Table of Contents

System Diagram

A system diagram is a visual model of a system, its components, and their interactions. With supporting documentation, it can capture all the essential information of a system's design. There are many variations of diagramming style that all fall under this rubric. The style presented here is intended to be optimally consistent with the rest of this courseware.

What is a system diagram?

A *system diagram* is a visualization of a system as a flow-chart-like diagram.

A system is marked by a box. The box marks the boundary of the system and completely contains it. The boundary need not be physically distinct. We place a label in the box identifying the system.

Since systems transform their inputs into their outputs, we use *labelled arrows* to represent specific interactions between systems:

- mass transfers are noted with thick arrows;
- energy transfers are noted with thin arrows; and
- information transfers are noted with thin, dashed arrows.

Inputs are arrows coming from outside the system, and either:

1. ending at the system's boundary if no subsystems have been established yet; or
2. passing through the subsystems and eventually exiting the system, if subsystems have been established.

Fig. 1: A simple system diagram.

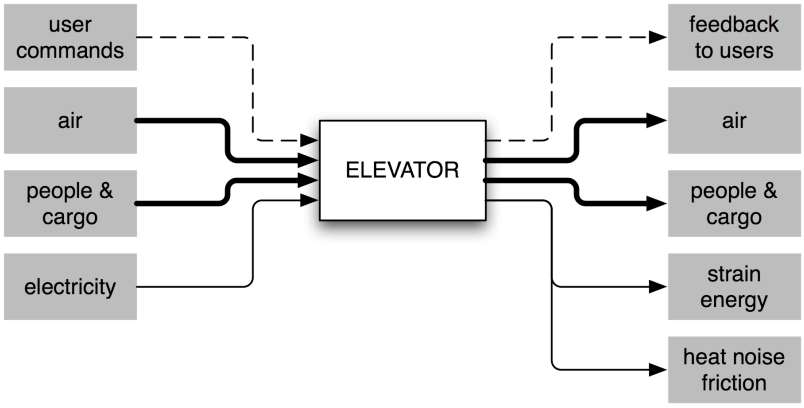


figure 1 is an example of a very simple system diagram. It shows the very beginning of a system design for an elevator. We have marked the elevator system as a labelled box, and indicated the principal inputs and outputs as different types of arrows.

Some notes about figure 1 are in order.

- **System design is a top-down (or outside-in) process:** We start by defining what is on the *outside* of the boundary that marks the system we have to design, because we cannot design a

system that will “fit” in a situation without first understanding the situation.

- **Air is distinguished from people and cargo:**

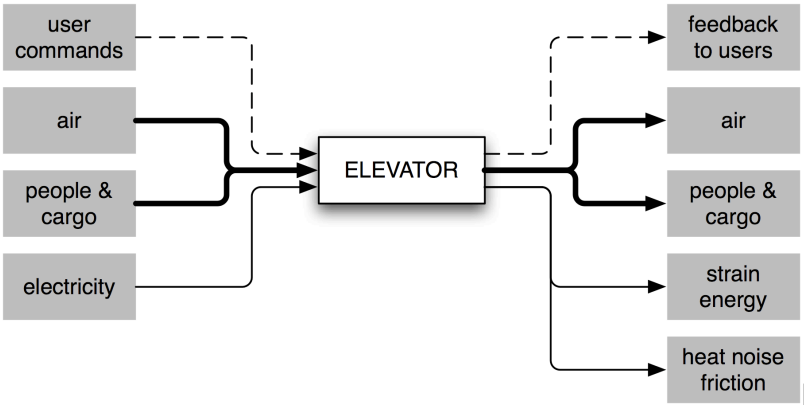


Fig. 2: A flawed system diagram for the elevator. If pursued literally, this design could result in the asphyxiation of the passengers. All three of these inputs (outputs) are treated as mass inputs (outputs), but air is treated as a separate input (output) stream. This is because of the intention of those inputs. Treating air, people, and cargo as a single input stream (as in figure 2) implies that the only time air can enter (exit) the elevator is when people and cargo enter (exit) it. This would be problematic at best. By making a separate interaction stream for air, we are embedding in our model the requirement that air must be treated entirely differently than people or cargo. Remember: it's the simple mistakes that get you.

- **User commands (and feedback) are purely functional:** At the beginning of a design project, it is always good to maintain a functional perspective. It doesn't matter if the elevator is operated via buttons, or vocally, or via some remote control - the key is to know that there will be *information* provided to the elevator that will guide its operation from moment to moment. (Likewise for the feedback provided by the elevator regarding its status.)
- **Inputs and outputs “line up”:** Notice in both Figures 1 and 2 that corresponding inputs and outputs are arranged to line up vertically on the left and right sides of the diagram. This helps comprehension of the diagram by others. Remember that designing is, among other things, an act of communication, so making diagrams that are easily understood is an important goal.
- **One output stream for waste:** There is only one waste output from our system, which splits in two *outside* the system. This means we as designers are not responsible for that split (since it happens outside what we are designing). We *are* responsible to create as little waste as possible, and, depending on circumstance, our clients or users may be responsible for dealing with that waste, so we are responsible for being honest about the waste our designs create. *This is only one alternative.* For example, when designing a nuclear power plant, one would likely have a number of different waste output streams because some would obviously be more significant than others, and would have to be considered in entirely different ways.

Since systems are composed of elements that may be other systems, a box representing one system may include smaller boxes for its subsystems.

Fig. 3: A more complete system diagram of the elevator.

Figures 1 and 2 show an *initial* system diagram, one in which the inputs and outputs have been established (in accordance with the requirements), but the design of the product itself has not.

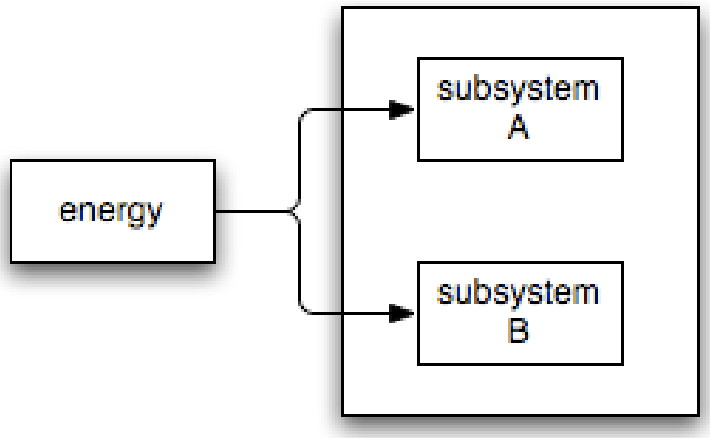
Once a system has been designed, one can expect to generate a system diagram more like figure 3.

The nodes (boxes) inside the main system represent the functional subsystems of the elevator.

The arrows representing flows of mass, energy, and information have been extended from the boundary of the elevator to specific subsystems. Some points of interest about this type of diagram:

- **Any subsystem with an input must also have at least one output, and vice versa.** A system with one or more inputs and no outputs cannot exist. Make sure you can follow all inputs to some corresponding output, whether at the system or at the subsystem level.
- **An input that passes through a system without also passing through at least one subsystem is irrelevant to the system.** Such an input would be untransformed by the system; it is therefore unnecessary and should be removed from the design.
- **Use labels on arrows to explain flows.** Some flows might be obvious, such as the various `electricity` flows. Others require more explanation (e.g. `commands` versus `feedback`. It is important to be as specific as possible, without making assumptions about design decisions that have not yet been made.

Fig. 4: An energy input split outside a system.



> ***Exercise for the reader:*** Take some time to study Figure 3, and try to identify as many shortcomings or ambiguities as possible. Discuss them in class.

- **The diagram is still not complete.** There is still information missing from this diagram - the height of the building, for instance, which would set a limit on how high the elevator would have to be. Presumably, this constraint is available in the PRS as a requirement. Whenever possible, one should include as much information as possible in the system diagram. Of course, to keep the diagram from getting cluttered, one might use various “short cuts” - for instance, one might assign a number to every requirement, and then reference requirements in the system diagram by noting their numbers only. In a computerized system, one might embed cross references that would, when followed, move one between a system diagram document and a requirements document.
- **The location of flow splits matter.** Notice in Figure 3 that the elevator system has a single electricity input, which then splits three ways *inside* the system. By specifying the system this way, the designers are asserting that they expect only one electricity source to be made available to the elevator, and that they will design the system to split the electricity safely and properly. This is as opposed to Figure 4, in which an energy flow splits outside the system boundary; this means that the system designers expect two energy sources to be made available to them (i.e. they are *not*

responsible for splitting a single energy flow to suit their needs).

Fig. 5: A sample system architecture diagram for a toaster.

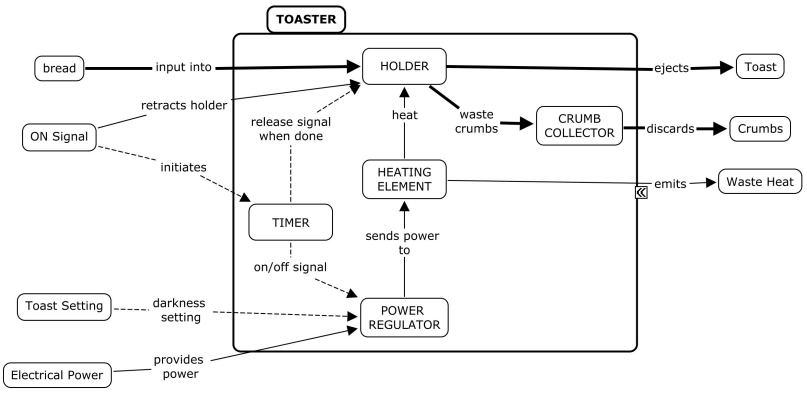


figure 5 shows an entirely different example of a system diagram. In this case, the product is a toaster (and was created with a different software package than the preceding figures, which accounts for the stylistic differences). In this case, there are far more annotations on the arrows and as such this constitutes a better diagram. Still, this diagram is not perfect.

> ***Exercise for the reader:*** Take some time to study Figure 5, and try to identify as many shortcomings or ambiguities as possible. Discuss them in class.

Why use system diagrams?

Diagrams are "richer" than plain text

As the saying goes, a picture is worth a thousand words. A good diagram can capture a huge amount of information in a very small space. This makes diagrams a *dense* way to represent information.

Diagrams can also represent non-linear information, such as the multi-dimensional relationships between systems. Text is *linear*, which prevents it from representing non-linear information efficiently.

Therefore, diagrams are a *richer* form of information capture than simple text.

Recording decisions

One very important reason for creating a specification of a system is to record the decisions made during its design. Besides the legal requirements - being able to account for your actions as an engineer - documentation provides one with the means to remember what one did. It is not likely that one will remember everything one does, and why one did it. A record of decisions taken can work as a memory aid, so that one can work in the future knowing sufficiently well what one did in the past and why.

Communicating information

Diagrams, if properly constructed, are excellent mechanisms to communicate complex information to others, much like a well-made technical drawing. Designing is, among other things, an act of communication, so making sure one has an accurate and relatively easy to read account of design decisions is important for communicating that information to others.

How does one construct a system diagram?

A system diagram captures the results of design activities. While it is possible to use other means to record one's work during designing and then generate the system diagram at the end of the systems design task, it is not recommended.

System diagrams can be used as thinking tools. By studying a system diagram, one can discover problems and shortcomings of the design it represents, and at the same time construct a final document that will capture the entire design.

Thus, system diagrams should be constructed during systems design, and used as the basis for refining and specifying various aspects of the system.

One may use pencil and paper, or software, to build a system diagram. Software-based diagrams are easier to maintain and to produce in a neat and concise form. The final decision is, however, up to each design team.

Software for creating system diagrams

Salustri's recommendation is draw.io. It's free, works online, and has standalone versions (also free) for most platforms. Also, it works very well.

- Visio (approx $200 USD; Windows)
- SmartDraw (approx $70 USD, free trial; Windows)
- CMapTools (free for teaching & learning purposes; multi-platform)
- Dia is an open-source (i.e. free) replacement for Visio from the venerable GNU project (but it is not yet as powerful; multi-platform)
- yEd (free; multi-platform)
- GoVisual (free; Windows)
- Diagram Designer (Windows; free)
- Omnigraffle (Mac only; approx $60 USD, free trial).
- Graphity by yWorks is free, runs in your browser, but allows you to save files to your local computer.

There are also some online web-services for diagramming, many of which have free trial periods - Creately, and Gliffy being the most popular.

Summary of System Diagram Rules

The development of a system diagram is often done at the same time as the identification of subsystems because the two tools complement each other. The PAS forces the designers to think in terms of the operation of the product, which can often bring to light shortcomings in the system identification chart.

Here are the rules for PAS diagrams in a nutshell.

Boxes for systems: Each box, or *node*, marks a subsystem. Generally, all boxes look the same, and contain the name of the subsystem.

- There is one large central node to denote the overall system being designed.
- Nodes within the system will represent *subsystems* only.

Inputs and outputs: Every system has inputs *and* outputs by definition.

- Inputs to the system will be shown as nodes on the left side and outside the system.
- Outputs from the system will be shown as nodes on the right side and outside the system.

Interactions and links: An *interaction* is shown as a transfer of mass, energy, or information between system elements.

- System interactions will be shown with links (arrows) connecting nodes together.
- Every link between subsystems shall have a *short descriptive label*.
- Links between system inputs and subsystems, and between system outputs and subsystems, do not need to have labels (but they may be added for clarity).
- One link can have one root and many heads. The split in the arrow is important, so placement of the split inside or outside the system matters.

Three arrow types: By convention:

- thick arrows represent flows of *mass*,
- thin arrows represent flows of *energy*, and
- dashed arrows represent flows of *information*.

Connectivity matters: Every subsystem must:

- connect to at least one other subsystem, or
- process a system input to a system output.
- Also remember:
    - Make sure you account for all significant waste outputs of your system.
    - Everything that goes in must come out somehow.
    - Only system inputs & outputs can come from / go to the environment.
    - Use different arrows to mark substantively different inputs or outputs, even if they are of the same type.

Interface specifications: Each arrow (interaction) represents an interface between system elements. Be as specific as possible on the nature of that interface, without deciding how the interface will be

implemented – unless you can justify your design decisions.

- Interfaces become requirements at the next lower level of systems design.

Example: Medical Stretcher

Here is a bad example of a PAS for a medical stretcher.

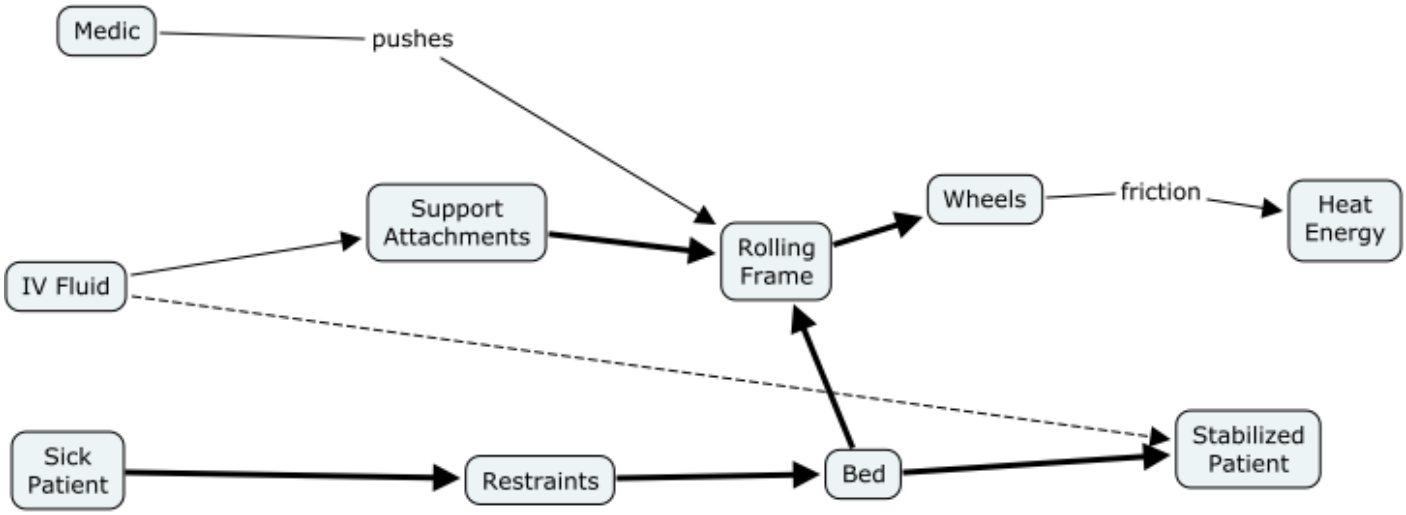


Fig. 6: Example of a //poor// PAS for a medical stretcher.

Here is a better version of the same thing.

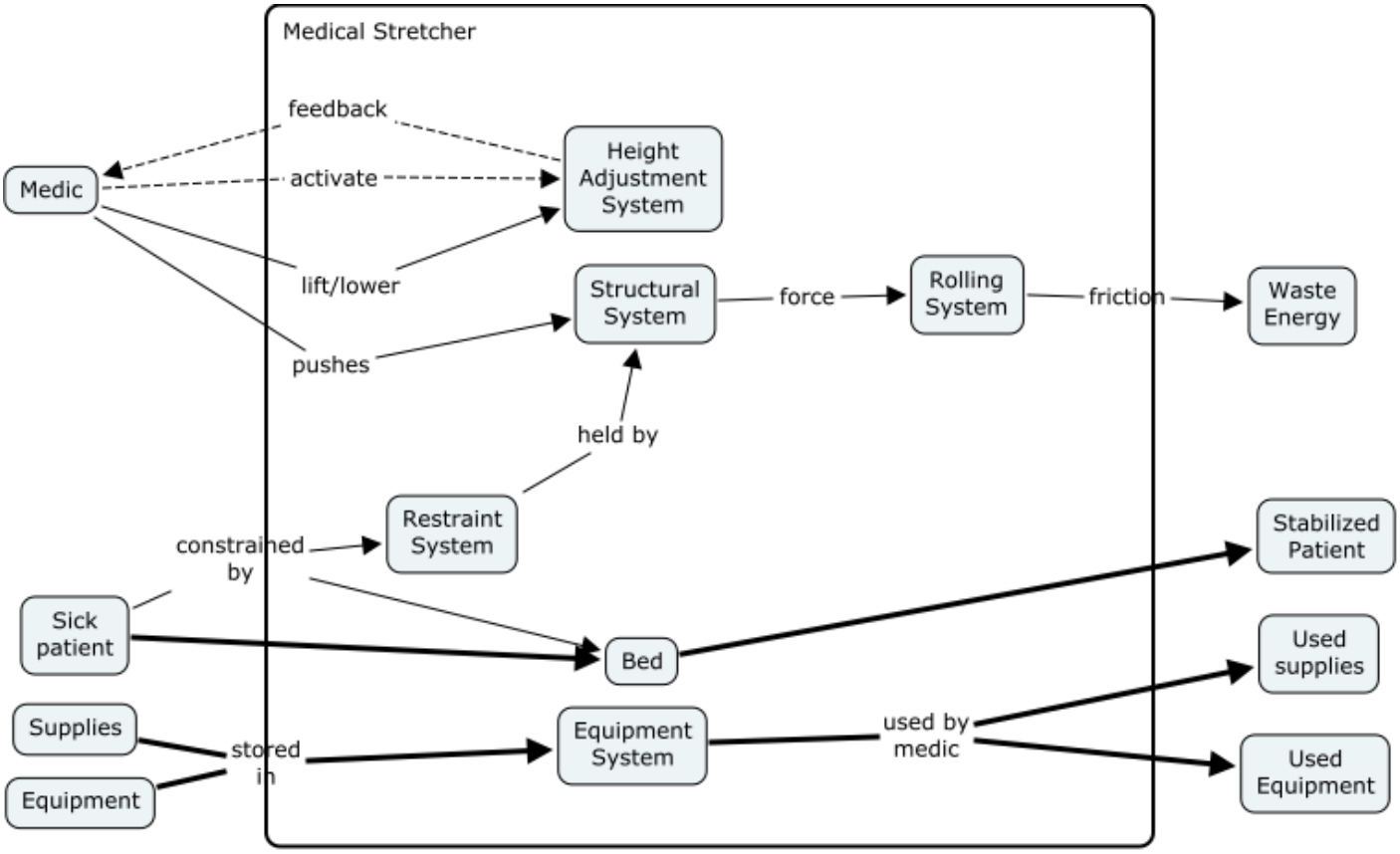


Fig. 7: Example of a //better// (but not perfect) PAS for a medical stretcher.

> ***Exercise for the reader*** Explain why the bad PAS is bad, and how the good one is better. How could you improve the good PAS even more?

systems