Table of Contents

What is a subsystem? 1
How do we identify subsystems? 2
Step 1 3
Step 2 3
Step 3 4
Step 4 4
Step 5 4
Step 6 5
Step 7 5
Deliverables 6
However 6

Subsystem

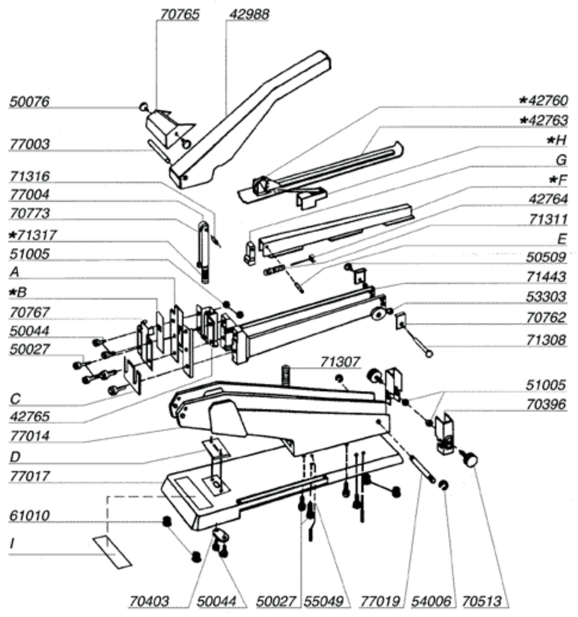
A *subsystem* is a system that is an element of a larger system.

What is a subsystem?

A *subsystem* is a system that is an element of another system. It is “smaller” than the containing system, in that it provides only some of the functions that the larger system provides. It is wholly contained by the larger system; no subsystem can cross the boundary of its containing system.

Let's consider a simple product. Refer to the figure to the right, showing an exploded view of a *Roma stapler*. Let's look at one possible system hierarchy for this product:

Fig. 1: Exploded view of a Roma stapler.



Roma Stapler: (source)

- structural system
  - includes all the large parts and the fasteners that hold them together.
- stability system
  - includes parts 77017, 77014, 42988, and all the fasteners needed to hold them together.
- stapling system (actually doing the stapling)
  - includes parts 42988, 70773, 42765, 77014, D, G, A, B, 70767, 50044, 50027, C, 42765, 77017, and the fasteners needed to hold them together.
- staple-loading/storage system
  - includes parts 42760, 42763, F, 53303, and the fasteners needed to hold them together.

Notice that some parts are obviously in many systems at once. Notice also that there are two kinds of parts: *primary* parts that are mainly responsible for providing function, and *secondary* parts (e.g. the fasteners) that are really only there to ensure the primary parts are working right.

Partitioning a system into subsystems has mostly to do with managing complexity. For instance, if one started with a Boeing 777 jet liner and moved directly to the design of the roughly three million parts that make it up, the task would immediately become intractable.

Breaking a system into subsystems based on function provides three advantages:

1. It helps manage the inherent complexity of the problem, so that designers, engineers, and others can better analyze and test the design to ensure it performs according to the requirements; that is, it helps designers and others understand what they're doing.
2. It helps designers visualize how mass, energy, and information can flow within the product.
3. It helps manage the project by providing sensible ways to organize and allocate resources, which in turn leads to lower costs, faster development times, and improved product quality.

The purpose of identifying subsystems, then, is to partition the total set of requirements into groups, each of which can be treated as a smaller, simpler problem to solve.

This approach is particularly useful because it can help design team members communicate with one another. For example, say two members of your team are assigned to further develop the `cab` subsystem of an elevator, and another two members are assigned to further develop the `lifting` subsystem. Whenever a decision is made that impacts the function of lifting a mass, no matter who made the decision, then those two and only those two sub-teams should exchange information about that decision. This can streamline project management and keep the project moving quickly, without jeopardizing the effectiveness and quality of the resulting design.

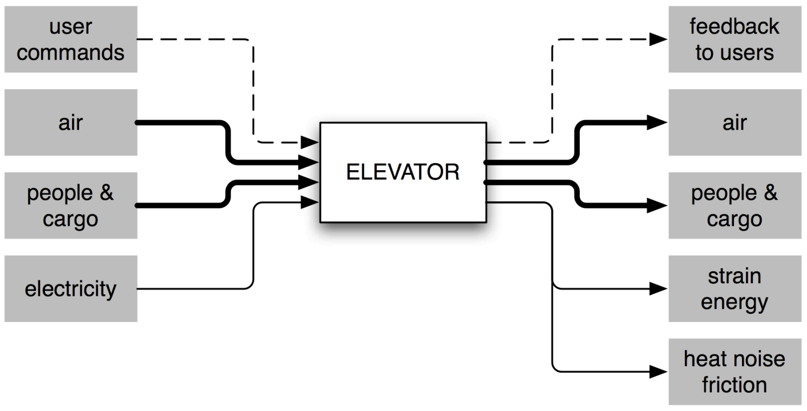
This approach is very common in industry.

You begin system design by identifying the product system, because the product system is the boundary between what you know (the requirements) and what you don't know (the rest of the design). Once you have defined the product system, you can open up that "black box" and start defining the lower/smaller level of systems; these are the subsystems of the product system.

This means that to identify subsystems, you need a rather crisply defined system, as well as a well-defined product strategy to guide decisions about the types of subsystems that make the most sense in a given situation.

How do we identify subsystems?

Fig. 2: A system diagram of the elevator design problem used on this page.



Consider the following example:

> Design an elevator to be used in a 3-storey shopping mall, that can accommodate people and the typical cargo one would expect in a mall. The elevator must provide to riders a view of the mall while the elevator is in operation.

figure 2 shows the initial system diagram for this elevator. Assume that a PRS provides the following functional requirements:

- protect passengers: this includes keeping them safe and comfortable, as well as accommodating emergency situations.
- lift a mass: and lower it of course.
- provide a view: as stipulated in the design problem.
- interact with passengers: to accept commands and provide appropriate feedback to human users.
- control machinery: the elevator will doubtlessly have various subsystems; the whole must behave in a controllable way; this includes internal self-control (noticing what it should do under, for instance, emergency situations) and external control (for safety, security, maintenance, etc).

To determine what subsystems you need in your design, you can build a *System Identification Matrix* (SIM).

Step 1

Set up a table similar to the one shown below. In the FR column, list all the top-level FRs for the product as a whole (from the PRS). List only the FRs. Systems are transformers and thus are all about function (**not** characteristics or constraints).

Functional Requirement	Subsystems
protect passengers	
lift a mass	
provide a view	
interact with passengers	
control the machinery	

Step 2

Examine the first FR and identify a subsystem or subsystems that will satisfy it, or at least contribute to it significantly. Create a column for that subsystem and add an appropriate mark in the corresponding cell.

In the next table, the first FR we consider is `protect passengers`. The `cab` subsystem is specified for this FR. The X in the top right cell indicates that the subsystem provides the function.

Note that we've used a very concrete noun, `cab`, to label the subsystem that contains people/cargo to be moved by the elevator. This is acceptable, so long as it remains clear that the label is just an abbreviation and does not necessarily mean the subsystem will be implemented as typically found in other elevators (or, worse, taxicabs!)

Functional Requirement	**Subsystems**
	Cab
protect passengers	X
lift a mass	
provide a view	
interact with passengers	
control the machinery	

Step 3

Go to the second listed function and identify a subsystem for it.

In this case, we identify the `lifting system` to provide the function of lifting a mass, so we create a new column for the `lifting system` and put an X under the `lifting system`.

Note: it is possible that you do *not* need to add a new system for every FR. Take advantage of the fact that one system can provide multiple functions.

Step 4

Go back to the other existing subsystems you have so far to see if any of them also contribute to the second function. In this case, the `cab` system is needed to help deliver the function of `lift a mass` - after all, the people and cargo lifted in the elevator are *in* the cab.

In the example, the `cab` subsystem also contributes to the FR `lift a mass`. See the table below.

Functional Requirement	**Subsystems**	
	Cab	Lifting System
protect passengers	X	
lift a mass	X	X
provide a view		
interact with passengers		
control the machinery		

Step 5

Move on to the third function, identifying more subsystems, and continue until all the functions have been treated, per steps 1-4, above. Refer to the table below.

Functional Requirement	**Subsystems**				
	Cab	Lifting System	Shaft	User Interface	Control System
protect passengers	X				
lift a mass	X	X			
provide a view	X		X		
interact with passengers	X		X	X	
control machinery					X

Notes:

- The cab and shaft interact with the passengers by letting passengers & cargo on and off the elevator.
- Only the control system controls the machinery; this means that we distinguish all the control elements of all the elevator machinery as being part of one, integrated system, even if they are not physically connected together.
  - This is a decision taken by the design team, and is not the only way it could have been done.

Exercise for the Reader What alternatives can you suggest to this approach? Which alternative do you think is best, and why?

Step 6

When all the functions have been assigned to at least one subsystem, revisit each column (that is, each subsystem) and verify that there is a reasonable justification that each subsystem contributes to all the marked functions. Discuss it in your team; make sure everyone agrees that the X marks are where they should be.

In the example, we may revisit the cab subsystem and come up with a reasonable justification that it contributes to all functions except `control`, which might appear a little asymmetric or “lopsided” to your team.

Sometimes this kind of asymmetry is an indication of a deeper problem with the design. It can be very useful to spend a little extra time making sure that everything about such asymmetries still make sense, and re-checking the PSS and PRS to make sure there isn't a problem in decisions you made earlier.

Step 7

Fig. 3: A system diagram of the elevator having identified some subsystems.

Finally, make sure you've documented your work, by

1. keeping the SIM “on file,” including notes on the rationale behind your selection of subsystems and

your grouping of functions into subsystems, and
2. adding new nodes to your system diagram (see figure 3) to represent the subsystems.
    - The new nodes are put inside the box representing the main product system because these subsystems are entirely inside our product. Their location within the larger system don't matter.

Deliverables

See the **Deliverables** section of the PAS.

However

The greatest danger in identifying subsystems is that you will end up identifying *too many* subsystems, each of which is *too specific* for a given stage of the design's coevolution.

For instance, in the elevator example above, you might have ignored the `cab` system and rather had a `passenger containing system`, a `passenger command system`, a `maintenance command system`, a `feedback system`, an `emergency braking system`, and so on. The problem here is that you would have been treating the problem at too fine a level of detail. Skipping levels of details can lead to unjustified decisions. For instance, replacing the `cab` system with (among other things) an `emergency braking system` rather implies that that system is in the cab. That's a decision. But how do you know the system isn't part of the `shaft` system? In fact, you don't know that at all.

Similarly, it is also rather easy to identify *too few* subsystems that are each *too general*. For instance, say we were designing a bicycle and had decided it required a `drivetrain` system that includes both accepting power from the human rider (i.e. the pedals), transmitting that power to the driving wheel (i.e. a chain, or belt, or other system), and managing the torque that is actually transmitted to the driving wheel (i.e. a gearbox of some sort). The reason why having too few systems will become apparent during concept design; the sort version is that these functions must all occur in parallel and simultaneously for the `drivetrain` to work. Where possible it is best to distinguish between simultaneous and dependent functions as early as possible, so long as doing so does not require you to make any unnecessary decisions that are unjustifiable so far.

If in doubt, ask for assistance.

analysis, systems