Design Objectives and Design Understandability

by Gerrit Muller  Buskerud University College

e-mail: gaudisite@gmail.com

www.gaudisite.nl

Abstract
The complexity of systems limits the understanding by the architect of the impact of changes. Many objectives are pursued, from customer needs to implementation lessons learned, while designing a system. From architecting perspective understandability of the design is an important issue. Some design choices may create very efficient systems, but might be difficult to grasp. For example simple local autonomy might prove to be efficient and robust, but at the same time other system qualities are emerging and difficult to predict. We discuss the notion of understandability, illustrated by a number of design patterns.
1. performance example: do we understand our design?

2. complexity of context and system abstraction

3. cache example; impact of autonomous low-level mechanism on performance

4. discussion and conclusion

---

**Figure Of Contents™**

**1** performance example: do we understand our design?

**2** complexity of context and system abstraction

**3** cache example; impact of autonomous low-level mechanism on performance

**4** discussion and conclusion

---

**Design Objectives and Design Understandability**

2. Gerrit Muller

version: 0  
March 6, 2013
Image Retrieval Performance

Sample application code:
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}

alternative application code:
event 3*3 -> show screen 3*3

<screen 3*3>
    <row 1>
        <col 1><image 1,1></col 1>
        <col 2><image 1,2></col 2>
        <col 3><image 1,3></col 3>
    </row 1>
    <row 2>
        <col 1><image 1,1></col 1>
        <col 2><image 1,2></col 2>
        <col 3><image 1,3></col 3>
    </row 2>
    <row 3>
        <col 1><image 1,1></col 1>
        <col 2><image 1,2></col 2>
        <col 3><image 1,3></col 3>
    </row 3>
</screen 3*3>

application need:
at event 3*3 show 3*3 images instantaneous

Design Objectives and Design Understandability
3  Gerrit Muller

version: 0
March 6, 2013
PINTROsampleCode
What If....

Sample application code:

```plaintext
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}
```

UI process

store

screen
More Process Communication

What If....

Sample application code:

```java
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}
```

Design Objectives and Design Understandability

version: 0
March 6, 2013
PINTROwhatIf2
What If....

Meta
-----
---------
--------

Attribute = 1 COM object
100 attributes / image
9 images = 900 COM objects
1 COM object = 80µs
9 images = 72 ms

Sample application code:

```plaintext
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}
```
I/O overhead

What If....

Sample application code:

```java
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}
```

- I/O on line basis ($512^2$ image)

$$9 \times 512 \times t_{I/O}$$

$$t_{I/O} \approx 1 \text{ms}$$
Non Functional Requirements Require System View

Sample application code:

```java
for x = 1 to 3 {
    for y = 1 to 3 {
        retrieve_image(x,y)
    }
}
```

can be:
- fast, but very local
- slow, but very generic
- slow, but very robust
- fast and robust

... 

The emerging properties (behavior, performance) cannot be seen from the code itself!

Underlying platform and neighbouring functions determine emerging properties mostly.
Performance and behavior of a function depend on realizations of used layers, functions in the same context, and the usage context.
Challenge: How to understand MW, OS, HW with only a few parameters

Performance = Function (F&S, other F&S, MW, OS, HW)
MW, OS, HW >> 100 Manyear : very complex
performance example: do we understand our design?

What does Customer need in Product and Why?

How can the product be realized

What are the critical decisions

What does Customer need in Product and Why?

How can the product be realized

What are the critical decisions

Complexity of context and system abstraction

Cache example; impact of autonomous low-level mechanism on performance

Discussion and conclusion
Exponential Pyramid, from requirement to bolts and nuts
Major Bottleneck: Mental Dynamic Range

Design Objectives and Design Understandability

version: 0
March 6, 2013
ATmentalDynamicRange
Organizational Problem: Disconnect

**What** does Customer need in Product and **Why**?

How can the product be realized. What are the critical decisions.

- **Customer** objectives
- **Application**
- **Functional**
- **Conceptual**
- **Realisation**

system requirements
design decisions
parts connections
lines of code
and growing every year....

Design Objectives and Design Understandability

version: 0
March 6, 2013
RATWdisconnect
**Architect: Connecting Problem and Technical Solution**

<table>
<thead>
<tr>
<th><strong>What</strong> does Customer need in Product and <strong>Why?</strong></th>
<th><strong>How</strong> can the product be realized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer objectives</td>
<td><strong>What</strong> are the critical decisions</td>
</tr>
<tr>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>Functional</td>
<td></td>
</tr>
<tr>
<td>Conceptual</td>
<td></td>
</tr>
<tr>
<td>Realisation</td>
<td></td>
</tr>
</tbody>
</table>

- **What** does Customer need in Product and **Why?**
- **How** can the product be realized
- **What** are the critical decisions

**Design Objectives and Design Understandability**

- **system requirements**
- **design decisions**
- **parts connections lines of code**
- **and growing every year....**

Design Objectives and Design Understandability

**version: 0**
March 6, 2013
RATWbreadthAndDepth
1 performance example: do we understand our design?

Performance example:
do we understand our design?

2 complexity of context and system abstraction

What does Customer need in Product and Why?

How can the product be realized

What are the critical decisions

Customer objectives

Application

Functional

Conceptual

Realisation

3 cache example; impact of autonomous low-level mechanism on performance

hit rate is context dependent. Life cycle changes or peak loads may degrade hit rate.

Hit rate of well designed system is ample within working range (e.g. 95%).

0" order formula is valid:

Load(h) = \(2000 - 1980 \times h\) [ms]

Load(h) = \(2000 - 2000 \times h\) [ms]

Load(h) = \(2000 - 1.02 \times h\) [ms]

Load(h) = \(2000 - 2.02 \times h\) [ms]

4 discussion and conclusion
## Hierarchy of Storage Technology

### Figures of Merit

<table>
<thead>
<tr>
<th>Type</th>
<th>Technology</th>
<th>Latency</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor cache</td>
<td>L1 cache</td>
<td>sub ns</td>
<td>n kB</td>
</tr>
<tr>
<td></td>
<td>L2 cache</td>
<td>ns</td>
<td>n MB</td>
</tr>
<tr>
<td></td>
<td>L3 cache</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast volatile</td>
<td>main memory</td>
<td>tens ns</td>
<td>n GB</td>
</tr>
<tr>
<td>Persistent</td>
<td>disks</td>
<td>ms</td>
<td>n*100 GB</td>
</tr>
<tr>
<td></td>
<td>disk arrays</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>disk farms</td>
<td></td>
<td>n*10 TB</td>
</tr>
<tr>
<td>Archival</td>
<td>robotized optical media</td>
<td>&gt;s</td>
<td>n PB</td>
</tr>
<tr>
<td></td>
<td>tape</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Performance as Function of Data Set Size

random data processing performance in ops/s

<table>
<thead>
<tr>
<th>data set size in bytes</th>
<th>L1 cache</th>
<th>L3 cache</th>
<th>main memory</th>
<th>hard disk</th>
<th>disk farm</th>
<th>robotized media</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10^6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10^9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10^12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10^15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Design Objectives and Design Understandability

version: 0
March 6, 2013
MAFTstoragePerformance
## Multiple Layers of Caching

<table>
<thead>
<tr>
<th></th>
<th>Cache Miss Penalty</th>
<th>Cache Hit Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>application cache</strong></td>
<td>1 s</td>
<td>10 ms</td>
</tr>
<tr>
<td><strong>network layer cache</strong></td>
<td>100 ms</td>
<td>1 ms</td>
</tr>
<tr>
<td><strong>file cache</strong></td>
<td>10 ms</td>
<td>10 us</td>
</tr>
<tr>
<td><strong>virtual memory</strong></td>
<td>1 ms</td>
<td>100 ns</td>
</tr>
<tr>
<td><strong>memory caches L1, L2, L3</strong></td>
<td>100 ns</td>
<td>1 ns</td>
</tr>
</tbody>
</table>

![Diagram showing multiple layers of caching](image-url)
Why Caching?

- **Project Risk**
  - Performance
  - Response Time

- **Life Cycle**
  - Cost

**Design Objectives and Design Understandability**

- **Limit Storage**
  - Needs to fit in fast local storage

- **Frequently Used Subset**

- **Design Parameters**
  - Caching Algorithm
  - Storage Location
  - Cache Size
  - Chunk Size
  - Format

- **Low Latency**
  - Fast Storage

- **Less Communication**
  - Local Storage

- **Latency Penalty Once**
  - Overhead Once

- **Resource Intensive Processing**

- **Processing Once**

- **In (Pre)Processed Format**

- **Overhead Once**
  - Communication

- **Long Latency**
  - Mass Storage

- **Long Latency**
  - Communication
Example Web Shop

Design Objectives and Design Understandability

version: 0
March 6, 2013
MAFExampleWebShop
Impact of Picture Cache

- Fast response
- Less load
- Less server costs

Design Objectives and Design Understandability

23  Gerrit Muller

version: 0
March 6, 2013
MAFTwebShopPictureCache
Risks of Caching

- frequently used subset
- fast storage
- local storage
- larger chunks
- in (pre)processed format
- robustness for application changes
- ability to benefit from technology improvements
- robustness for changing context (e.g. scalability)
- robustness for concurrent applications
- failure modes in exceptional user space

Design Objectives and Design Understandability
24 Gerrit Muller

version: 0 March 6, 2013 MAFTrisksOfCaching
Zero Order Load Model

**zero order web server load model**

\[
\text{Load} = n_a \times t_a
\]

- \(n_a\) = total requests
- \(t_a\) = cost per request
**First Order Load Model**

*first order web server load model*

\[
\text{Load} = n_{a,h} \cdot t_h + n_{a,m} \cdot t_m
\]

- \( n_{a,h} = \text{accesses with cache hit} \)
- \( n_{a,m} = \text{accesses with cache miss} \)
- \( t_h = \text{cost of cache hit} \)
- \( t_m = \text{cost of cache miss} \)

\[
n_{a,h} = n_a \cdot h
\]

\[
n_{a,m} = n_a \cdot (1-h)
\]

- \( n_a = \text{total accesses} \)
- \( h = \text{hit rate} \)

\[
\text{Load}(h) = n_a \cdot h \cdot t_h + n_a \cdot (1-h) \cdot t_m
\]

\[
= n_a \cdot t_m - n_a \cdot h \cdot (t_m - t_h)
\]
Quantification: From Formulas to Insight

quantified mid office server example

\[ t_h = 0.02 \text{ ms} \]
\[ t_m = 2 \text{ ms} \]
\[ n_a = 1000 \]

Load(h) = 1000 * 2[ms] - 1000 * h * 1.98[ms]

Load(h) = 2000 - 1980 * h [ms]

---

Design Objectives and Design Understandability

version: 0
March 6, 2013
MASTMquantified
Hit Rate Considerations

quantified mid office server example

\[ t_h = 0.02 \text{ ms} \]
\[ t_m = 2 \text{ ms} \]
\[ n_a = 1000 \]

Load(h) = \[ 1000 \times 2 \text{[ms]} - 1000 \times h \times 1.98 \text{[ms]} \]
Load(h) = \[ 2000 - 1980 \times h \text{[ms]} \]

Hit rate of well designed system is ample within working range (e.g. 95%)
0th order formula is valid:
Load = \[ 0.12 \times n_a \text{[ms]} \]

Hit rate is context dependent.
Life cycle changes or peak loads may degrade hit rate.
Response Time

Design Objectives and Design Understandability
29  Gerrit Muller  version: 0  March 6, 2013  MASMresponseTime
1 performance example: do we understand our design?

2 complexity of context and system abstraction

3 cache example; impact of autonomous low-level mechanism on performance

4 discussion and conclusion
central full control does not imply understandability

delegated autonomous behavior does not imply understandability

a few simple rules can create very complex behavior

understanding does not imply determinism or predictability

valid abstractions facilitate understanding

simulations provide numbers, not understanding

only humans understand!
Conclusions

control, predictability, and determinism are illusions

simple rules can create complex non-understandable systems

challenge: to model systems at "right" abstraction level