Functional Architectures with SysML

Jesko Lamm
Senior Systems Engineer
jla@bernafon.ch

Tim Weilkiens
Managing Director
tim.weilkiens@oose.de
We believe in a world, in which people with restricted hearing can communicate again without limitations thanks to advanced technology.

Jesko Lamm works at Bernafon, a Swiss manufacturer of hearing instruments. As a Senior Systems Engineer, he is responsible for the processes in system architecture and for working as a system architect in development projects, based on model-based systems engineering with SysML.
We enable organizations to achieve their business goals themselves with innovative methods of Software and Systems Engineering.

Consulting and Training
Headquarter Hamburg, Germany

Tim Weilkiens, managing director of the German consultancy oose GmbH, is a member of INCOSE MBSE Challenge Team SE^2 (Telescope modeling). He is also an active member of the OMG working groups about SysML and UML and has written sections of the SysML specification.
Same functionality – evolving components: “Produce Sound”
Same functionality – evolving components: “Produce Sound”
Same functionality – evolving components: “Produce Sound”
Same functionality – evolving components: “Produce Sound”
Functional descriptions of a system can be re-used

Function “Produce Sound”
What is Functional Architecture?

- **Architecture**
  identifies the elements of a system and relates them to each other.

- **Functional Architecture**
  is based on functional elements whose input and output are related to each other via a function.

\[
    x \xrightarrow{y = f(x)} y \xrightarrow{z = g(y)} z
\]
**Information model**

- **Functional Architecture**
  - **Functional Element**
    - **Function**
      - **Sub-function**
    - **Sub-element**
  - **Connection**
    - **Exchange of Information**
      - **Flow of Force**
        - **Flow of Energy**
        - **Flow of Material**

© by oose innovative Informatik GmbH & Bernafon AG
Modeling Functional Architecture

Functional Block

Functional Architecture

- Provider
  - Requester
    - Functional Unit
      - Sub-element
      - Sub-function
    - Functional Group
      - Functional Unit
      - Sub-element
      - Sub-function

Connection

- Exchange of Information
- Flow of Force
- Flow of Energy
- Flow of Material

Representation in the model

Concept
Modeling Functional Architecture

- Functional Block
  - 0..*
  - Functional Unit
    - 0..*
    - Sub-element
      - 0..*
      - Sub-function
    - 1
  - Functional Unit
    - 1
  - Functional Element
    - 0..*
    - Sub-element
      - 0..*
      - Sub-function
    - 1
    - Requester
      - 0..*
    - Provider
      - 0..*
  - Functional Architecture
    - 0..*
    - Requester
      - 0..*
    - Provider
      - 0..*
  - Connection
    - 0..*
  - Exchange of Information
    - Flow of Force
      - Flow of Energy
        - Flow of Material

Representation of a Function

© by oose innovative Informatik GmbH & Bernafon AG
Functional Architecture (without SysML)

Functional Block

Representation of a Function

Representation of a Sub-function

Simulink® (Version 7.6)

Arbitrary Response

Default Amplification

Gain Change

volume ChangeGain change

function change = ChangeGain(volume, factor)
change = ConvertDelta(volume, factor);

function factor = GainChange(change, factor)
factor = ChangeGain(change, factor);

MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc.
Functional Architecture (SysML representation)

Representation in the model

Concept

Example

Functional Block

0..* - Sub-function

Functional Element

Function

0..* - Sub-function

<<Functional Block>>

Change Gain

Sub-function1()

Sub-function2()
Method for Creating Functional Architectures

- Amplification
  - Listen to Amplified Signal
  - Adjust Volume
- Volume
  - Adjust Volume

Functionality:
- Apply Gain Change
- Apply basic amplification
- Gain Change
- Volume
- Amplify Sound

System context:
- Functional System View
- Functional Domains
- Functional Block - Amplify Sound
- Functional Block - Change Gain
- Functional Block - Adjust Volume
- Sound Environment
Functional Requirements vs. Use Cases

- Functional Requirement: Amplification
- Functional Requirement: Volume
- Continuous use case: Listen to Amplified Signal
- Use case: Adjust Volume

Hearing Instrument User
Create Activity Diagrams

![Activity Diagram]

1. **Adjust Volume**
   - **funcationalRequirement**: Volume
   - **refine**

2. **Hearing Instrument User**

3. **Act** (Activity) Adjust Volume:
   - **Operate Volume Control**:
     - Volume Setting
   - **Compute Gain Change**:
     - Volume Setting
     - Gain Change
   - **Apply Gain Change**:
     - Gain Change
     - Audio Signal
   - **Judge Volume**:
     - Audio Signal
Activity Trees* of Use Cases

* also called „Function Trees“
Heuristics for Grouping Activities
Heuristics to Derive Functional Blocks

- Use grouping criteria of existing groups
- Abstract and secondary use cases define a functional group
- One functional group takes the functions that are related to system actors
- Function calls imply cohesion
- Functions that share data can be grouped
Example: Functions that Share Data Can Be Grouped

The remaining functions yield a group “Adjust Volume”.

Functions around the Audio Signal yield a group “Amplify Sound”.

© by oose innovative Informatik GmbH & Bernafon AG
Example: Functional Architecture

: System

: Amplify Sound

: Apply basic amplification

: Apply Gain Change

: Adjust Volume

: Change Gain

: Sound Environment

Environmental Sound

feeds end-user's hearing

Volume

Gain Change

Gain Change

Gain Change

Gain Change

Volume
Mapping Functions to Components

<table>
<thead>
<tr>
<th>Functional Architecture</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjust Volume</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Amplify Sound</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bdd [Model] Sound System</th>
<th>Functional-to-Physical Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>«allocate»</td>
<td>«allocate»</td>
</tr>
<tr>
<td>«allocate»</td>
<td>«allocate»</td>
</tr>
<tr>
<td>«allocate»</td>
<td>«allocate»</td>
</tr>
<tr>
<td>«allocate»</td>
<td>«allocate»</td>
</tr>
</tbody>
</table>

- «Physical Block» Microphone
- «Physical Block» Input Stage
- «Physical Block» Signal Processor
- «Physical Block» Filter 1
- «Physical Block» Filter 2

© by oose innovative Informatik GmbH & Bernafon AG
Functional Architectures Live Longer Than Technologies

Innovative Informatik

© by oose innovative Informatik GmbH & Bernafon AG
Conclusion

Functional Architecture ...

- represents the purpose of the system.
- is independent from the technical solution.
- focuses the user.

- is stable.
- does not impose technical solutions.
- can be re-used across product families and product generations.

- reduces development effort.
- increases customer benefit.
- enables innovative solutions.
Outlook: Tool support

- Tool support can facilitate the creation of functional blocks
- A first study* with Artisan Studio has been done
- A MagicDraw plugin is under development

News: www.fas-method.org

* Korff, Lamm, Weilkiens, Talk at the German Systems Engineering Conference “TdSE”, Nov. 2011
  (http://www.oose.de/fileadmin/Dateien/Publikationen/2011_TdSE_AutoFAS_english_version2.0.pdf)
Functional Architectures with SysML

Jesko Lamm
Senior Systems Engineer
jla@bernafon.ch

Tim Weilkiens
Managing Director
tim.weilkiens@oose.de