Defining Your System's Scope

Context Diagrams

Amazing! You've just come up with most exciting and brilliant idea for a new invention. You immediately rush to write down all of your ideas and begin designing it right away. Your passion drives you as work to make your first pieces. Weeks or even months later you discover some problems with the idea; somehow those first pieces aren't working quite the way you thought. But you know it's going to be awesome. So you focus on solving these pieces' 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:

Inventor: "Check out my awesome new creation. This is Better, (Cheaper, Faster) than anything else out there!"

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

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

Imagine you were designing a car. When people buy a car, they don't really care about all the things that make that car work. They care that it will take them from point A to point B. That it will do so safely, quickly, reliably, comfortably, and with good fuel efficiency. They may also care about other things such as its appearance and if it feels "fun". Anything that meets these needs well is in fact a good solution. As a designer, it is important to understand that even when the user asks for something specific, i.e. "a car", they are really asking for something that meets a set of needs. It is your job as a professional designer to first determine all of those needs before you start focusing in on a specific solution because otherwise we may inadvertently miss some needs or mis-weigh the importance of the various needs.

Additionally, some users may also have special needs, such as a car with additional cargo space. While for others it might be fitting into small parking spaces. So while a pick-up truck might work best for the first user, it would actually be an invalid solution for the second user, regardless of how awesome that truck is; even if it's currently the best, fastest, cheapest truck in the world. So understanding the specific users needs, *including the situations and elements they have to interact with*, i.e. fitting into small parking spaces, is an essential part of defining the problem before you actually begin defining what your product should be.

And how important are each of those needs relative to each other? We can greatly increase the cargo space, but this will most likely also increase the cost and decrease the fuel economy. Are there requirements on the minimum allowable cargo space, or fuel economy? And if there is a range of acceptable cargo space and fuel economy values that can be met, where does the optimal trade-off lie? Can you be confident that your user, and your boss, will agree with you on your decision and see the value in your decision?

This guide provides an introduction to a number of tools/techniques you can use to help discover the needs of the challenge you're trying to solve. Additionally this guide when combined with others will also help you create objective *performance metrics* that you can use to objectively measure how well your solution or any solution meets the challenge's needs; including measuring criteria like how much "fun" something is. With an understanding of the need, and objective metrics to measure any solution's performance, you can thereby prove the value of your solution to your users, boss, stakeholders, and the world!

We encourage you to read through all of the steps provided below and then decide the best order for you. i.e. The steps provided here are a common order to work with the presented tools, but every design process is different, and as you explore your design process you many find that some steps may easily overlap and happen concurrently. By breaking the process into Steps though will help you to recognize all that should be doing so as to make sure you don't accidently completely skip over any key part of the design process.

Step 0: Don’t name what you’re going to design! If you haven’t said I’m going to build a specific item yet, great! Don’t do it. Good job! You’re done with Step 0! No really, by declaring we’re going to build, say “a car”, we’ve already begun to limit our design space. Anything that we wouldn’t define naturally as a car is already invalid. A motorcycle, a bus, a plane, a bicycle etc., all may have been good, even better, solutions but would be quite difficult to develop if the problem is already defined as building a car. For example, a team was once tasked to determine the best cutting edge human search & rescue technology. The stakeholders expected a fancy satellite reconnaissance system but the team showed that the best solution for their needs was actually… a trained search dog.

The surprising truth is, and make sure you’re ready for a shock, but rarely is our first idea our best idea. But on exams we’ve always been told to go with our first instinct! Yes, in the time pressured situation of re-iterating knowledge that we’ve been recently studying intently, and where the problem we’re asked to solve has a very clear and singular correct answer that comes out of that predefined limited knowledge set, our first instinct may be a great one to go with. But when you’re working over an entire design, the first step should not be to focus on a specific solution, but first determine what are the needs.

In this guide, regardless of what we’ll be creating whether it’s a physical product, a new algorithm or control system, or a new service or operations plan, we’ll refer to it as simply “the System”. It may be uncomfortable to deal with this ambiguity, especially for many “problem solvers” who want to identify the problem as soon as possible so they can get at the meat of solving it. Despite our best efforts to think openly, we will naturally place artificial constraints on what something can or cannot be or do once we’ve named what it is. Not naming the solution at first is a helpful aid to help us be better designers and prevent us from drilling down to a specific solution too quickly.

Step 0 is complete: When your System has no name.

Step 0a: “Ahhh! It already has a name!” or “But they told me this is what I have to design!” Not to worry. This obviously happens quite often. Whether it’s the contract your boss signed, what the folks in marketing say the customer wants, or simply the Eureka! idea stuck in your mind, you still need to assess the needs behind the project. In fact, one of the best things you can do is to un-name it and just think about it as a “System”. For some very short term, small scope assignments where the answer is essentially already known by someone this could be overkill, i.e. the entire challenge is “Solve this homework problem in the back of the book so that it matches the answer key” but those are hardly design problems to begin with. But even in doing something that appears route like “Make an LED turn on when the power turns on”, you can still explore the need by asking with what and how it will be used? Perhaps you’ll learn that although you instinctively wanted to use a green LED, you shouldn’t because they already use a green LED for something else. Or perhaps it turns out that a buzzer sound is better than an LED because users won't be able to see the LED from wherever their positioned. The functional need of the LED is to alert the user that the system is turned on. The LED is just one structural solution to that need – and there are probably many others. So as you go forward with you design, try to "Think Functionally! not structurally."

Step 0a is Complete When: You recognize that you have to determine the needs behind the request. Yes, you may find out that you're constrained to create something that would qualify as an example of the structural solution you've been asked to create. However, you'll be a far better designer if you put considerable effort into exploring the needs behind the request, then if you plug ahead with just what you think they're asking for.

Stakeholder: "I asked for an iPhone! What is this!?"

Designer: "Well the phone's clarity and reception are even better than before…"

Stakeholder: "Who cares about making calls on an iPhone? What about all the other features?!"

For the common cases when you've been asked to deliver something specific, make sure you review the Defining Client Deliverables guide in detail as well for helpful ways to explore what the stakeholders are actually asking for.

Step 1: Define your Primary Stakeholders. Just as when you develop a presentation you should recognize who is your audience, when designing you should also first consider who are your stakeholders.

Stakeholder is a catch-all term used to define those who have any influence on what/how you design something but is better to be thought of as anyone or anything your design affects. Stakeholders can be people, such as users, your boss, co-workers, etc. and especially whomever you consider your client. However, they can also be things that what you're designing must work with such as pieces or parts of equipment, software, or logistic processes.

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 you want, it's about designing something that meets the stakeholder's need.

Step 1 is Complete When: You have a list of all of the primary stakeholders associated with your project. In many cases, you may also want to declare a few or even just one primary stakeholder as your client; those whom you must satisfy to in order to get the project completion approval or get paid.

Step 2: Define your Secondary Stakeholders. Secondary Stakeholders are often those that may not be directly associated with using you design but are strongly influenced by it or its outcomes, such as an overall company, a neighborhood, the location it's being used within, a wildlife population, the environment, etc

Secondary stakeholders often present constraints on your design or may at least influence the way you judge a solution's performance. They should always be considered to make sure you aren't accidently "violating" any of these stakeholders' needs, perhaps for just a small gain for your primary stakeholders, i.e. increasing the profit by a small amount for a large cost to the environment, or making your design look that much "cooler" but it becomes significantly more difficult for people with disabilities to use.

Step 2 is Complete When: You have a list of all secondary stakeholders. It's typically better to make this list longer and then trim later on if you feel that particular stakeholder really doesn't have that strong of an influence. But you may never know for sure whether a stakeholder is influential unless you include them in your investigations to begin with. It is far better to have these kinds of "surprises" early in the design when you still have time/resources to make changes.

Step 3: List all of the stakeholders that will interact with your System; begin to make a Context Diagram. This list is a good place to start to tease out the needs of your stakeholders. You can also brainstorm this list graphically by first putting a box with the word "System" in the center of your page and then around the outside placing separate boxes for each one of your stakeholders. This can either be done on a large sheet of paper or perhaps a large PowerPoint slide or even Visio diagram; whatever works best for you, although software based ones may make it easier to move things around later. An example is provided for the car design challenge in the file Context Diagram Example which starts with just one stakeholder, the passengers, but suggests others later on like the Department of Motor Vehicles.

Notice that the System box has another dashed box around it to represent the System boundary, i.e. what's inside the boundary is what you can control. What is outside are those things you as a designer don't have direct control over the design of but need to work with / satisfy instead.

Traditionally only one box, the System box, is inside the System boundary dashed box. This is done to help you continue to think at a higher level at this stage. So as much as you might want to start to break up your System into different special pieces, try to hold off for now as this will help you not name or put design constraints on your System solution too early. Besides, you could always make separate Context Diagrams for your special pieces / sub-Systems later on. So really try to have just your single System box in the boundary right now -- it may be difficult to think this way the first time(s) you try it, (as you may have spent at least most of your academic life working on more well defined problems instead of more open design problems) but you might also be surprised what ideas this helps keep open farther down the road if you do try it.

Step 3 is Complete When: You have the System box in the System boundary box and all other stakeholder boxes listed around and outside of the System boundary box. As you will most likely be moving these boxes around the page later, don't worry about making the arrangement look "pretty" until the end of Step 7.

Step 4: Determine what else your System interacts with. Using the car example, that System most likely also interacts with the road, gas stations, garages, bad weather, driveways, etc. We may later find we're able to create the flying DeLorian from "Back to the Future" and we don't need roads, but it's okay to list items here for now even if they're eliminated later.

You obviously could list a lot of things your system could interact with but as a rule of thumb, list those items that would make your System have to do something differently or something special. Even if it may not be a common situation, if it's one you know you'll have to deal with, say avoiding a deer on the road, you may still want to include "deer" or "wildlife" as a box for your System to interact with. If you're still uncertain as to whether to include a new box or not, it may be best to add it for now and you may be able to determine a reason for keeping it or eliminating it in Step 7.

Just as you did for your stakeholders, make a box for all of these items outside of your System boundary.

Step 4 is Complete When: You've added boxes of other things that the System may interact with that are outside of what you as the designer have direct control over, and connections have been added between these new boxes and the System.

Step 5: Determine how the outer box items will use/interact with your System and what your System must do with the outer box items. To start, draw a line that connects each outer box item to the Systems Box. To signify that the outer box item is going to do something to/with the System, write what they will do on the connecting line but on the end closest to the outer box item. To signify that the System is going to do something to/with/for the outer box item, write what the System will do on the connecting line but on the end closest to the System box.

If there are multiple interactions you want to write, you don't need to make a separate line as that could make things too cluttered. Instead, you can separate each interaction with a comma. If the line is still becoming too crowded, you can simply make label the line with a symbol of some kind, as shown in slide 14 of the Context Diagram Example, and then have an attached list of all the interactions that would normally be written at that symbol.

At this point, you do not need to specify exact values, i.e. you don't need to write "the System shall transport up to 7 passengers at speeds up to 120 mph". Instead, simply putting the System transports Stakeholder A, is sufficient. It is also okay to just use adjectives such as "transport *large numbers of* passengers" or "*quickly* transport" to provide some more information. Here you are just recognizing the relationship and the fact that later on the exact number of passengers and max speed/acceleration of the transporting will have to be a formalized need you specifically define.

Step 5 is Complete When: You have connected each Stakeholder box with a line to the System box and at least one end of each line is labelled with an interaction. You have created your first pass at a Context Diagram.

Context Diagrams can actually be a formal design document used in everything from toy design to designing cutting edge rockets for NASA. What is presented here is not the formal style but the ideas here stay true to the Context Diagram's purpose. To learn more about how to create formal professional Context Diagrams we encourage you to read books on or enroll in classes on Systems Engineering.

Step 6: Take a break. You've put a lot of information into your context diagram and as a result it may have become rather cluttered. Now's a great time to take a break before you try to review it as a whole or reorganize it. Good work on putting the effort in to explore your System's stakeholders and its interface boundaries!

Step 6 is Complete When: you've been able to clear your mind for a bit of the work you've done so far and you're ready to review the hard work you've done.

Step 7: Iterate on your Context Diagram by breaking up, combining, and/or removing outer box items. In reviewing your first pass at brainstorming what your System must interface with, you may notice that some of the "boxes" actually represent multiple different kinds of the same thing. For example, the weather box, is being used to represent snow, heavy rain, fog, or any other kind of inclement weather. However, if the System would have do something special or do something to a more extreme level to handle, say snow, that it wouldn't have to do for any other kind of bad weather, it would be worthwhile to add snow as a separate box or break bad weather into more boxes to properly represent the range of needs.

Boxes can be combined as well by looking for overlapping needs or "worst case" needs. For example, we could list all of the kinds of interior messes the car might have to handle: water, dirt, soda, fast food, juice, gas, milk, animal excrement, (gross I know but hey, it's a real situation the System would have to deal with) etc. But if we can find one or two such that if we met those couple boxes' needs, we would actually meet the needs of all of the sources. So for that reason we might choose "oil" representing all greasy and flammable substances and "vomit" which represents all organic, smelly, and biohazard substances, which together sound like they just might do the trick of covering the needs of all likely interior messes. Lovely, right?

Step 7 is Complete When: You have reviewed your Context Diagram for both completeness and conciseness. You have rearranged to boxes, lines, and words in the diagram so that they are easy to read.

Sometimes in order to make the problem more digestible, you may want to split the diagram into more than one; perhaps one that focuses on what you see as being the "core" problem or one those needs you feel will have the strongest influence on your design decisions. Other diagrams could focus on all of the special cases or all of the secondary stakeholders. As you are doing this for your benefit, there really isn't a wrong way to do it. Your boss may require you to submit an official version that meets more stringent requirements but that doesn't mean you can't still hold onto those versions that helped you along the way.

You can also present the same information in a matrix, but a lot of people find drawing the diagram is more useful for brainstorming. Either is great, especially if it's the form that works best for you.

Step 7a: Take another break. Especially if you've haven't thought about exploring and defining needs to this level before, this can be mentally taxing. You may be asking your brain to solve problems in a way it's never done before, so just like using muscles to play a new sport, it's natural that you might be a bit sore.

You've organized all of the main connections to your System now and you have at least helped bring these interface needs to the forefront of your problem definition more than you might have otherwise. Well done. You're on your way towards becoming a more professional designer who can really think about the whole problem's needs in order to develop a truly great solution.

Step 7a is Complete When: you've been able to clear your mind again and feel like you're ready to delve into defining the problem further.

Step 8: Create a list of the situations your system will be used in from your context diagram, a.k.a. Develop use cases. The scenarios or the ways that a system may be used are often referred to as *uses cases*. Creating a list of use cases is a fundamental part of any needs analysis. Too often designers focus on only the main way something is being used. Fortunately, your context diagram is already a great tool to help you brainstorm a range of scenarios that need to be handled.

The words you wrote on each connection are a great place to start. From the context diagram some examples you might pick are, "Driving the System", "Fueling the System", "Getting the System washed", "Maintaining the System", "Manufacturing the System", "Loading the System". Notice that in your first pass many of these may be high level, but that's okay. They typically have rather short names that focus on a single action / verb. It's also crucial to think about unintentional or even undesired use cases such as "Running over potholes" or "Surviving an accident". Each of these will help to generate a specific list of needs that might have otherwise been overlooked.

Step 8 is Complete When: you have created your first list of use cases.
You may discover some new interactions or even boxes that you hadn't originally included in your context diagram. That's actually pretty common and you can decide whether you want to update your context diagram to reflect these discoveries.

Step 9: Add in use cases from other sources. The context diagram can help you determine perhaps the majority of your use cases but there may be many that are either more internal to your system or come directly from a stakeholder request.
Examples of use cases that are more internal to a car-like System could be "System converts fuel into motion", "System measures it's speed", or "System protects against wear and corrosion".

Examples of direct stakeholder requests can vary greatly and could include, "System alerts driver of potential driver drowsiness", or "System automatically parallel parks", or "System completes the test course successfully" and many may take the general form of "System performs task X" or "System performs task X to meet criteria Y".
Some of these may have already been in or brought out through your context diagram, which is certainly fine too. Regardless, at this Step you should refer back to whatever original problem statement you were given, whether it came from your boss, an open call Request For Proposal, or your own idea. Discussions with the source of that problem statement can also be extremely valuable as hinted at in the Defining Client Deliverables guide.

Step 9 is Complete When: You have a list of use cases that covers all of the major functionalities your System will have to perform and scenarios it will have to handle.

Step 10: Refine your use cases list. You might start out with use cases like "Driving the System" but you quickly realize that that really doesn't capture the variety of tasks that are being performed, i.e. it's too high level or actually includes several different kinds of use cases. So you might add "Parking the System", "Stopping at an intersection", "Changing lanes". And special versions of "Changing lanes" might be "Merging onto the highway" and maybe "Exiting the highway". Your original "Driving the car" use case may be said to *include* these other use cases.

You may also recognize that "Driving the System" might have special cases where your System has to function in a unique manner such as "Driving through snow". So it is good to go through and recognize as many of these special cases as you think are realistic. Declaring special cases from a uses case is sometimes called *extending* that use case.

Step 10 is Complete When: You have refined your list of use cases to a reasonable level. Wait a second… what's a reasonable level? You'll hear statements like "to a reasonable level" a lot in need determining deliverable & needs discussions. A great indicator is to read ahead to Defining Your System Part II and see if you think you could do the next steps with your current set of uses cases. In short though, some good rules of thumb are that you feel that you can:

1) Clearly state the required initial and ending conditions of the use case are.
2) Describe what occurs during the use case as a series of step by step functions your system must perform, without it feeling "too long" or rather running across several use cases
3) At least a significant subset of the functions that would be performed in this use case are unique; i.e. this use case captures various needs that might otherwise have been missed if this use case wasn't considered.

Just as you might learn to recognize a pattern or opportunity in playing a sport, a musical instrument, or even a video game, you're training yourself to think and act in a new way. The first time you play though, it probably won't go perfectly (nor second time, nor the third time…). But as you continue to use these practices, you'll develop the intuition to better know how to make every Step meet your own design needs. It's really what will work best for you.

Combining your understanding of who your stakeholders are and their needs, how your System must interface with other systems/users/etc, and the variety of use case situations your System must perform within, you have begun to define what your System is. It is something that must perform well in the variety of situations you've identified. It must be able to successfully interface with other systems/users/etc you've outlines in your Context Diagram. And it must do all of this to meet the needs of your stakeholders.

Step Next: With the work you've done here you have begun to define what your System will be. It will be something that must perform well in the variety of use case situations you've identified. It will be able to successfully interface with other systems/users/etc you've outlined in your Context Diagram. And it will do all of this to meet the needs of your stakeholders.

What functions your System must perform in order to do these things and how you will measure whether your System performs these functions well are the next steps to defining what your System will be and are at the heart of Defining Your System Part II. But once you have an understanding of all of these aspects, you will have developed a professional designer's definition of what your System is. Empowering you to be able to move forward as a better designer and be confident that what you create truly will be a far more complete, efficient and higher performing solution to your challenge than you would have ever been able to have create otherwise.

.