There is a nice way in which to visualize market segments. It is called the **Multidimensional Market Definition** (MMD). It consists of a 3D graph. The three axes of this graph indicate the customer, the function, and the technology/material of the product. An example of an MMD (applied for the bottling market) can be seen in figure 4.

Let's take a closer look at the MMD. The positions in the graph are so-called **cells**. Each cell indicates a certain combination of customer, function and technology. A cell, or a combination of cells, thus indicates a market segment.

3.2 The Supply Chain

Products don't just arrive at the market. They have to be supplied to it. This supplying can be visualized by a **Supply Chain**. An example of such a Supply Chain can be seen in figure 5.

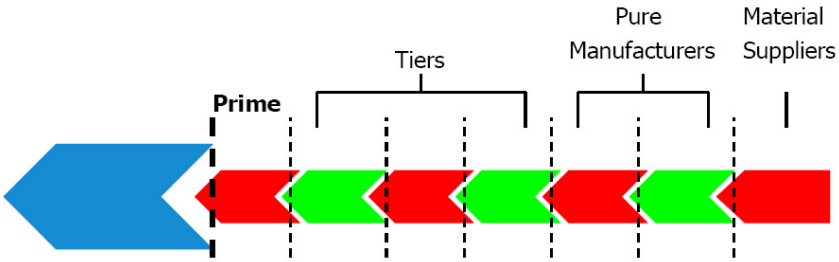

Figure 5: An example of a Supply Chain.

In a supply chain, the steps a product goes through, before it reaches the market, are displayed. First wer start at the **raw material suppliers**. The raw materials are then turned into basic parts by **pure manufacturers**. These basic parts are eventually assembled, and ready to be supplied to the market.

3.3 Market Analysis

Let's suppose we know what our market looks like. Now we want to position our product onto the market. To do that in an optimal way, we need to perform a **Market Analysis**.

An important tool to perform a market analysis, is the **SWOT analysis**. We want to give our product a certain position on the market. In a SWOT analysis, we look at the **Strengths**, the **Weaknesses**, the **Opportunities** and the **Threats** caused by this. If we know these four things, we might be able to improve the position of our product on the market.

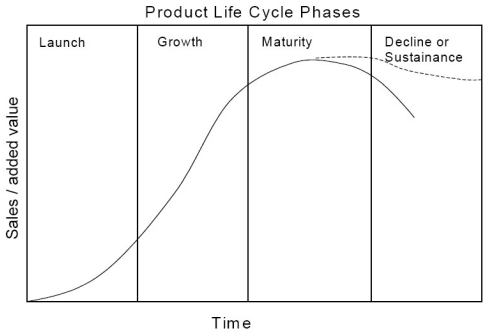

Figure 6: The general shape of the Product Life Cycle graph.

There is another way to examine the position of our product in the market. We can also look at the **Product Life Cycle** (PLC). This cycle describes the sales of the product, while it is on the market. An example of a Product Life Cycle graph can be seen in figure 6.

Let's take a closer look at the PLC. Initially, in the **introduction/launch stage**, sales are low. However, growth is accelerating, and soon we reach the **growth phase**. When this growth is starting to halt, we reach the **maturity phase**. Finally, the product either leaves the market (**decline**) or will have a steady sale level (**sustenance**).

4 System functions

4.1 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 7.

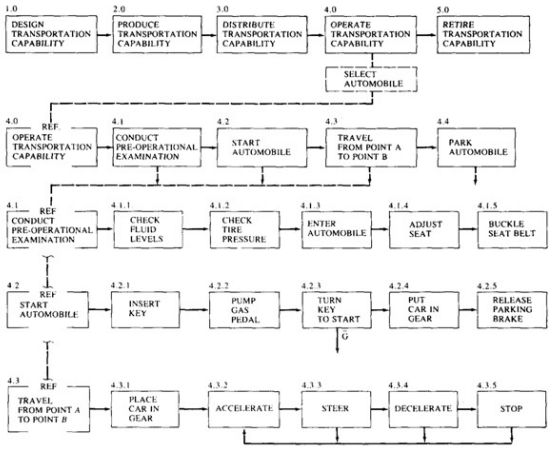

Figure 7: An example of a Functional Flow Diagram for an automobile.

4.2 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 8.

4.3 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 9. Function A will only be done correctly, if the functions B, C, D and E are all performed correctly.

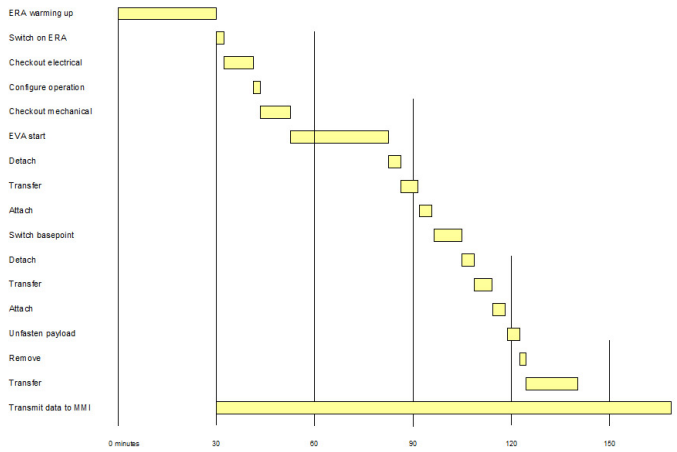

Figure 8: An example of a Time Line Analysis for the European Robotic Arm.

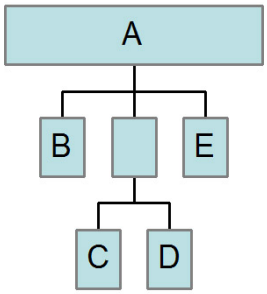

Figure 9: General form of the Functional Breakdown Structure.

4.4 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 10.

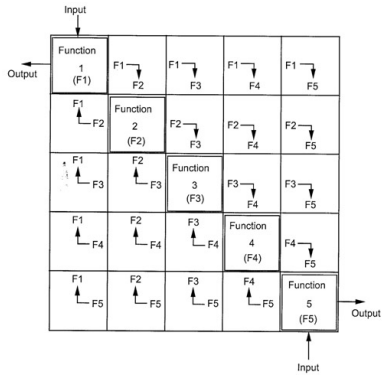

Figure 10: General form of the N^2 diagram.

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.

5 Requirements

5.1 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 taht 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

5.2 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 11.

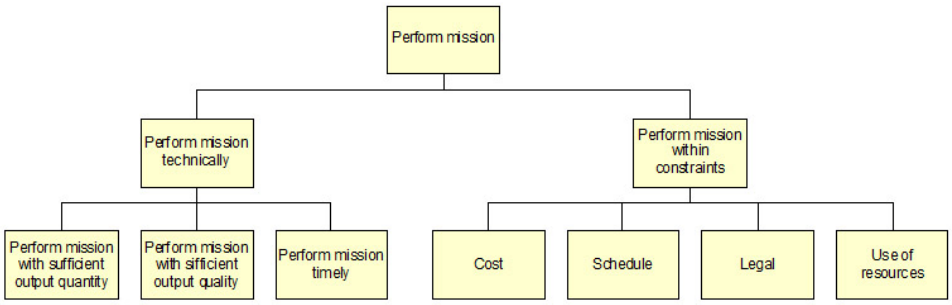

Figure 11: A general example of a Requirements Discovery Tree.

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.

5.3 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.

5.4 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 12. 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 a customer. This goes on a scale from 1 to 5. (We 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.)

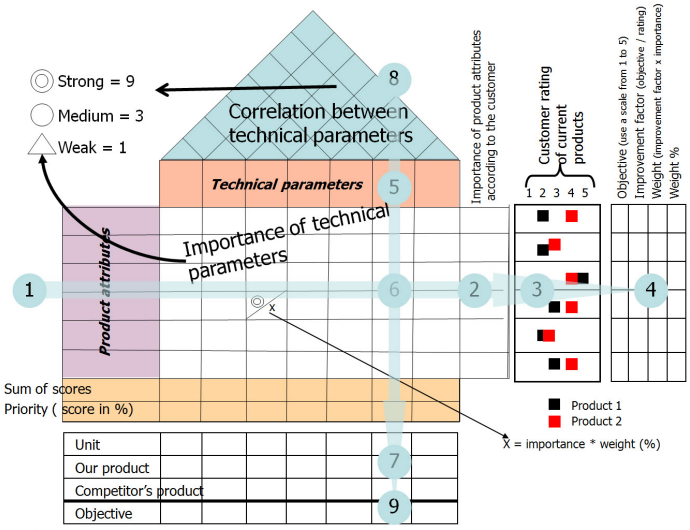

Figure 12: The general shape of the Quality Function Deployment diagram.

3. We will be comparing our desk design to other desks. To do this, we need to know what our customer thinks of other desks. So we ask our customer what he thinks of other desks. 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 the desk. 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. (Example 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 the looks only a bit (3 or 1). Also, the quality of the paint effects the looks quite a bit (9 or 3), but 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 hard 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 values to the technical parameters. (We can, for example, decide that we want wooden planks of exactly 1 centimeter thick.)

6 Resource Management

6.1 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 Burget** (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 it doesn'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**.

6.2 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'll still not be above the specificataion 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.

7 Risk Management

7.1 What is Risk Management?

When designing a system, unexpected things can always pop up. That's the risk of designing. In fact, the **risk** is defined as the measure of uncertainty, when attaining a goal. Risk is always present. So it needs to be dealt with. The process of managing risk is (surprisingly) called **Risk Management** (RM).

Risk Management has two main branches. First, there is **Project Risk Management** (PRM). This branch deals with technical risks: anything having to do with development. The second branch is **Environmental Risk Management**. This concerns the management of environmental health and safety risks.

7.2 Determining risks

When performing risk management, we should first identify the risks. But once that is done, how do we know which risks we should reduce?

To see that, we need to look at two things. We need to examine the **likelihood of occurrence** and the **consequence of the event**. Multiplying these to things gives us the risk. (For example, if a lightning strike has a 20% chance of occurring in a year, and would cost 10 thousand euros, then the risk in a year is 2 thousand euros.)

When risks are too high, two things can be done. Either the likelihood of occurrence can be decreased, or the consequence of the event can be decreased. (Of course we could also do both.)

7.3 The Risk Map

A way to visualize risks, is by using a Risk Map. One axis of the Risk Map denotes the likelihood of occurrence. The other axis indicates the consequence of the event. An empty risk map can be seen in figure 13.

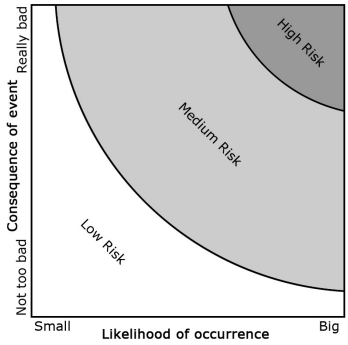

Figure 13: An empty risk map.

An event is respresented by a point on the risk map. The seriousness (the risk) of the event can then be read from the diagram. The darker the region it is in, the more serious the risk is.

8 Designing

8.1 The Design Option Tree

Let's suppose we're designing some system. How do we do that? Well, the first step is to start generating options. In what ways can we design our system?

A very handy tool for this, is the **Design Option Tree** (DOT). In such a tree, we start with a certain task our system needs to be able to perform. We then look at the ways in which this task can be performed. Every time, we split things up even more. It is important to note that the DOT is an 'or-tree'. Only one of the available options can be selected. An example of a DOT for registering images without delay can be seen in figure 14.

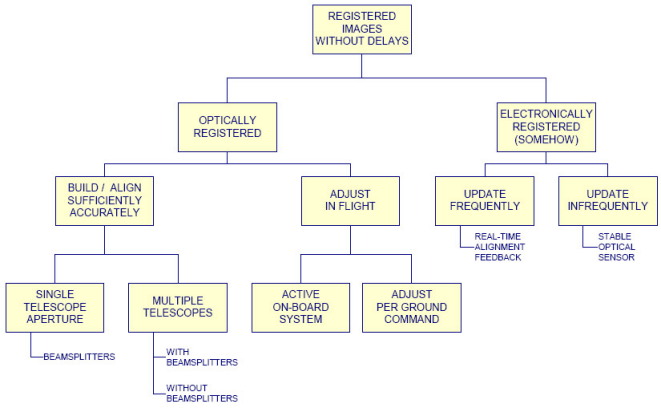

Figure 14: An example of a Design Option Tree for registering images without delay.

8.2 Choosing between designs

The DOT gives us a set of options. The next step is to choose the best option. This is a difficult process. There are therefore many ways to approach it.

A first thing we could do is eliminate concepts that are obviously not feasible. In this way, we can reduce the number of design options by quite a bit already.

The next step is to choose a **decision method**. We can distinguish two types of decision methods. In **ordinal (qualitative) methods**, the options are only ranked from best to worst, for every criterion. No individual score is given to them. So this is relatively easy. On the other hand, in **cardinal (quantative) methods**, the options are given a score. This requires a bit more work.

8.3 Ordinal methods

Let's suppose we have a set of design options. Also suppose that, for every design criterion, we have a ranking of options, from best to worse. How do we now select the best option? There are, again, multiple methods for this.

In the **Majority Rule**, we compare two alternatives. If, for a certain design option, alternative A is better than alternative B, then alternative A gets a point. Otherwise alternative B gets a point. We continue to do this for every design option. Eventually, the alternative with the most points wins.

The **Copeland Rule** is similar. However, we now compare an alternative with all other alternatives. Eventually, the alternative with the most points wins.

The **Rank-Sum Rule** is a more simple method. We simply add the ranking of all the criterions. So if an alternative is second for criterion α and third for criterion β, then its score is 5. The alternative with the lowest score now wins.

Finally, in the **Lexicographical Rule**, we first rank the criterions on their importance. We then only look at the most important criterion. The alternative that is best for this criterion wins. (In case of a tie, we look at the second most important criterion, and so on.)

8.4 Cardinal methods

To apply Cardinal methods, we first need to rate our alternatives. This must be done for every criterion. (So that will give us a lot of numbers.) Once this is done, how do we select the best alternative?

	Criterion 1		*Criterion 2*		……		*Criterion n*		*Maximum score*
Weighting factors	10		7		…….		4		yyyy
	Raw score	Wtd score	Raw score	Wtd score	Raw score	Wtd score	Raw score	Wtd score	Solution score
Alternative solution 1	5	50	8	56			6	24	xxxx
Alternative solution 2	…..	…..	…..	…..	…..	…..	…..	…..	…..
……	…..	…..	…..	…..	…..	…..	…..	…..	…..
Alternative solution m	…..	…..	…..	…..	…..	…..	…..	…..	…..

Figure 15: The general form of a Weighted Objectives Method table.

The most-often used cardinal method is the **Weighted Objectives Method** (WOM). To use this method, we must also give importance rankings for the criterions. Once that is done, we should find the score for the alternatives. This is done by multiplying the criterion importance by the rating score. The individual criterion scores are then added up. When doing this, a WOM table, as shown in figure 15, is often helpful. Eventually, the criterion with the best score wins.

9 Design verification

9.1 Verification methods

So now we've actually gotten a design for our system. This design should of course match the requirements. But we do have to be sure of this. The process of making sure the design meets the requirements is called **verification**.

There are several ways of verification (so-called **verification methods**). Examples are…

- **Review of design:** Inspecting the design documentation, to make sure that the product satisfies the requirements.
- **Inspection of the product:** By inspecting the product, ensuring that it meets the requirements.
- **Analysis:** Using (mathematical) analysis techniques to check the properties of the product. (The FEM method is often used.)
- **Testing:** Subjecting the product to actual tests, for example, by using a prototype.

9.2 The verification level

Next to the verification method, also the **verification level** matters. Verification is usually performed on a level as low as possible. In **low level tests**, as few elements as possible are involved. This means that tests can be done early, and aren't very complicated. (For example, you often don't need to consider the satellite structure if you want to test its on-board computer.)

However, sometimes low-level tests are not possible. This is when elements start influencing each other. In this case **high level tests** may be necessary. (For example, when doing an automobile crash test, a nearly complete car is used. This is partially because every part (having weight) influences the test.)

9.3 Test types

There are incredibly many ways to actually test a product. To list a few...

- **Functional tests:** Testing whether the product does the right things, and thus has the correct functionality.
- **Integration tests:** Testing whether all parts (mechanical/electrical/software) fit together in the correct way.
- **Structural tests:** Check whether the structure meets the demands. By doing this, also the analysis tools (like FEM) and the workmanship are tested.
- **Aerodynamic tests:** Check whether the aerodynamics of the product are sufficient. It also checks whether the mathematical model used during design was accurate. (Especially important for aircraft applications.)
- **Thermal tests:** Check whether the thermal system works in a sufficient way. (Especially important for space applications.)

9.4 Aircraft design verification

When designing an aircraft, there are a lot of regulations it should comply with. Checking whether the aircraft design actually meets these regulations is called **compliance finding**.

The process of compliance finding can be quite elaborate. Several tests need to be done to show compliance with the airworthiness requirements. When this is done, the authorities can issue a **type certificate**. This certificate shows that the type of aircraft could be airworthy.

10 Design for production

10.1 What is design for production?

Let's suppose we're designing an aircraft. This aircraft eventually needs to enter production. With **production**, we mean all activities needed to turn a design into an actual product. During the design, we could make sure that production becomes relatively easy. By doing this, we **design for production**.

10.2 Recurring and non-recurring processes

A lot of processes are necessary to produce an aircraft. We can make a rather import distinction between recurring and non-recurring processes. **Non-recurring processes** are processes that are executed only once for every product type. (Examples include creating the design, programming machines and acquiring tools.) On the other hand, **recurring processes** are processes repeated for every product of a specific type. (Examples are manufacturing of parts, assembly of parts, and testing of the product.)

10.3 Designing the production process

To produce an aircraft, we need a production process. This production process of course also needs to be designed. The designing a production process consists of five phases. These are...

- **Concept generation:** Creating the general concept of the production process.
- **Feasibility check:** Checking whether the general concept is feasible/realistic.
- **Process definition:** Actually define the production process in detail.

- **Full scale development:** Set up the actual production process, according to the process definition.
- **Process validation:** Check whether the production process actually meets the requirements and creates the right products.

When designing the production process, there are several things we need to pay attention to. The five most important points are...

- **Quality:** Is the quality sufficient? (Remember, we don't have to have the best quality for our product. A sufficient quality is good enough, and costs less.)
- **Time:** Is everything produced in time?
- **Money:** Is production as cheap as possible? (Is the amount of waste material minimized? Is the amount of elapse time minimized? Are transportation costs minimized? Etcetera.)
- **Volume:** Is the current production volume adequate to fulfill the demand? (Not too small, but also not too big?)
- **Law:** Does the production process obey safety, health, environmental and airworthiness regulations?

When all these points have received sufficient attention, then the production process is well designed.

10.4 Lean manufacturing

There is a philosophy in the world of production called **lean manufacturing**. According to this philosophy, production should be customer-focused, knowledge-driven, eliminating waste, creating value, dynamic and continuous. This is generally reached by eliminating waste. According to lean manufacturing, **waste** is anything that uses resources, but does not add real value to the product or service. There are eight important kinds of waste, being

1. Overproduction
2. Waiting time
3. Work in progress (WIP) or inventory
4. Processing waste
5. Transportation
6. Movement or motion
7. Rework
8. Underutilizing people

11 Reliability, maintainability and availability

11.1 The reliability function

In this part, we will examine reliability. The **reliability** R is the probability that a system will perform in a satisfactory manner, for a given period of time. The reliability of a system usually depends on time. It is therefore described by the **reliability function** $R(t)$.

From the reliability function, we can derive several other functions. The **failure distribution** $F(t)$ (also known as the **unreliability function**) is given by $F(t) = 1 - R(t)$. This failure distribution is then, in turn, related to the **failure density function** $f(t)$, according to

$$F(t) = \int_0^t f(\tau) d\tau. \tag{11.1}$$

Finally, the **hazard function** $r(t)$ (also known as the **failure rate**) is defined as

$$r(t) = \frac{\text{Failure Density}}{\text{Reliability}} = \frac{f(t)}{R(t)}. \tag{11.2}$$

11.2 A probability refreshment

We saw that reliability is defined as a probability. So let's quickly refresh our knownledge of the world of probabilities. We define $P(A)$ as the probability that event A occurs. We also define $P(A|B)$ as the probability that event A occurs, given that event B occurs. There are several rules and definitions concerning this probability.

Two events A and B are **independent** if $P(B)$ does not depend on whether A occurs or not. (In an equation, $P(B|A) = P(B)$.)

The **conditional theorem** states that $P(A \cap B) = P(A)P(B|A) = P(B)P(A|B)$. For two independent events A and B, we can simplify this to $P(A \cap B) = P(A)P(B)$. This relation is also known as the **multiplication theorem**.

Two events A and B are **mutually exclusive** if they can not occur at the same time. So if A occurs, then B does not occur. (In an equation, $P(A \cap B) = 0$.)

The **addition theorem** states that $P(A \cup B) = P(A) + P(B) - P(A \cap B)$. If A and B are mutually exclusive, this can be simplified to $P(A \cup B) = P(A) + P(B)$. So to find the probability that either of two mutually exclusive events A and B occurs, we can simply add up their probabilities $P(A)$ and $P(B)$.

Finally there is Bayes' rule, stating that

$$P(A|B) = \frac{P(B|A)P(A)}{\sum_{i=1}^{n} P(B|A_i)P(A_i)}. \tag{11.3}$$

11.3 Failure types and distributions

Reliability is all about predicting failures. This is difficult, because there are many types of failures. (Think of fatigue, structural overload, electrical overload, wear, etcetera.) And every failure mode often has its own type of failure distribution.

There are many types of failure distributions. The type of failure distribution mainly depends on the failure rate. If there is, for example, a constant failure rate $r(t) = \lambda$, then we will get a **negative exponential distribution**. This distribution is described by

$$f(t) = \frac{1}{\theta} e^{-t/\theta}, \tag{11.4}$$

where $\theta = 1/\lambda$ is the **Mean Time Between Failures** (MTBF). From this, we can derive the reliability function. It is given by

$$R(t) = e^{-t/\theta} = e^{-\lambda t}. \tag{11.5}$$

Negative exponential distributions often occur when considering failures caused by a random event, like bad weather, bird strikes, etcetera.

Of course not all failure distributions have a constant failure rate. Many types of failures have an increasing failure rate as time progresses. (Older products are more likely to fail.) An increasing failure rate can be described by a **normal distribution**. The average of the normal distribution then indicates the average life time of the product.

Some failure types have a decreasing failure rate. (An example is the failure of humans. In the first few years, humans are relatively likely to fail. If they live through their first years, they'll quite likely survive the next 50 years too.) Such failure rates can be described by a **Weibull distribution** or a **gamma distribution**.

11.4 Reliability of systems

Let's suppose we have a system consisting of n components. Also suppose that the system is build up, such that, if one component fails, then the whole system fails. (The components are said to be in **series**.) The reliability of the system can now be found, by multiplying all individual reliabilities. In an equation,

$$R_{total}(t) = \prod_{i=1}^{n} R_i(t). \tag{11.6}$$

We can also build up the system a bit differently. This time all components have to fail, for the system to fail. (The components are said to be in **parallel**). We now have

$$R_{total}(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)). \tag{11.7}$$

11.5 Fault Tree Analysis

As stated before, there are many ways in which a system can fail. To investigate those ways, a **Fault Tree** can come in handy. A fault tree describes what events are necessary to cause a certain type of failure. An example of a Fault Tree is shown in figure 16.

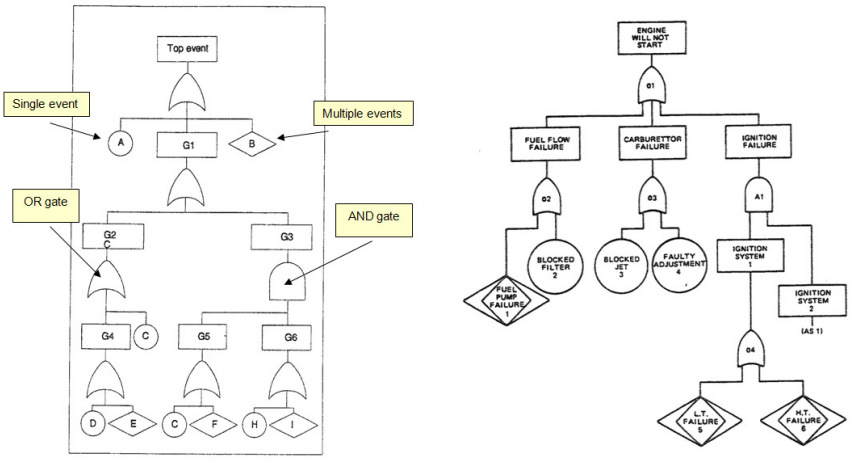

Figure 16: The general Fault Tree form (left) and an example applied to engine failure (right).

When examining failures, it is important to know the minimum cut sets. A **minimum cut set** is defined as a minimum set of events leading to failure. For example, for the Fault Tree in figure 16, events F and H would cause failure. The set (F, H) is thus a minimum cut set. The set (E, C) would also cause failure. This set is, however, not a minimum cut set. (Note the word 'minimum'.) This is because event C is not necessary to cause failure. (Event E by itself already ensures failure.)

11.6 Maintainability

The **maintainability** M of a system is defined as the ease, accuracy, safety, and economy in the performance of maintenance actions. There are two important kinds of maintenance. There **preventive maintenance**, aimed to prevent failure. (Inspection and periodic replacement of parts are part of this.)

There is also **corrective maintenance**. The goal of this is to patch up the system after a failure has occurred. (Repairs are thus part of corrective maintenance.)

There are several parameters with which the maintainability can be expressed. The most important ones are...

- **Mean Time To Repair** (MTTR): Average time needed to repair/restore the system.
- **Mean Preventive Maintenance Time** (MPMT): Average time needed to perform preventive maintenance.
- **Mean Time To Maintain** (MTTM): Average time needed to perform both preventive and corrective maintenance.
- **Mean Down Time** (MDT): The MTTM plus delays (for example, due to logistic reasons).

11.7 Availability

The **availability** A is the probability that a system will be available when required for use. If we only consider the effects of failures and ignore maintenance, then we are examining the **inherent availability** A_i. It is given by

$$A_i = \frac{MTTF}{MTTF + MTTR}. \tag{11.8}$$

If we, however, don't consider failure, but only maintenance, then we are dealing with the **achieved availability** A_a. This is given by

$$A_a = \frac{MTBM}{MTBM + MTTM}. \tag{11.9}$$

12 Life Cycle Costs

12.1 Cost types

A system of course has costs. All the costs that are made in the life of a system/product are called the **Life Cycle Costs** (LCC).

The LCC can be split up into two important categories: Recurring and non-recurring costs. **Non-recurring costs** are costs that only occur once for every product type. (Examples include design costs, machines/tool costs and facility costs.) **Recurring costs** are costs that are present for every product of a specific type. (Examples are material costs, labour costs and energy costs.)

12.2 The Cost Breakdown Structure

There can be a lot of different types of costs, in the life of a system. One way to display these costs, is by using a **Cost Breakdown Structure** (CBS). The CBS is a tree, in which the Life Cycle Costs are split up into categories. An example of a CBS for an aircraft can be seen in figure 17.

After the CBS has been created, numbers often need to be attached to categories. For this, **Cost Estimating Relations** (CER's) are often used. These relations are usually derived from experiences from earlier projects.

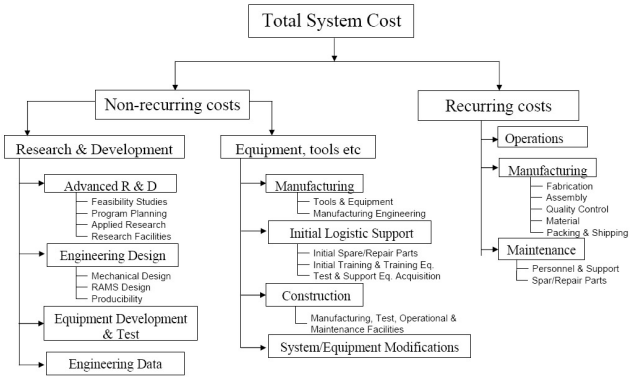

Figure 17: The Cost Breakdown Structure for an aircraft.