Abstract
The creation of a Product Family is an alternation of decomposition and synthesis steps. The products and intermediate compositions can be viewed as recursive aggregation levels. Careful trade-offs are required between the size of an aggregation level and the way it will be deployed, to balance amongst others flexibility and (configuration) manageability.
Aggregation Levels viewpoints

- Source Code Management
- Composition
- Deployment

Integration and Test

Product Creation

Documentation
## Concerns per viewpoint

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>Requirements, Specification, Design, Transfer, Test, Support</td>
</tr>
<tr>
<td>Source Code Management</td>
<td>Storage, Management, Generation</td>
</tr>
<tr>
<td>Composition</td>
<td>System, Subsystem, Function, Application</td>
</tr>
<tr>
<td>Deployment</td>
<td>Releasing, Distribution, Protection, Update, Installation, Configuration</td>
</tr>
<tr>
<td>Integration and Test</td>
<td>Confidence, Problem Tracking</td>
</tr>
<tr>
<td>Viewpoint</td>
<td>Entities</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Documentation</td>
<td>Product Family, Product/System, Function/Feature, Subsystem, Component,</td>
</tr>
<tr>
<td></td>
<td>Building Block, Module</td>
</tr>
<tr>
<td>Source Code Management</td>
<td>Package, File</td>
</tr>
<tr>
<td>Composition</td>
<td>Product, Executable, Dynamic Library, Component</td>
</tr>
<tr>
<td>Deployment</td>
<td>Distribution Medium (&quot;CD&quot;), Unit of Licensing (&quot;SW key&quot;), Package,</td>
</tr>
<tr>
<td></td>
<td>Patch, Configuration data</td>
</tr>
<tr>
<td>Integration and Test</td>
<td>Test Configurations, Intermediate Integration results</td>
</tr>
</tbody>
</table>
Documentation Viewpoint

What is asked for
(Requirements)

What will be realized
(Specifications)

How
(Design)

Verify
report
(Test)

Transfer
to Product Creation
(Support) and Customer
Oriented Process
(Engineering)

Aggregation Levels in Composable Architectures

version: 2.4
March 6, 2013

Gerrit Muller
## Typical Sizes of SW for Aggregation Levels

<table>
<thead>
<tr>
<th>Entity</th>
<th>Typical size loc</th>
<th>packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>repository</td>
<td>1M-10M</td>
<td>10-100</td>
</tr>
<tr>
<td>package</td>
<td>10k–100k</td>
<td></td>
</tr>
<tr>
<td>file</td>
<td>100-1k</td>
<td></td>
</tr>
</tbody>
</table>
Rules of thumb file-size

- Files should be larger than 100 loc;
  The overhead per file and the "value" per file must be balanced.
- Files should be less than 1000 loc;
  Large files reduce the overview within the module. Larger files are an indication for a lack of modularity.
• at least 10 files per package;
  Packaging files or modules generates some overhead in usage and management. The value of this packaging must be substantial to offset this additional overhead.

• at most 100 kloc per package to maintain overview;
  For unambiguous package-ownership and sufficient overview.
Composition Viewpoint: Granularity

Small number of Large Components

Large number of Small Components
### Nr Components vs Nr of Architects; Naive

<table>
<thead>
<tr>
<th>Capacity of architects $c$</th>
<th>10</th>
<th>20</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of components $n$</strong></td>
<td><strong>Number of relations $r = n \sqrt{n}$</strong></td>
<td><strong>Number of Architects $a = r / c$</strong></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>32</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>89</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>40</td>
<td>253</td>
<td>25</td>
<td>13</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>300</td>
<td>5196</td>
<td>520</td>
<td>260</td>
</tr>
<tr>
<td>1000</td>
<td>31623</td>
<td>3162</td>
<td>1581</td>
</tr>
<tr>
<td>Number of components $n$</td>
<td>Number of relations $r = n\sqrt{n}$</td>
<td>weight $w$</td>
<td>Number of Architects $a = (r \times w)/c$</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------</td>
<td>------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>32</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>89</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>40</td>
<td>253</td>
<td>2</td>
<td>39</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>1</td>
<td>114</td>
</tr>
<tr>
<td>300</td>
<td>5196</td>
<td>1</td>
<td>534</td>
</tr>
<tr>
<td>1000</td>
<td>31623</td>
<td>1</td>
<td>3176</td>
</tr>
</tbody>
</table>
Field Deployment viewpoint

- granularity of sellable features and services
- lifecycle support
- internal logistics and production process
Integration and Test viewpoint

The graph illustrates the relationship between confidence level after integration, duration of integration, cost of bottom-up testing, arbitrary capacity scale, and arbitrary elapsed time scale.

- **Confidence Level after Integration**: Increases as the scale increases.
- **Duration of Integration**: Decreases as the scale increases.
- **Cost of Bottom-up Testing**: Generally increases as the scale increases.
- **Arbitrary Capacity Scale**: Represents the scale of confidence.
- **Arbitrary Elapsed Time Scale**: Represents the time required for integration.

The graph also indicates key points such as:

- **System anno 2000**: Refers to a point in time or scale.
- **Component**: Represents a higher level of integration.
- **Building Block**: Represents a lower level of integration.
- **File**: Represents the smallest unit of integration.
- **Size in LOC**: Represents the measure of code size.