A ToolEd Method to Define, Analyse, Design & Validate System, Software, Hardware Architectures

Supporting Efficient Collaboration in Engineering

Validating/Justifying solution against Operational Need Easing Impact Analysis

Compatible with most processes top-down bottom-up, iterative, legacy-based, mixed...

Operational Analysis
What the users of the system need to accomplish

Functional & Non Functional Need
What the system has to accomplish for the users

Logical Architecture
How the system will work to fulfill expectations

Physical Architecture
How the system will be developed and built

Shared & Capitalised Architecture Models

Shared & Capitalised Architecture Models

Operational Need Analysis
Functional / Non Functional Need Analysis
Logical Architecture Design
Physical Architecture Design
Contracts for Development & IVVQ

Top-down
Iterative, Incremental
Bottom-up, Legacy, Reuse

Operational Analysis
Functional / Non Functional Need Analysis
Logical Architecture Design
Physical Architecture Design
Contracts for Development & IVVQ

Done & Checked
Done & Checked
Done & Checked
Done & Checked
Done & Checked

Time
Customer Operational Need Analysis
What the users of the system need to accomplish

- Define operational capabilities
- Perform an operational need analysis

System/ SW/HW Need Analysis
What the system has to accomplish for the Users

- Perform a capability trade-off analysis
- Perform a functional and non-functional analysis
- Formalise and consolidate requirements

Logical Architecture Design
How the system will work so as to fulfil expectations

- Define architecture drivers and viewpoints
- Build candidate architectural breakdowns in components
- Select best compromise architecture

Physical Architecture Design
How the system will be developed & built

- Define architectural patterns
- Consider reuse of existing assets design a physical
- Design a physical reference architecture
- Validate and check it

Development Contracts
What is expected from each designer/sub-contractor

- Define a components IVVQ strategy
- Define & enforce a PBS and component integration contract
**CONCEPTS**

- Operational capabilities
- Actors, operational entities
- Actor activities
- Interactions between activities & actors
- Information used in activities & interactions
- Operational processes chaining activities
- Scenarios for dynamic behaviour

**DESCRIPTION MEANS**

- Dataflow: functions, op. activities interactions & exchanges
- Functional chains, operational processes through functions & op. activities

**SAME CONCEPTS, PLUS:**

- Components
- Component ports and interfaces
- Exchanges between components
- Function allocation to components
- Component interface justification by functional exchanges allocation

**DATA MODEL:**

- Breakdown of functions & components
- Data model: dataflow & scenario contents, definition & justification of interfaces
- Allocation of op. activities to actors, of functions to components, of behav.components to impl.components, of dataflows to interfaces, of elements to configuration items

**OTHER CONCEPTS:**

- Configuration items tree
- Parts numbers, quantities
- Development contract (expected behaviour, interfaces, scenarios, resource consumption, non-functional properties...)

**MODES & STATES:**

- Modes & states of actors, system, components

**SAME CONCEPTS, PLUS:**

- Behavioural components refining logical ones, and implementing functional behaviour
- Implementation components supplying resources for behavioural components
- Physical links between implementation components
Verifying & checking solution against Non-functional & Industrial Stakes

<table>
<thead>
<tr>
<th>Method layers</th>
<th>Performance specific data sample</th>
<th>Safety specific data sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPERATIONAL NEED ANALYSIS</td>
<td>Max reaction time to threat</td>
<td>Feared events</td>
</tr>
<tr>
<td>FUNCTIONAL/NON FUNCTIONAL NEED ANALYSIS</td>
<td>Functional chain (FC) to react to threat. Maximum allowed latency on FC</td>
<td>Critical functional chains associated to events</td>
</tr>
<tr>
<td>LOGICAL ARCHITECTURE DESIGN</td>
<td>Processing &amp; exchanges complexity</td>
<td>Redundancy paths securing functional chains</td>
</tr>
<tr>
<td>PHYSICAL ARCHITECTURE DESIGN</td>
<td>Resource consumption on FC Resulting computing latency</td>
<td>Common failure modes Fault propagation on FC</td>
</tr>
<tr>
<td>CONTRACTS FOR DEVELOPMENT &amp; IVVQ</td>
<td>Allocated resources to satisfy latency</td>
<td>Needed reliability level</td>
</tr>
</tbody>
</table>

- **Cost & Schedule**
- **Interfaces**
- **Performance**
- **Maintainability**
- **Safety/security**
- **IVVQ**
- **Product Policy**

**Operational Performance**
- Latency: New attribute of functional chain (expected + computed)
- < 2 seconds

**Safety**
- Redundancy: Extension of functional chain
- Voter: Extension of component

**Technical Performance**
- Load balanced cluster: New concept

**Recovering**
- Functions or components

**ARCHITECTURE CHECK**
- Compute function traversal time according to implement component & communication perf
- Deduce achieved latency & compare to expected
- Compare redundancy level with functional chain criticality
- Propagate HW failures to functional chains
- Compute load based on functions & check adequacy
- Both on computing power and communication bandwidth

**MULTI-VIEWPOINT CHECK OF ARCHITECTURE**