Sentence types

**Ubiquitous**
- The <system name> shall <system response>
  - The kitchen system shall have an input hatch.

**Event-driven**
- When <optional preconditions> <trigger>, the <system> shall <system response>
  - When the chef inserts a potato to the input hatch, the kitchen system shall peel the potato.

**State-driven**
- While <in a state>, the <system> shall <system response>
  - While the kitchen system is in maintenance mode, the kitchen system shall reject all input.

**Unwanted behavior**
- If <optional preconditions> <trigger>, then the <system> shall <system response>
  - If a spoon is inserted to the input hatch, then the kitchen system shall eject the spoon.

**Optional**
- Where <feature>, the <system> shall <system response>
  - Where the kitchen system has a food freshness sensor, the kitchen system shall detect rotten foodstuffs.

Steps to take in applying EARS

Identify whether you are working with a requirement, or something else (e.g. note or example) → Identify compound requirements, i.e. whether the requirement needs to be split → Identify the acting system, person or process → Analyse the needed sentence type(s) → Identify possible missing requirements (e.g. 2 states and 3 events usually produce 4 requirements) → Review requirements if possible → Analyse the translated requirements for ambiguity, conflict and repetition → Identify possible missing requirements → Iterate as required

Some characteristics of a good requirement

- **Unambiguous**: One interpretation
- **Traceable**: Has unique identifier
- **Consistent**: Does not conflict other requirements
- **Verifiable**: Possible to check system meets requirement
- **Complete**: Not lacking relevant information
## EARS: Using combined sentences

**Example:** Optional feature combined with state-driven and event-driven

- Where the car has an ABS system, while the car is moving, when the driver applies brake, the ABS system shall detect blocked wheels.
- When the ABS system detects a blocked wheel, the ABS system shall reduce effective brake pressure for that wheel until the wheel is unblocked.

## Troubleshooting EARS problems

<table>
<thead>
<tr>
<th>Issue</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>No sentence type fits!</td>
<td>- Are you translating a requirement?</td>
</tr>
</tbody>
</table>
| I can't identify the actor! | - Use a higher abstraction level until it makes sense  
- Or get more information from relevant stakeholder |
| There's no system response! | - Usually the case with nonfunctional requirements  
- Can be expressed as “the system shall be ...” |
| There's no template for "shall not"! | - Feature of EARS, try stating as “shall be immune” or similar workaround  
- As last resort just use “shall not” structure |
| EARS produces too many atomic requirements! | - Deep technical requirements aren’t well suited to EARS  
- If necessary, use a list as accompaniment  
- Consider other format for technical requirements if EARS seems inappropriate |

## Beyond EARS: Other good practices

**Use a template that:**

- Provides for necessary metadata, e.g. requirement identifier  
- Has provision for non-requirements, e.g. notes and examples  
- But don’t be dragged down by too heavy templates

**Remember to keep your requirements up to date**

**Remember characteristics of good requirements**

**Requirements are about communicating between stakeholders**

- Ensure you can see the forest from the trees  
- Methods aren’t the meaning, they are a means to an end