System functions, requirements and resources

In this chapter, we examine the kind of functions systems can have. We also look at how requirements are generated, and how we should deal with resources.

1 System functions

1.1 The Functional Flow Diagram

Every system has functions: It should do certain things. To accomplish these functions, several steps (sub-functions) need to be taken. A good way to visualize this, is by using a **Functional Flow Diagram** (FFD). An example of an FFD for an automobile can be seen in figure 1.

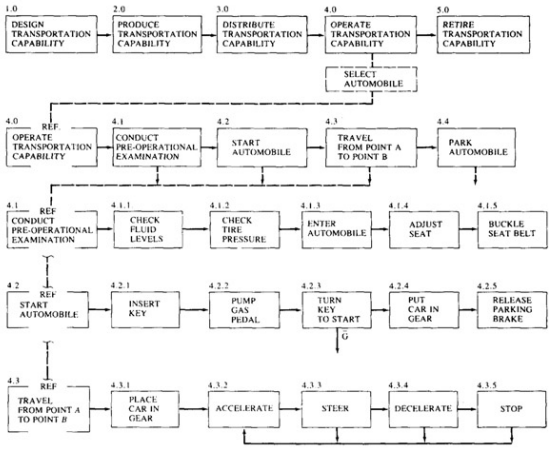

Figure 1: An example of a Functional Flow Diagram for an automobile.

1.2 Time Line Analysis

The FFD shows the sequential relationship between functions. But it does not show the actual duration of these functions. **Time Line Analysis** (TLA) does provide this functionality. When doing this analysis, we represent functions by blocks. The width of the block then denotes the duration of the function. An example of a TLA, for the European Robotic Arm (ERA), is shown in figure 2.

1.3 Functional Breakdown Structure

Quite often, a function isn't as easy to understand as you might want. In that case, a **Functional Breakdown Structure** (FBS) comes in handy.

In an FBS, a function is split up into several sub-functions. All of these sub-functions must be performed, to perform the actual function. (The FBS is therefore an 'and-tree'.) For example, let's examine the FBS shown in figure 3. Function A will only be done correctly, if the functions B, C, D and E are all performed correctly.

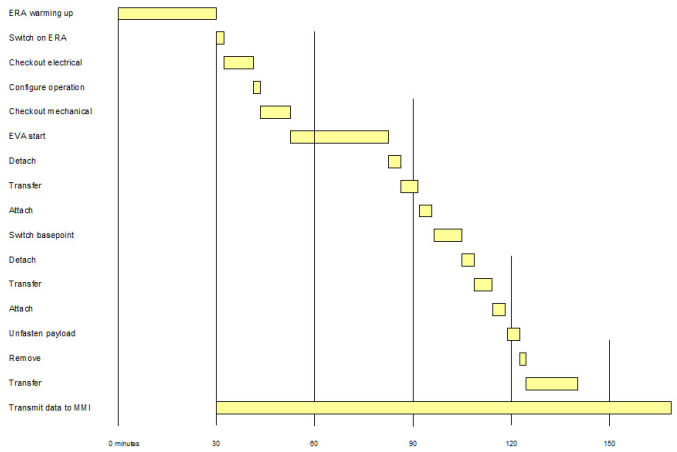

Figure 2: An example of a Time Line Analysis for the European Robotic Arm.

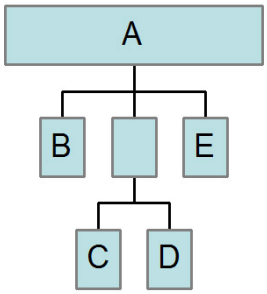

Figure 3: General form of the Functional Breakdown Structure.

1.4 Relations between functions

Let's suppose we have N functions. There are often relationships between functions. One way to display these relations, is by using an N^2 **diagram** (also known as an **interface diagram**). The general form of an N^2 diagram is shown in figure 4.

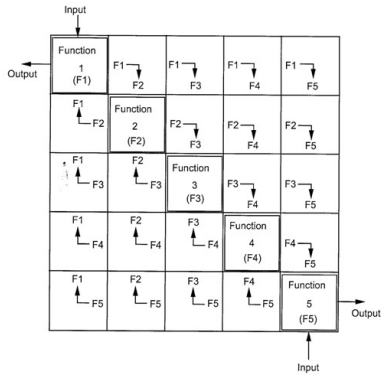

Figure 4: General form of the N^2 diagram.

An N^2 diagram is, in fact, an $N \times N$ table. On the diagonal, we place the functions. We use the remaining fields to display the relationships between functions. In those fields, we indicate which functions send what data (as input/output) to what other functions. By convention, input is displayed vertically and output is display horizontally.

You may wonder, how does this displaying work? Well, let's suppose function 1 sends data to function 2. In this case, we write this down in the second column of the first row. If, however, function 1 does not send anything to function 2, the corresponding cell remains empty.

2 Requirements

2.1 What are requirements?

Let's suppose we're designing a system. The **requirements** state what our system should do. When designing the system, we should keep these requirements in mind a lot.

We can sort requirements, based on their importance. **Key requirements** are requirements having an importance that is above average. There could also be requirements which have a very strong influence on the project. Such requirements are often called **driving requirements**. Finally, **killer requirements** are requirements driving the project to an unacceptable extent. In other words, they are virtually impossible to reach.

Requirements can be about a lot of things. For example, they can be about...

- Functions
- Configurations
- Interfaces
- Looks
- Environmental influences
- Quality
- Operation
- Support

- Verification

2.2 Generating requirements

How do we get requirements? The requirements are usually set by the customers. They are derived from the the POS and/or the MNS. Generating requirements is generally a difficult thing. There are therefore tools that can help you with it. The **Requirements Discovery Tree** (RDT) is one of them. An example of such a tree can be seen in figure 5.

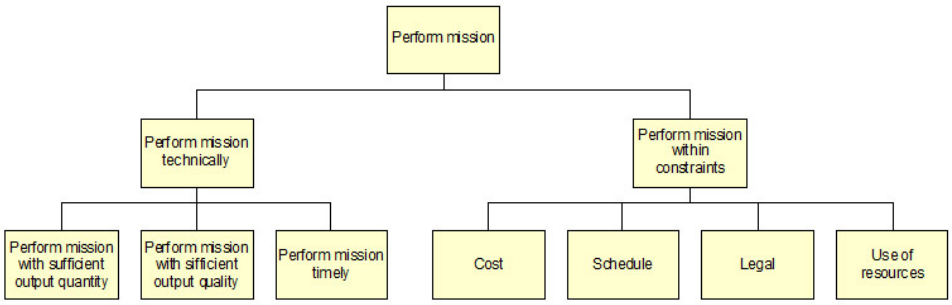

Figure 5: A general example of a Requirements Discovery Tree.

To generate an RDT, we start with the main requirement. This requirement is split up into separate sub-requirements. Each of these sub-requirements is then again split up. And this continues, until the requirement tree has sufficient detail. In this way, the whole Requirements Discovery Tree is generated.

2.3 The Requirements Traceability Matrix

When generating the requirements, we have seen that requirements often originate from other requirements. In fact, requirements usually have parent and children requirements. The relation between these requirements is summarized in a **Requirements Traceability Matrix** (RTM). By the way, the RTM looks quite a lot like the Requirements Discovery Tree.

The RTM is quite handy to check the requirement structure. If a certain requirement has no parent, then something has gone wrong. It could, on the other hand, also occur that a rather general requirement does not have any children. In this case, further requirements need to be added.

2.4 Valuing requirements

Let's suppose we have a list of requirements. Of course, some requirements are more important than others. To determine which requirements are more important than others, we can use a **Quality Function Deployment** (QFD) diagram. The general shape of a QFD diagram is shown in figure 6. A QFD diagram is often also called a **house of quality**, due to its house-like shape.

Let's suppose we're generating the QFD diagram for the requirements of a wooden desk. To generate it, we have to take certain steps.

1. First we write down the **product attributes**. These are the properties of the product, as seen by the customer. (The looks of the desk, the ease in use, the strength, etcetera.)
2. We then indicate the importance of the product attributes, according to the customer. This goes on a scale from 1 to 5. (The customer could, for example, want a very efficient desk. Then the ease in use would get a 5, while the looks would only get a 2.)

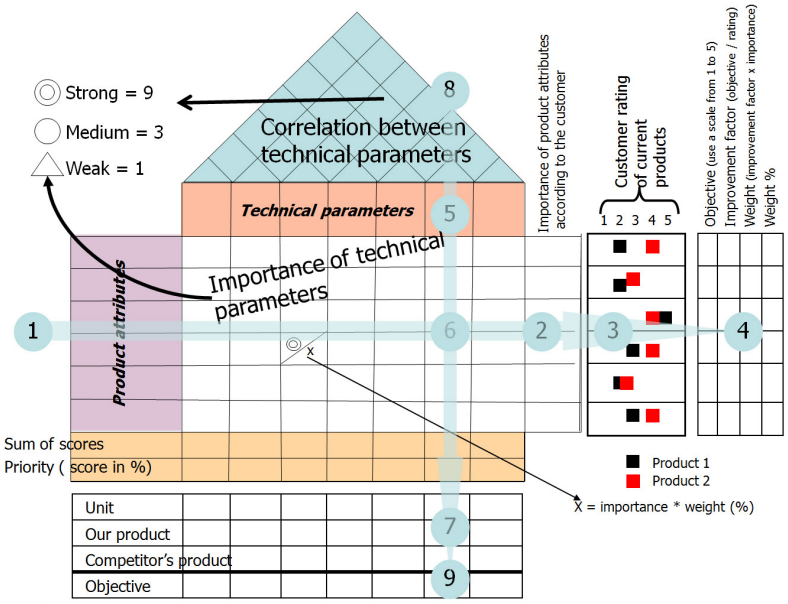

Figure 6: The general shape of the Quality Function Deployment diagram.

3. We will be comparing our product design to other similar products. To do this, we need to find out what our customer thinks of those other products. We do this for all product attributes. (So, for example, we have to know what our customer thinks of the looks, the ease in use and the strength of that brand new IKEA desk.)

4. We continue by setting the objectives for our product. This is again on a scale from 1 to 5. (How good do we want our desk to look? How strong should it be?) Once we have done this, we can also fill in the remaining columns on the right of the QFD. This eventually gives us the **weight** of the product attributes.

5. We then start to write down the **technical parameters**. These are the parameters, as seen by the designers. (Examples are the thickness of the wooden planks, the number of supporting beams, the wood type, the paint quality, etcetera.)

6. It is time to fill in the middle part of the QFD. Here, we insert the effect of the technical parameters on the product attributes. This can be either strong (9), medium (3), low (1) or non-existing (0). (For example, the thickness of the wooden planks effects the strength a lot (9), but it effects the looks only a bit (3 or 1). Also, the quality of the paint effects the looks quite a bit (9 or 3), but it does not effect the strength at all (0).) Once we know the effects, we multiply them by the weight (determined at step 4) to find the **importance**.

7. The hardest part is now over. To fill in the fields of the bottom rows, we simply need to add up all the values of the columns above them.

8. We can also indicate the **correlation** between the technical parameters. They can effect each other in a strong way (9), a medium way (3) a light way (1) or not at all (0). We do this for every pair of technical attributes. (For example, the number of supporting beams in the desk effects the thickness of the wooden planks. More supports means that the planks can be lighter.)

9. Finally, based on the results that were found, we give actual values to the technical parameters. (We can, for example, decide that we want wooden planks of exactly 1 centimeter thick.)

3 Resource Management

3.1 Technical Resource Budget

Let's suppose we're designing a system, like an aircraft. This design has several **technical resources**, like mass, capacity, availability, and so on. The set of all these parameters, which determine the success/failure of the project, is called the **Technical Resource Budget** (TRB).

During the design process, the technical resources change. However, they usually change in a bad way. (The mass usually increases during design.) It is our task to make sure they don't increase too much. (We should not consume too many technical resources.) The process of doing this is called **Technical Performance Measurement** (TPM). The technical resources involved in this are therefore also called **TPM parameters**.

3.2 Applying Technical Performance Measurement

How does Technical Performance Measurement work? To apply TPM, we need to set four values.

- First we set a **specification value** for our TPM parameters. This is the maximum our parameter may be. For example, we can say that the mass of our aircraft should be at most 100 tons.
- The next step is to construct a **target value**. This is the target we are strifing for. It is slightly below the specification value. For example, it could turn out that (during aircraft design) the mass often turns out to be 25% higher than planned. (This percentage is called the **contingency**.) In this case the target value for the mass will thus be 80 tons. (If the mass now rises by 25%, we're still not above the specification level.)
- The third step is to find the **actual value**. For example, it could be that (according to our current design) our satellite will weigh 88 tons.
- Finally, we determine the **current value**. It is equal to the actual value, plus the contingency. In our example, we would have a current value of 110 tons (being 25% above 88 tons).

When the current value is bigger than the specification value, there is a problem. There are three things that can be done. We could either improve our design, change the specification value, or change the uncertainty in our estimates (the contingency).