The conceptual view

by Gerrit Muller     University of South-Eastern Norway-NISE
         e-mail: gaudisite@gmail.com
                www.gaudisite.nl

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

- **Applications**: view, PIP, adjust, view TXT, viewport, menu
- **Services/Toolboxes**: browse, networking, file-system
- **Driver**: drivers, scheduler, OS, CPU, RAM, etc.
- **Hardware**: signal processing subsystem, control subsystem
- **Domain Specific** vs. **Generic**

The conceptual view

version: 0.7
September 6, 2020
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></td>
<td></td>
<td>IP cells</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IP core</td>
</tr>
<tr>
<td>storage</td>
<td></td>
<td></td>
</tr>
<tr>
<td>update</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>unit of aggregation for organisation</th>
<th>package module</th>
<th>box</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td></td>
<td>IP core</td>
</tr>
<tr>
<td>release</td>
<td></td>
<td>IC</td>
</tr>
</tbody>
</table>
Example functional decomposition

- **acquisition**
- **acquisition processing**
- **compress**
- **encoding**
- **storage**
- **display**
- **display processing**
- **de-compress**
- **decoding**
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 <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
September 6, 2020
CVquestionGeneratorPlanes

Gerrit Muller

8
Example partial internal information model

The conceptual view

version: 0.7
September 6, 2020
CVInformationModel
The conceptual view
10  Gerrit Muller

version: 0.7
September 6, 2020
CVprocessDecomposition
Execution architecture

dead lines
timing, throughput
requirements

functional
model
receive
display
store
process
demux

other architecture
views
Map

execute
architecture

process
task
thread
interrupt
handlers

execution architecture
issues:
concurrency
scheduling
synchronisation
mutual exclusion
priorities
granularity

hardware
CPU
DSP
RAM
tuner
drive

repository
structure
Applications
UI toolkit
processing
DCT
foundation
classes
queue
list
hardware
abstraction
tuner
DOD-drive

The conceptual view
version: 0.7
September 6, 2020
CVexecutionArchitecture
Performance Model

The conceptual view

12 Gerrit Muller

version: 0.7
September 6, 2020
CVreconstructionPerformanceModel

\[ t_{\text{recon}} = t_{\text{filter}}(n_{\text{raw-x}}, n_{\text{raw-y}}) + n_{\text{raw-x}} \cdot (t_{\text{fft}}(n_{\text{raw-y}}) + n_{y} \cdot (t_{\text{fft}}(n_{\text{raw-x}}) + t_{\text{corrections}}(n_{x}, n_{y}) + t_{\text{col-overhead}} + t_{\text{row-overhead}} + t_{\text{control-overhead}}) + t_{\text{fft}}(n) = c_{\text{fft}} \cdot n \cdot \log(n) \]
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. Discover kernel HW
2. Initialise kernel data structures
3. Determine next layer
4. Load and initialise loader
5. Determine loading HW
6. Determine next layer
7. Bring in initial state
8. Load and initialise firmware
9. Configure services
10. Allocate resources
11. Load, initialise and start services
12. Configure UI
13. Allocate resources
14. Load, initialise and start UI
15. Discover kernel HW
16. Initialise kernel data structures
17. Determine next layer
18. Load and initialise loader
19. Determine loading HW
20. Determine next layer
21. Bring in initial state
22. Load and initialise firmware
23. Configure services
24. Allocate resources
25. Load, initialise and start services
26. Configure UI
27. Allocate resources
28. Load, initialise and start UI
29. Discover kernel HW
30. Initialise kernel data structures
31. Determine next layer
32. Load and initialise loader
33. Determine loading HW
34. Determine next layer
35. Bring in initial state
36. Load and initialise firmware
37. Configure services
38. Allocate resources
39. Load, initialise and start services
40. Configure UI
41. Allocate resources
42. Load, initialise and start UI
43. Discover kernel HW
44. Initialise kernel data structures
45. Determine next layer
46. Load and initialise loader
47. Determine loading HW
48. Determine next layer
49. Bring in initial state
50. Load and initialise firmware
51. Configure services
52. Allocate resources
53. Load, initialise and start services
54. Configure UI
55. Allocate resources
56. Load, initialise and start UI
57. Discover kernel HW
58. Initialise kernel data structures
59. Determine next layer
60. Load and initialise loader
61. Determine loading HW
62. Determine next layer
63. Bring in initial state
64. Load and initialise firmware
65. Configure services
66. Allocate resources
67. Load, initialise and start services
68. Configure UI
69. Allocate resources
70. Load, initialise and start UI
71. Discover kernel HW
72. Initialise kernel data structures
73. Determine next layer
74. Load and initialise loader
75. Determine loading HW
76. Determine next layer
77. Bring in initial state
78. Load and initialise firmware
79. Configure services
80. Allocate resources
81. Load, initialise and start services
82. Configure UI
83. Allocate resources
84. Load, initialise and start UI
85. Discover kernel HW
86. Initialise kernel data structures
87. Determine next layer
88. Load and initialise loader
89. Determine loading HW
90. Determine next layer
91. Bring in initial state
92. Load and initialise firmware
93. Configure services
94. Allocate resources
95. Load, initialise and start services
96. Configure UI
97. Allocate resources
98. Load, initialise and start UI
99. Discover kernel HW
100. Initialise kernel data structures
101. Determine next layer
102. Load and initialise loader
103. Determine loading HW
104. Determine next layer
105. Bring in initial state
106. Load and initialise firmware
107. Configure services
108. Allocate resources
109. Load, initialise and start services
110. Configure UI
111. Allocate resources
112. Load, initialise and start UI
113. Discover kernel HW
114. Initialise kernel data structures
115. Determine next layer
116. Load and initialise loader
117. Determine loading HW
118. Determine next layer
119. Bring in initial state
120. Load and initialise firmware
121. Configure services
122. Allocate resources
123. Load, initialise and start services
124. Configure UI
125. Allocate resources
126. Load, initialise and start UI
127. Discover kernel HW
128. Initialise kernel data structures
129. Determine next layer
130. Load and initialise loader
131. Determine loading HW
132. Determine next layer
133. Bring in initial state
134. Load and initialise firmware
135. Configure services
136. Allocate resources
137. Load, initialise and start services
138. Configure UI
139. Allocate resources
140. Load, initialise and start UI
141. Discover kernel HW
142. Initialise kernel data structures
143. Determine next layer
144. Load and initialise loader
145. Determine loading HW
146. Determine next layer
147. Bring in initial state
148. Load and initialise firmware
149. Configure services
150. Allocate resources
151. Load, initialise and start services
Example work breakdown

TIP:NBE
R1

scanning

viewing

database

computing

system

project organization

work packages

scanning

xDAS

XFEC

reconstruction

hardware

VDU

console

algorithms

gfx

UI

database

clinical

bulk data

import
export

archive

database engine

foundation classes

start up
shutdown

exception handling

exception handling

host

OS

SPS

SDS

TPS

integration

alta test

beta test

conf man

project

segment

make SW

make HW

buy SW

buy HW

system
Core, Key or Base technology

- Core
- Key
- Base

Technology life cycle

- Own value IP
- Critical for final performance
- Commodity

Total Product

- Make
- Outsource
- Buy
- Refer customer to 3rd party

Partnering

The conceptual view

Gerrit Muller
version: 0.7
September 6, 2020
SSScoreKeyBase
Example integration plan

The conceptual view