Abstract

Requirements engineering is one of the systems engineering pillars. In this document we discuss the fundamentals of systems engineering, such as the transformation of needs into specification. Needs and requirements prescribe what rather than how.
## Definition of “Requirement”

<table>
<thead>
<tr>
<th>Requirements describing the needs of the customer:</th>
<th><strong>Customer Needs</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements describing the characteristics of the final resulting system (product):</td>
<td><strong>System (Product) Specification</strong></td>
</tr>
<tr>
<td>The <em>requirements management process</em> recursively applies this definition for every level of decomposition.</td>
<td></td>
</tr>
<tr>
<td>Requirements describing the needs of the company itself over the life cycle:</td>
<td><strong>Life Cycle Needs</strong></td>
</tr>
</tbody>
</table>
Flow of Requirements

What

customer needs:
What is needed by the customer?

product specification:
What are we going to realize?

system design:
How are we going to realize the product?

What

What are the subsystems we will realize?

How

How will the subsystems be realized?

choices
trade-offs
negotiations

up to "atomic" components
System as a Black Box

- interfaces
- system seen as black box
- inputs
- functions
- quantified characteristics
- outputs
- restrictions, prerequisites
- boundaries, exceptions
- standards, regulations
Good Requirements are “SMART”

• **Specific** quantified

• **Measurable** verifiable

• **Achievable** (Attainable, Action oriented, Acceptable, Agreed-upon, Accountable)

• **Realistic** (Relevant, Result-Oriented)

• **Time-bounded** (Timely, Tangible, Traceable)
Specific Requirements have Specific Circumstances

**Typical Use Case**
- What is the user typically doing with the system in the system context
- Quantify the operation and context in this typical case

**Other Use Cases**
- Operational variants
- Boundary behavior
- Exceptional cases

SysML use cases

these use cases >> SysML use cases