles roughly to be about the same size. And in general it's better to have larger bubbles with a little bit of blank space than to try to cram in a lot of additional text into a smaller bubble.

Now, notice when you actually write the bubble, when you're writing that use case name, you don't actually include the actor in there. So, when you write "driver drives a system," you actually can write it in the bubble as simple "drives a system." The idea behind this is that we are going to connect a line between the actor and the drives the system bubble in order to indicate that the driver drives the system.

Now many times it's because while use cases may end in the term, "the system," it's also common to simply drop that text as well so that in our use case diagram we simply see driver, connected to a bubble that says drives. And that represents driver drives the system, in a much cleaner format overall.

So for right now, the key thing is we want to connect only our highest level use cases at this point, and we'll make the connection to some of those lower level use cases later.

Back to Table of Contents

Activity: Populate Your Use Case Diagram

In this activity, you will start to build your use case diagram by adding in use cases. Review the guidelines to ensure you know what to consider and include in your use case diagram.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Populate the Use Case DiagramTask: Populate the Use Case Diagram
This is Complete When...This is Complete When...
GuidelinesGuidelines

Back to Table of Contents

Watch: Understanding Use Case Connection Types

In the following videos you can learn about the different connection types found in use case diagrams.

Includes

You will want to show connections between higher-level use cases and their sub use cases. In this video, Dr. Schneider describes how to use the *include* connector arrow for this function.

Video Transcript

Includes is one of the most common connection types that's used in use case diagrams. It's used mainly as a way to represent that a use case is really a sub use case of a higher-level use case, overall. Let me give you an example. So a driver drives a system, is a good example of a high-level use case. You could say that it includes other use cases, such as driver accelerates a system, or driver brakes a system, and many others. So by saying that these ones are included, we're saying the driver accelerates the system and driver brakes the system are sub cases of the driver drives the system.

In order to draw this, we use a dashed arrow, and it is drawn from the main use case to the sub use case. The direction of the arrow is very important. Notice that there's no direct connection made to the actor, instead it is implied through the tree that you're making in that the higher use case connects back into the actor. So, the way that this is read is it simply states that drives includes accelerates and brakes. Formally, the dash arrow should be labeled with the term includes, which should be written in lower case with double, greater than, and less than symbols on either side of the term.

Extends

Sometimes the connection between a sub use case and a higher-level use case can be seen as an extension of the higher-level use case to account for a special circumstance. In this video, Dr. Schneider discusses when and how to use the *extends* connector arrow.

Video Transcript

Extends is a use case diagram connection that is used to indicate that one use case is a special sub case of another use case. Let me give you an example. Driver starts the system can be said to extend driver drives a system because starting the system is a special initialization sub case of the overall operation of the system as a whole.

There are more examples that are provided in the example files, and in those examples you'll see that a dashed arrow is used to draw from the sub use case to the main use case. This is the opposite direction that the includes arrow is drawn in. Like all use case diagrams, the connection to the arrow actor is implied through the tree that you are making. This is read simply as starts is an extension of drives. Formally, dash arrows should be labeled with the word extends written in lowercase, with double greater than or less than symbols on either side.

Generalize

Sometimes you will want to show a connection from a specific case to a more general case. In this video, Dr. Schneider describes how to use the *generalize* connector arrow for this function.

Video Transcript

Generalize is a use case arrow that is used to indicate that one use case is a special version of another use case. In other words, the special version has to do everything that the original use case does, but in some special way, or having some additional functionality that needs to be required as a part of that. Let me give you an example. So let's say we had a use case called drives the vehicle. Now, some other ones that might be generalization are going to more special case or version of this drives a vehicle could be drives a semi truck or maybe drives the super spy car. In either of those situations you're going to have some special requirements, special functionality that will have to occur than you might have under normally just driving the vehicle as a part of that.

So generalized arrow is normally drawn as a solid line arrow from the special case to the more general case. The thing that also distinguishes the generalized arrow is that it has an open triangle head. And commonly, there is no text actually written on this arrow, and it is a very unique-looking arrow. Due to its computer science roots, and UML roots of this connection, it's also often spoken in a very different way. So you might say, driver drives the semi truck inherits from driver drives the system. Or driver drives a system is a parent of driver drives the semi truck. Or said another way, driver drives a semi truck is a child of driver drives a system.

Trigger

In certain situations, one use case will cause another use case to occur. To capture these situations, you will want to make a connection from an initial use case to a triggered use case. In this video, Dr. Schneider discusses when and how to use the *trigger* connector arrows within your Use Case Diagram.

Video Transcript

Trigger is an arrow commonly used in use case diagrams. However, it is not an official UML approved arrow, but you'll often find it to be very commonly used because it is very useful for representing a specific relationship. Often you might find that one use case could be the cause of another use case, and hence the trigger arrow is drawn between them in order to indicate this.

Let me give you a real simple example, if you have one use case called system detects fire, you could then draw a trigger arrow to another use case saying system extinguishes fire. So, the text fires use case triggers the extinguishing fire use case. In order to draw this arrow, use a dash arrow. Draw from the initial use case to the triggered use case. It is normally labeled with the term trigger written in all lowercase, and normally having the double greater than, or less than carrots on either side.

Back to Table of Contents

Watch: Selecting Use Case Connection Types

You have seen several different use case connection types. In fact, there are many other connection types that have not been described in this course. But these other connection types are seldom used.

In this video, Dr. Schneider describes how to choose among the most commonly used connection types as you connect use cases in your diagram.

Video Transcript

Even amongst the four connection types that we have discussed for use case diagrams, you might be questioning yourself. What is really the best connection to use? The truth is, several could actually be equally valid, depending how your team might view it. Let me give some more discussion on this. Do you want to think about a complex use case as being comprised of several use cases, such as including several use cases, or do you want to think about that more complex use case is actually an extension of a simpler use case? And that more complex use case just has to do with everything the simpler one does but in a special way, or just having a little bit extra functionality as a generalization. Or do you want to think about it that the simpler use case is simply a special situation that is considered a part of the complex use case, and hence should be extended?

Let me give an example, to go into more detail on this. So in this example, does exiting the highway include changing the roads because exiting highway comprises all of the functionality that changing the roads does and more? Or is exiting the highway a special version of changing the roads, and hence would be considered a generalization connection? Or is exiting the highway a special sub-case of changing the roads and hence you should use an extension? As you can see, you could actually make an argument for all three. The point is though, is that there is a connection, and that when you make your final design you recognize that when I'm designing to handle one use case, I have to consider how it will affect handling the other use cases that I'm doing it. And so on your first go, try not to spend too much time arguing over what connection is actually used. As you further your design you'll be able to recognize these things better and update it as you go.

Now there is one special caveat to this I'd like to go over. And that's if you are doing your use case diagram for software development. In this case, use case diagrams can play a larger role helping to establish what is your real software architecture. Now unfortunately it's beyond the scope of this course. I would like to offer my personal take on this nonetheless. People have spent an awful a lot of time arguing over use case diagrams or what connection arrows to be used, but in all honesty the use case diagram alone is not sufficient to be able to determine what your software architecture should be. So, you should really find that you are taking your best guess at these. And later, as you flesh out your system in your architecture, you may recognize that the choice that you made in the use case diagram may need to be changed and may need to be modified.

And for a lot of computer scientists, the need to have to go back and change various assumptions can be a difficult task. Let's face it, as computer scientists, we like to think of things as either true or false. Once something has been made certain, we sometimes have a hard time changing our logic behind it. But if you enter into the use case diagram creation with the mindset that, yes, I may have to change these things when I recognize the influence of these choices, you will find it to be a far more productive tool. Again, unfortunately this is outside the use or scope of our course to go into more details, but we are providing some additional references for you in the education materials that you can look into if you're more interested.

If you'd like to learn more about use case diagrams in a software setting, try using the following search term in a web browser: "Microsoft use case diagrams reference."

Back to Table of Contents

Activity: Make Connections Between Use Cases

In this activity, you will add connectors between use cases. When making these connections, keep in mind the purpose of each arrow connector. Also be sure to use these connectors correctly. Review the format guidelines for each arrow connector as well as an example of how each arrow connector is used.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Connect Use CasesTask: Connect Use Cases
Is Complete When...Is Complete When...
GuidelinesGuidelines
Back to Table of Contents

Watch: Adding Secondary Actors to Your Use Case Diagram

At this point, your use case diagram has one actor that interacts with your system, but there are other situations that involve different actors.

In this video, Dr. Schneider describes how to manage the different actors in your use case diagram.

Video Transcript

So far we've focused our use case diagrams largely on one actor. But I want to state very plainly that it is perfectly allowable to have more than one actor in a use case diagram. However before you do it there are a couple of things you will want to consider. The first is, as you add more and more actors in it, you can imagine that your use case diagram is going to grow very, very large. And particularly, potentially too large to be allowed for submission to some government agencies. Hence you really want to be careful that you keep your diagrams to a digestible size.

In order to handle this, with multiple actors, normally you'll have multiple use case diagrams, maybe one that focuses on each main actor that you have. And it's okay to repeat use cases in between different use case diagrams. For example, an elderly driver would probably have many of the same use cases that our original driver would have, plus probably some additional ones that you have as well. So in order to do this you can do it very quickly by simply copying your original use case diagram and all the use cases that you had, and then simply changing the actor name making it very very quick and easy to make this change. Then you can add any additional use cases or any additional changes to the existing use cases in order to make it right for that actor.

Now, this is actually useful, not only to make your diagrams more readable overall, but it's very helpful for a design concept referred to as viewpoint analysis. This design concept, the idea is, to be able to take a look at the system's needs for each potential user individually. That helps you recognize, what is the real value or purpose of a project for that special user group. It's also very helpful for determining scope and priorities and even different ways that an end user might evaluate that final solution as a whole.

Now you can add other actors to your use case diagram as well even if they are not the main use case stimuli for this use case overall, but they still play a significant role. Let

me give an example. You might add into our existing use case diagram a navigator as someone who might read the map or indicate where to turn. They're not actually driving the action, but still playing an important role. Similarly, if you were designing a toy, a child might be a main actor but a parent might actually be good to indicate that they are an essential part of some of the use cases.

These additional actors are sometimes referred to as secondary actors. They're not necessary to be included in use case diagram and sometimes they're not allowed. They do offer a more complete picture of what's going on. In common practice, in order to signify a difference between the primary actors and the secondary actors, an actor is placed on the left hand side of the bounding box if they are the primary actor, as we naturally read from left to right. Actors that are secondary actors are signified as such, as either placed elsewhere, outside of the bounding box. And sometimes, their connections use a dashed line, instead of a solid line.

Back to Table of Contents

Watch: Formatting Your Use Case Diagram

You have created a use case diagram that adequately conveys the interactions of your system with various use cases. Now it's time to finish formatting your diagram.

In this video, Dr. Schneider describes the final touches to wrap up your use case diagram.

Video Transcript

One of the last things to do for your use case diagram is to simply add in a border and a title. The border, as the name implies, is simply just a box around everything. The title, well there are various standards but let me give you one common way: precede your title with the characters UC then a period. This stands for "use case." Then capitalize all the words in your title name but do not include any spaces between the words. This is very similar to what is referred to as camel case, with the exception that the beginning title word which we are capitalizing as well in this example. In general, longer titles are better as they're far more explanatory and help you be able to distinguish between many different use case diagrams you may have across the team overall. But once you've added in the title and the border and done any final arranging for it, you're done.

Now before we conclude I'd like to add one final statement. You may be wondering, why did we not create a use case diagram from the beginning, instead of first making this list of use cases. As you get more experience, you may find that that's a way that you like to work, and just iterate through many versions of the use case diagram. However, when first learning these tools, it's really beneficial to break down the activity into smaller pieces. It lets you focus on what each tool really is.

When you were creating that list of use cases, you didn't have to worry about the name length fitting inside a bubble, whether your bubbles are arranged nicely, whether the right connection arrows are used, should you use multiple diagrams, what are secondary actors, what are primary actors. You just focused on discovering use cases and really identifying the scope of your work and ultimately taking the first step into finding needs of what any good solution should be able to do, which is what you really should be focusing on as a good designer. I still always start with a list of use cases, as do most of my respected colleagues.

Activity: Completing Your Use Case Diagram

It's time to complete your use case diagram. In this activity, you will review and add to your diagram a few components needed to finalize your use case diagram. Review the format guidelines to confirm your diagram has the correct formatting for the components you are adding in this activity.

Note: Remember as you complete this activity that you will be making progress toward completion of your Course Project. So although you won't turn in anything after completing an activity, you will need to save copies of your documentation produced in the activity. Some or all of it may be necessary to fulfill the course project requirements.

Task: Add a Border, Title, and ActorsTask: Add a Border, Title, and Actors
Is Complete When...Is Complete When...
Format GuidelinesFormat Guidelines
Back to Table of Contents

Discussion: Review and Refine Scope

At this point you have finalized your use case diagram, which means you have a good idea of the use cases and some of the sub use cases that your system must handle. Looking back at the context diagram you created earlier in the course, you can see the ways in which your system interacts with outside elements. Now you need to use your use case diagram and context diagram together to identify and address gaps in your system scope definition.

For example, you may find that you need to gather more information from you stakeholder. Possible reasons for needing more information include:

- A connection (interaction) is poorly understood
- The details of a use case are poorly understood.

In this discussion you will share documentation of your system's scope and get feedback from your fellow students that may help you refine the scope.

Instructions:

You are required to participate meaningfully in all discussions in this course.

Discussion topic:

Create a post in which you:

Part I

1. Share a brief description of your system. You should be able to copy the description from your Course Project Part I.
2. List any additional information you'll need from your stakeholder to help you define your system's scope.
3. Attach a context diagram in for your system using the *Attach* link in the Reply window.
4. Attach a use case diagram for your system.

Part II

Choose another student's discussion post. Read through their brief description and review both the context and use case diagrams. Looking at the context diagram, try to

identify:

- Additional exterior elements in the context diagram
- Connection labels that can be added or refined

While looking at the use case diagram, try to identify additional use cases that connect in some way with the existing use cases. Recall the connection types: includes, extends, generalize, and trigger.

In a reply to the original post, include your ideas for possible scope refinement.

To participate in this discussion:

Click **Reply** to post a comment or reply to another comment. Please consider that this is a professional forum; courtesy and professional language and tone are expected. Before posting, please review eCornell's policy regarding plagiarism (the presentation of someone else's work as your own without source credit).

Please note:

While discussion board postings will be accepted through the end of the course, we strongly encourage everyone to move through this course as a group. As such, postings that are made to this board after 5pm ET on the due date will be read and graded but may not receive a response on the board from your facilitator and/or your peers. Please let your facilitator know if you have any questions.

Back to Table of Contents

Watch: Using Scope Trees

Now that your use case diagram is complete, it is time to think about what is needed to deliver the end deliverables. The path to creating these deliverables often consists of completing a series of intermediate sub-deliverables and tasks. You can build this path and create a visual representation of these sub-deliverables and tasks using a tool called a scope tree. This tool is also sometimes called a "deliverable tree" or "question tree."

Creating a complete scope tree can actually be a rather involved process, but the result is a valuable tool for organizing the team effort. It can also play a crucial role when you are trying to manage stakeholder expectations.

In this video, Dr. Schneider describes a scope tree and how to use it as a planning and communication tool.

Video Transcript

With the work you've done so far, you're now prepared to try your first attempt at something called a scope tree. A scope tree is a very valuable tool that can be used, as the name applies, to really help define the scope of a project, with the focus on really what are the deliverables that you're going to be providing to whoever your customers or stakeholders might be. Now, the way that a scope tree works is you start out actually with what your end deliverables are. These become the root node to your tree. Then what you do is you start to think about what are all the things you need to do in order to deliver that end deliverable? And then you repeat.

Let me give you an example on this. So, maybe you're asked to deliver a report. Maybe one of those key elements of that report is some kind of analysis. You must be able to do some tests to get the information needed to run the analysis. In order to run those tests, you need to be able to first create a prototype as a part of that. In order to create a prototype, what are the tasks that need to happen to make that happen? And you continue to drill down again and again and again until you get to some very basic task, often referred to as an atomic task, as you're not able to break it down further. But once you have all these atomic tasks listed out, you can essentially follow back up your tree and see the path that is necessary in order to arrive at your end deliverable. Hence, you have defined a scope of all the tasks necessary in order to meet your goals.

This is a great tool for helping to build a really good timeline or even charts, as an example. However, it's really also a very good tool for being able to review what is being expected of you with your customer. Saying, "All right, you asked to report, this is what we believe a report entails. This is what we believe an analysis entails. These are the kinds of tests we are going to run. Is that what you are expecting or were you expecting something else?" It's great to be able to have that conversation with your customer early on in order to make sure, again, that, at the end when you deliver your report. You don't get a response of, "Well, this is very nice, but where is X? Where is Y? We said we wanted a report. We said we wanted an analysis. We naturally assumed that you would do all these other things, which you didn't actually assume yourself." You don't want to be in that situation. And again, the scope tree is a great tool to help to pull out that additional information and really make it be far more formal and concrete.

Now, in principle, these are fairly straightforward tools overall. Now, you can layer in a couple other things to make it be even a more valuable tool, such as you can start to describe what is the amount of resources that are needed in order to perform any task, as well as the ways that you might assess the performance of any one of these tasks. Again, another great thing to go over with a customer, because they help you to determine what makes for a good version of a deliverable, versus a bad version of a deliverable or a better version of a deliverable overall, and having that discussion with your customer, again, can be very good at helping you determine what is your overall scope of your work as a whole.

Now, in practice, when people do these, it does take a considerable amount of time and experience to do these things well. Now myself, often when I'm developing a scope tree, I try and leave myself at least a good week's time, in order to focus on developing the scope tree, recognizing I'm going to have to go through multiple iterations, and probably have to talk with teammates many times before I even come up with the very first version that I want to share with anybody externally as a part of this, or even a larger team that I'm part of. Nevertheless, it's very good to at least get some exposure to this tool at this point of the design process.

Back to Table of Contents

Tool: Scope Tree Template

Download the Tool

Scope Tree Template

As you develop an understanding of the scope of your system, it will be possible for you to begin determining deliverables. In addition to large end deliverables, you will want to recognize basic tasks that need to be completed along the way.

This Scope Tree Template is a less formal tool to help you define the scope of your project. This scope tree will help you when you want to build a timeline to reach your end deliverables. The scope tree will also help clarify with your team and stakeholders the expectations of what will be delivered.

Use this template as a starting point as you build out your own unique scope tree. This tool is provided as a set of PowerPoint slides. One slides provides a basic scope tree elements to get you started on your own scope tree. The other slide offers a general form of a completed scope tree for one end deliverable.

You may also find it useful to review the more detailed, step-by-step guide to building scope trees that can be found in the Course Resources module at the end of this course.

Back to Table of Contents

Watch: Moving Forward with Your System Design

You've reached some significant milestones in your system design process. At this stage, you've provided a good scope of what your system will do, but there is more work needed before you can say that your system is well-defined.

In this video, Dr. Schneider reviews what has been accomplished at this point and discusses the system definition work that lies ahead.

Video Transcript

With the work you've done so far, you've taken a great first step at defining what your system will really be. You figured out that it must be able to perform well in a variety of use case situations that you've identified so far. You've been able to successfully determine what are the interfaces that are necessary between your system, other systems, users, and you used tools like context diagram in order to highlight this. And you've been able to identify all this towards meeting your stakeholders' needs.

However, I want to be very clear that you should not stop here. You cannot stop here, if you really want to come up with what is a definition of what your system will be. So far, we've been able to establish a good scope of the overall project and content that we're going for, but we still need to drill down further into what are the functions that are really necessary for a system in order to accomplish those use cases. What are the requirements any valid solution must meet in order to again, establish that you are in fact meeting those needs?

Then be able to take it one step further to say, these are the ways that I would measure the performance of any solution to those problems, and thereby be able to effectively prove that your idea is one of the best, if not the best out there. And that establishment of performance on top of the requirements helps you to be able to make really strong tradeoff decisions, and be able to defend those decisions to whether it be your boss, your teammates, your customer and really be able to achieve influence without actually needing the authority to do so. And that is the real power behind systems engineering.

Back to Table of Contents

Assignment: Course Project, Part Two - Create a Use Case List, Use Case Diagram, and Scope Tree

In this part of the course project, you will use your previous work on the context diagram and use case list to build a use case diagram. You will also begin to build out a scope tree.

Instructions:

1. Open your saved course project document. (If needed, **download** it again now.)
2. Complete Part Two.
3. Save your work.
4. Submit your completed project for grading and credit.

Do not hesitate to contact your instructor if you have any questions about the project. You will add to this document as the course proceeds and will submit it to the course instructor at the end of the course.

Before you begin:

Before starting your work, please review the **rubric** (a list of evaluative criteria) for this assignment. Also review eCornell's policy regarding **plagiarism** (the presentation of someone else's work as your own without source credit).

Back to Table of Contents

Module Wrap-up: Aligning Your Scope to Stakeholder Expectations

In this module, you created a shareable use case diagram that organizes the relationship between use cases. You also built one branch of a scope tree to begin determining what tasks you need to do as you work toward your end deliverables.

At this stage, you have used a collection of tools to explore the scope of your system and helped you align that scope to your stakeholders' expectations. This well-defined scope will better prepare you as you continue on in the overall design of your system.

Back to Table of Contents

Thank You and Farewell

David R. Schneider
Senior Lecturer
College of Engineering
Cornell University

Congratulations on completing *Defining Scope*. As you have seen, exploring the interrelationships between your system and other factors through the use of context and use case diagrams can provide a great start for you in defining the scope of your project. It is my hope that these tools, along with the scope tree, have really helped you define what your system has to be and have enabled you to take a significant step toward developing your system design.

From all of us at Cornell University and eCornell, thank you for participating in this course.

Sincerely,

Professor David R. Schneider

Back to Table of Contents

Course Tools and Miscellaneous Documents

Defining Scope Checklist

Context Diagram Sample

Context Diagram Template

Use Case Diagram Sample

Use Case Diagram Template

Scope Tree Template

These foundational documents from Dr. Schneider were helpful in building the course and is being provided as an additional reference.

Defining Your System, Part 1

Step by Step Guide to Building a Context Diagram

Step by Step Guide to Building a Use Case Diagram

Written Guide to Building a Scope Tree

Step by Step Example of Building a Scope Tree

Step by Step Example of Building a Scope Tree, General Form