Abstract
The purpose of the conceptual view is described. A number of methods or models is given to use in this view: construction decomposition, functional decomposition, class or object decomposition, other decompositions (power, resources, recycling, maintenance, project management, cost, ...), and related models (performance, behavior, cost, ...); allocation, dependency structure; identify the infrastructure (factoring out shareable implementations), classify the technology in core, key and base technology; integrating concepts (start up, shutdown, safety, exception handling, persistency, resource management,...).
Example construction decomposition simple TV

The conceptual view

version: 0.7
March 6, 2013
CVconstructionDecomposition

The conceptual view

version: 0.7
March 6, 2013
CVconstructionDecomposition
## Characterization of the construction decomposition

### Management of Design

<table>
<thead>
<tr>
<th>Unit of Creation</th>
<th>SW Example</th>
<th>HW Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>PCB</td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>IP cells</td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td>IP core</td>
<td></td>
</tr>
</tbody>
</table>

### Unit of Aggregation for Organisation

<table>
<thead>
<tr>
<th>SW Example</th>
<th>HW Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Box</td>
</tr>
<tr>
<td>Module</td>
<td>IP core</td>
</tr>
<tr>
<td></td>
<td>IC</td>
</tr>
</tbody>
</table>
Example functional decomposition camera type device
Characterization of the functional decomposition

How;
what is the flow of internal activities
to realise external functionality?

some keywords:
activities
transformation
input output
data flow
control flow

multiple functional decompositions are possible and valuable!
How about the \textit{characteristic} of the \textit{component} when performing \textit{function}?
Selection factors to improve the question generator

Critical for system performance

Risk planning wise

Least robust part of the design

Suspect part of the design
  - experience based
  - person based
Addressing planes or lines

The conceptual view

version: 0.7
March 6, 2013
CVquestionGeneratorPlanes
Example partial internal information model

The conceptual view

version: 0.7
March 6, 2013
CVinformationModel
Example process decomposition

The conceptual view
Execution architecture

- **dead lines**
- **timming, throughput requirements**

**functional model**
- **input**
- **demux**
- **process**
- **display**
- **store**

**hardware**
- **CPU**
- **DSP**
- **RAM**
  - **tuner**
  - **drive**

**repository structure**
- **Applications**
- **UI toolkit**
- **processing**
- **foundation classes**
- **hardware abstraction**

**execution architecture**
- **process**
- **task**
- **thread**

**interrupt handlers**

**other architecture views**

**issues:**
- concurrency
- scheduling
- synchronisation
- mutual exclusion
- priorities
- granularity

The conceptual view

version: 0.7
March 6, 2013
CVexecutionArchitecture
Performance Model

\[ t_{\text{recon}} = t_{\text{filter}}(n_{\text{raw-x}}, n_{\text{raw-y}}) + n_{\text{raw-x}} \cdot (t_{\text{fft}}(n_{\text{raw-y}})) + n_{y} \cdot (t_{\text{fft}}(n_{\text{raw-x}})) + t_{\text{corrections}}(n_{x}, n_{y}) + t_{\text{control-overhead}} + t_{\text{col-overhead}} + t_{\text{row-overhead}} \]

\[ t_{\text{fft}}(n) = c_{\text{fft}} \cdot n \cdot \log(n) \]
Safety, Reliability and Security concepts

- containment (limit failure consequences to well defined scope)
- graceful degradation (system parts not affected by failure continue operation)
- dead man switch (human activity required for operation)
- interlock (operation only if hardware conditions are fulfilled)
- detection and tracing of failures
- black box (log) for post mortem analysis
- redundancy
Simplified start up sequence

1. **Power**
   - load
   - configure
   - initialise, start

2. **Boot-loader**
   - bring in initial state
   - load and initialise firmware

3. **Firmware**
   - determine next layer
   - load and initialise loader
   - determine loading HW

4. **Kernel**
   - discover kernel HW
   - initialise kernel data structures
   - determine next layer
   - load, initialise and start loader

5. **Loader**
   - determine loading HW

6. **Services**
   - configure services
   - allocate resources
   - load, initialise and start services
   - discover kernel HW
   - initialise kernel data structures
   - determine next layer

7. **User Interface**
   - configure UI
   - allocate resources
   - load, initialise and start UI

8. **Application**
   - load
   - configure
   - initialise, start

9. **Connect to Outside**
   - detect external services
   - publish internal services
   - connect where needed

The conceptual view

**Start up**
- load, initialise and start services
- allocate resources
- configure services
- load, initialise and start UI
- discover kernel HW
- initialise kernel data structures
- determine next layer
- load and initialise loader
- determine loading HW
- determine next layer
- bring in initial state
- load and initialise firmware

**Stop**
- flush ongoing activities
- close connections
- save persistent data
- free resources
- stop

**Shut down**
- stop in safe sequence
Example work breakdown

The conceptual view

15 Gerrit Muller

version: 0.7
March 6, 2013
CVworkBreakdown
Core, Key or Base technology

- Core
- Key
- Base

Technology life cycle:

- Own value IP
- Critical for final performance
- Commodity

Options:
- make
- outsource
- buy
- refer customer to 3rd party

Partnering
Example integration plan

The conceptual view

version: 0.7
March 6, 2013
CVIntegrationPlan