Abstract
This module addresses the Conceptual and Realization Views.
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
July 31, 2014
CVconstructionDecomposition
Characterization of the construction decomposition

<table>
<thead>
<tr>
<th>Management of design</th>
<th>SW example</th>
<th>HW example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of creation</td>
<td>file</td>
<td>PCB</td>
</tr>
<tr>
<td>Storage</td>
<td>IP cells</td>
<td>IP cells</td>
</tr>
<tr>
<td>Update</td>
<td>IP core</td>
<td>IP core</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit of aggregation</td>
<td>package</td>
<td>box</td>
</tr>
<tr>
<td>for organisation</td>
<td>module</td>
<td>IP core</td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td>IC</td>
</tr>
<tr>
<td>Release</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example functional decomposition camera type device

The conceptual view
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 <characteristic> of the <component> when performing <function>?

What is the memory usage of the user interface when querying the DB?
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
July 31, 2014
CVquestionGeneratorPlanes
Example partial internal information model

The conceptual view

version: 0.7
July 31, 2014
CVInformationModel
Example process decomposition

The conceptual view

version: 0.7
July 31, 2014
CVprocessDecomposition
Execution architecture

dead lines
timing, throughput
requirements

The conceptual view

12 Gerrit Muller
version: 0.7
July 31, 2014
CVexecutionArchitecture
\[ t_{\text{recon}} = t_{\text{filter}}(n_{\text{raw-x}}, n_{\text{raw-y}}) + n_{\text{raw-x}} \cdot (t_{\text{fft}}(n_{\text{raw-y}}) + n_{y} \cdot (t_{\text{fft}}(n_{\text{raw-x}}) + t_{\text{corrections}}(n_{x}, n_{y}) + t_{\text{control-overhead}}) + t_{\text{col-overhead}}) + t_{\text{row-overhead}}) + t_{\text{row-overhead}} \]

\[ t_{\text{fft}}(n) = c_{\text{fft}} \cdot n \cdot \log(n) \]
• 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

- **Power**
  - **Start up**
  - **Boot-loader**
    - Determine loading HW
    - Load and initialise loader
    - Determine next layer
  - **Kernel**
    - Discover kernel HW
    - Initialise kernel data structures
    - Determine next layer
  - **Services**
    - Configure services
    - Allocate resources
    - Load, initialise and start services
  - **User interface**
    - Configure UI
    - Allocate resources
    - Load, initialise and start UI
  - **Application**
    - Connect to outside
    - Detect external services
    - Publish internal services
    - Connect where needed

- **Stop in safe sequence**
  - Flush ongoing activities
  - Close connections
  - Save persistent data
  - Free resources
  - Stop

The conceptual view

15 Gerrit Muller

version: 0.7
July 31, 2014
CVstartUp
Example work breakdown

TIP:NBE
R1

scanning

- preparation
- xDAS
- reconstruction

viewing

- conversion
- xFEC
- hardware

database

- run time
- algorithms
- database engine
- clinical
- bulk data
- import
- export

computing

- acq
- conversion
- algorithms
- gfx
- UI
- archive
- integration
- start up shutdown
- exception handling

system

- alfa test
- beta test
- conf man
- SPS
- SDS
- TPS
- make SW
- make HW
- buy SW
- buy HW
- system

The conceptual view
16 Gerrit Muller

version: 0.7
July 31, 2014
CVworkBreakdown
Core, Key or Base technology

Technology life cycle

Technology life cycle

Own value IP

Critical for final performance

Commodity

make
outsource
buy
refer customer to 3rd party

Partnering

The conceptual view
17  Gerrit Muller
Example integration plan

2 partial systems for SW testing

2 existing base systems

new base systems

adopt existing base SW

new application

SW dev system
test and refine application

existing base system

integrate and refine application

SW for new HW subsystem
test SW for new HW subsystem

SW dev system
test HW subsystem

new HW subsystem

integrate subsystem

existing base system

integrate HW system

new base system

test new base system

integrate system

existing
ew

application integration

new subsystem integration

integrated system

time

The conceptual view
18 Gerrit Muller

version: 0.7
July 31, 2014
CVIntegrationPlan
The realization view

by Gerrit Muller       Buskerud University College

  e-mail: gaudisite@gmail.com
  www.gaudisite.nl

Abstract
The realization view looks at the actual technologies used and the actual implementation. Methods used here are logarithmic views, micro-benchmarks and budgets.

Analysis methods with respect to safety, reliability and security provide a link back to the functional and conceptual views.
Budget based design flow

can be more complex than additions

model

measurements existing system

micro benchmarks aggregated functions applications

spec

SRS
\[ t_{\text{boot}} = 0.5s \]
\[ t_{\text{zap}} = 0.2s \]

design estimates; simulations

V4aa

feedback

measurements new (proto) system

micro benchmarks aggregated functions applications profiles traces

budget

feedback

form

\[ t_{\text{proc}} + t_{\overline{\text{over}}} + t_{\text{disp}} \]

\[ t_{\text{proc}} \]
\[ t_{\overline{\text{over}}} \]
\[ t_{\text{disp}} \]

\[ T_{\text{proc}} \]
\[ T_{\overline{\text{over}}} \]
\[ T_{\text{disp}} \]
\[ T_{\text{total}} = 55 \]

The realization view

version: 0.1
July 31, 2014

Gerrit Muller
Example of a memory budget

<table>
<thead>
<tr>
<th>memory budget in Mbytes</th>
<th>code</th>
<th>obj data</th>
<th>bulk data</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared code</td>
<td>11.0</td>
<td></td>
<td></td>
<td>11.0</td>
</tr>
<tr>
<td>User Interface process</td>
<td>0.3</td>
<td>3.0</td>
<td>12.0</td>
<td>15.3</td>
</tr>
<tr>
<td>database server</td>
<td>0.3</td>
<td>3.2</td>
<td>3.0</td>
<td>6.5</td>
</tr>
<tr>
<td>print server</td>
<td>0.3</td>
<td>1.2</td>
<td>9.0</td>
<td>10.5</td>
</tr>
<tr>
<td>optical storage server</td>
<td>0.3</td>
<td>2.0</td>
<td>1.0</td>
<td>3.3</td>
</tr>
<tr>
<td>communication server</td>
<td>0.3</td>
<td>2.0</td>
<td>4.0</td>
<td>6.3</td>
</tr>
<tr>
<td>UNIX commands</td>
<td>0.3</td>
<td>0.2</td>
<td>0.0</td>
<td>0.5</td>
</tr>
<tr>
<td>compute server</td>
<td>0.3</td>
<td>0.5</td>
<td>6.0</td>
<td>6.8</td>
</tr>
<tr>
<td>system monitor</td>
<td>0.3</td>
<td>0.5</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>application SW total</td>
<td>13.4</td>
<td>12.6</td>
<td>35.0</td>
<td>61.0</td>
</tr>
<tr>
<td>UNIX Solaris 2.x</td>
<td></td>
<td></td>
<td></td>
<td>10.0</td>
</tr>
<tr>
<td>file cache</td>
<td></td>
<td></td>
<td></td>
<td>3.0</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td></td>
<td></td>
<td>74.0</td>
</tr>
</tbody>
</table>
Actual timing on logarithmic scale

- **application needs**
  - light travels 1 cm
  - (ps) $10^{-12}$
  - cycle 2 GHz CPU
- 100Hz video pixel time
  - (ns) $10^{-9}$
- zero message transfer
  - (s) $10^{-6}$
- appl level message exchange
  - (ms) $10^{-3}$
- appl level function response
  - (s) $1$
## Typical micro benchmarks for timing aspects

<table>
<thead>
<tr>
<th>Area</th>
<th>Infrequent operations, often time-intensive</th>
<th>Often repeated operations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>database</strong></td>
<td>start session, finish session</td>
<td>perform transaction, query</td>
</tr>
<tr>
<td><strong>network, I/O</strong></td>
<td>open connection, close connection</td>
<td>transfer data</td>
</tr>
<tr>
<td><strong>high level construction</strong></td>
<td>component creation, component destruction</td>
<td>method invocation, same scope, other context</td>
</tr>
<tr>
<td><strong>low level construction</strong></td>
<td>object creation, object destruction</td>
<td>method invocation</td>
</tr>
<tr>
<td><strong>basic programming</strong></td>
<td>memory allocation, memory free</td>
<td>function call, loop overhead, basic operations (add, mul, load, store)</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>task, thread creation</td>
<td>task switch, interrupt response</td>
</tr>
<tr>
<td><strong>HW</strong></td>
<td>power up, power down, boot</td>
<td>cache flush, low level data transfer</td>
</tr>
</tbody>
</table>
The transfer time as function of blocksize

<table>
<thead>
<tr>
<th>time</th>
<th>block size</th>
</tr>
</thead>
<tbody>
<tr>
<td>worst case</td>
<td>optimal block-size</td>
</tr>
</tbody>
</table>

$t_{overhead}$
Performance evaluation

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

\[ t_{\text{fft}}(n) = c_{\text{fft}} \cdot n \cdot \log(n) \]

The realization view

The focus is on overhead reduction, which is more important than faster algorithms. This is not an excuse for sloppy algorithms.
Performance Cost, input data

- 5400 rpm
- 7200 rpm
- 7200 rpm, 8 MB buffer
- pentium4

The realization view

Source: http://www.mpcomp.com/
September 5, 2002

GHz
GByte
performance
performance / cost
storage capacity

version: 0.1
July 31, 2014
RVperformanceCost
Performance Cost, choice based on sales value

The realization view

version: 0.1
July 31, 2014
RVperformanceCostChoice
Performance Cost, effort consequences

The realization view
28  Gerrit Muller

version: 0.1
July 31, 2014
RVperformanceCostEffort
But many many other considerations

The realization view

version: 0.1
July 31, 2014
RVcostPerformanceIssues
Safety, Reliability and Security analysis methods

<table>
<thead>
<tr>
<th>safety</th>
<th>potential hazards</th>
<th>probability</th>
<th>measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>hazard analysis</td>
<td>potential hazards</td>
<td>probability</td>
<td>measures</td>
</tr>
<tr>
<td>reliability</td>
<td>failure modes</td>
<td>effects</td>
<td>measures</td>
</tr>
<tr>
<td>FMEA</td>
<td>failure modes</td>
<td>effects</td>
<td>measures</td>
</tr>
<tr>
<td>security</td>
<td>vulnerability risks</td>
<td>consequences</td>
<td>measures</td>
</tr>
</tbody>
</table>
Make a first design:

- decomposition in functions

- decomposition in building blocks

- budgets for most important quality requirements
Exercise Design Side, second iteration

- Make a design:
  - that covers the most critical design aspects
  - that fulfills the most important and valuable customer needs

- Make a presentation of the design of maximal 8 sheets.