

# Introduction to System Performance Design

by *Gerrit Muller* Embedded Systems Institute

e-mail: `gerrit.muller@embeddedsystems.nl`

`www.gaudisite.nl`

## Abstract

What is System Performance? Why should a software engineer have knowledge of the other parts of the system, such as the Hardware, the Operating System and the Middleware? The applications that he/she writes are self-contained, so how can other parts have any influence? This introduction sketches the problem and shows that at least a high level understanding of the system is very useful in order to get optimal performance.

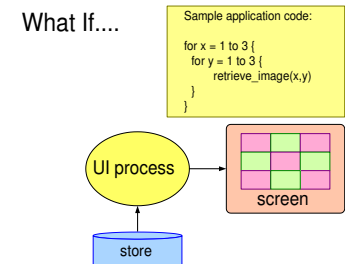
## Distribution

This article or presentation is written as part of the Gaudí project. The Gaudí project philosophy is to improve by obtaining frequent feedback. Frequent feedback is pursued by an open creation process. This document is published as intermediate or nearly mature version to get feedback. Further distribution is allowed as long as the document remains complete and unchanged.

February 10, 2011

status: preliminary  
draft

version: 0.5



# Content of Problem Introduction

---

*content of this presentation*

Example of problem

Problem statements

# Image Retrieval Performance

application need:

at event 3\*3 show 3\*3 images  
instantaneous

design

design

Sample application code:

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

or

alternative application code:

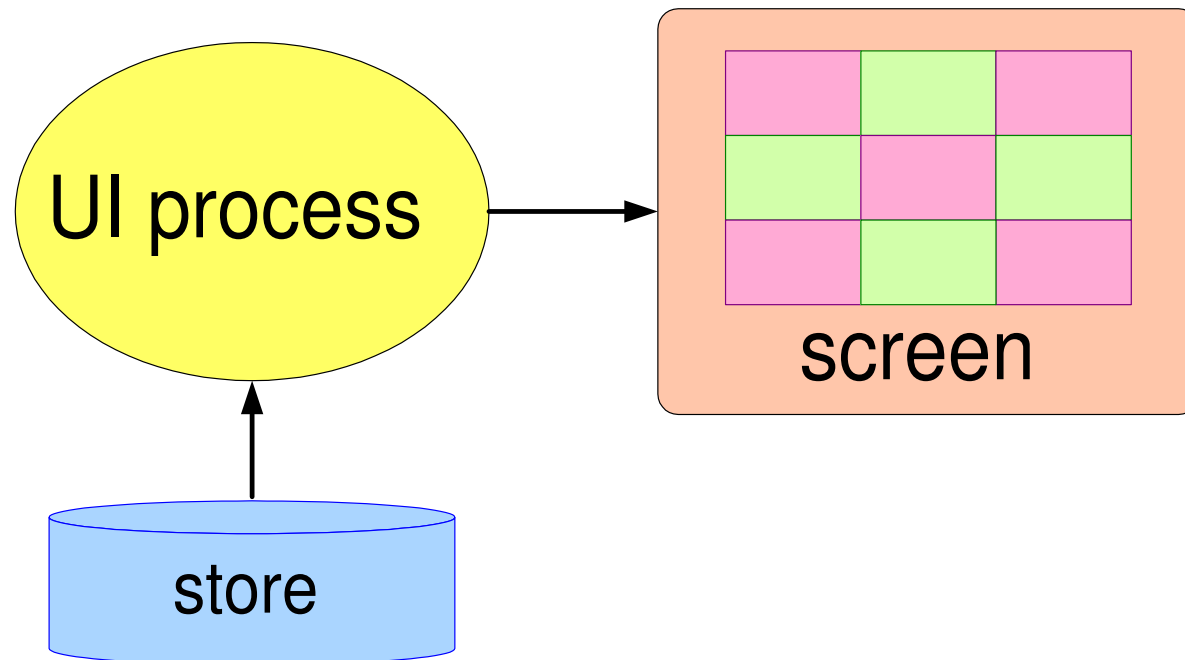
event 3\*3 -> show screen 3\*3

```
<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 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 3>  
</screen 3*3>
```

## What If....

Sample application code:

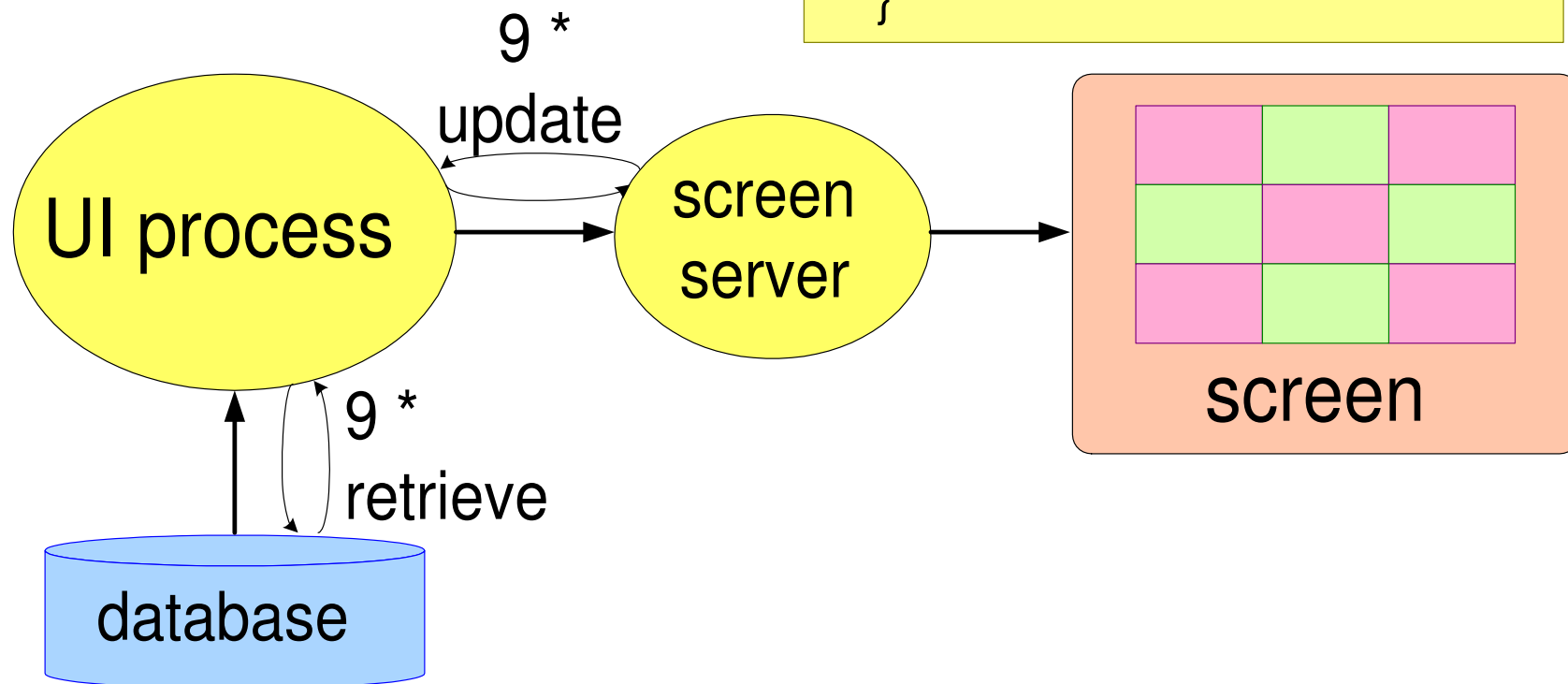
```
for x = 1 to 3 {  
  for y = 1 to 3 {  
    retrieve_image(x,y)  
  }  
}
```



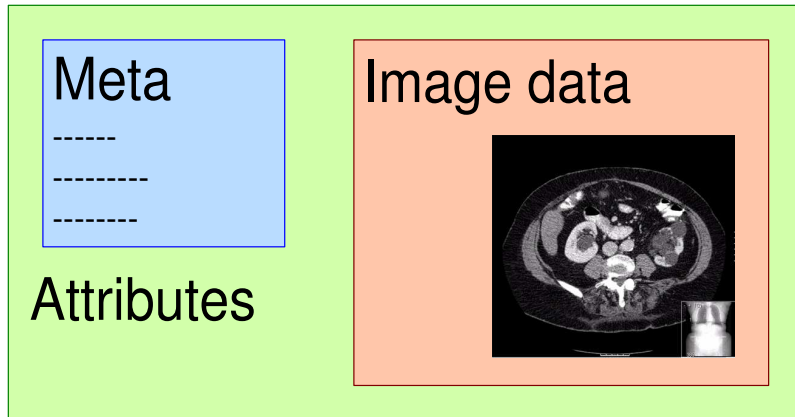
## What If....

Sample application code:

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

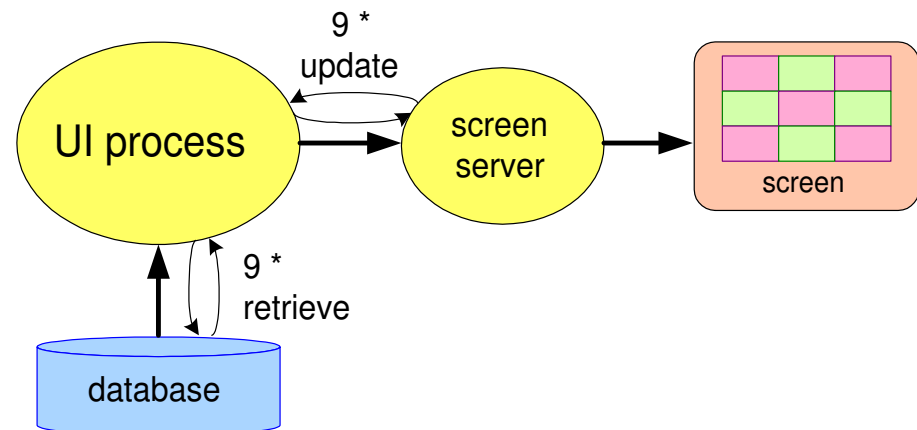


## What If....



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

Attribute = 1 COM object  
100 attributes / image  
9 images = 900 COM objects  
1 COM object = 80 $\mu$ s  
9 images = 72 ms



## What If....

Sample application code:

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

- I/O on line basis (512<sup>2</sup> image)

$$9 * 512 * t_{I/O}$$

$$t_{I/O} \approx 1ms$$

- . . .

# Non Functional Requirements Require System View

Sample application code:

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

can be:

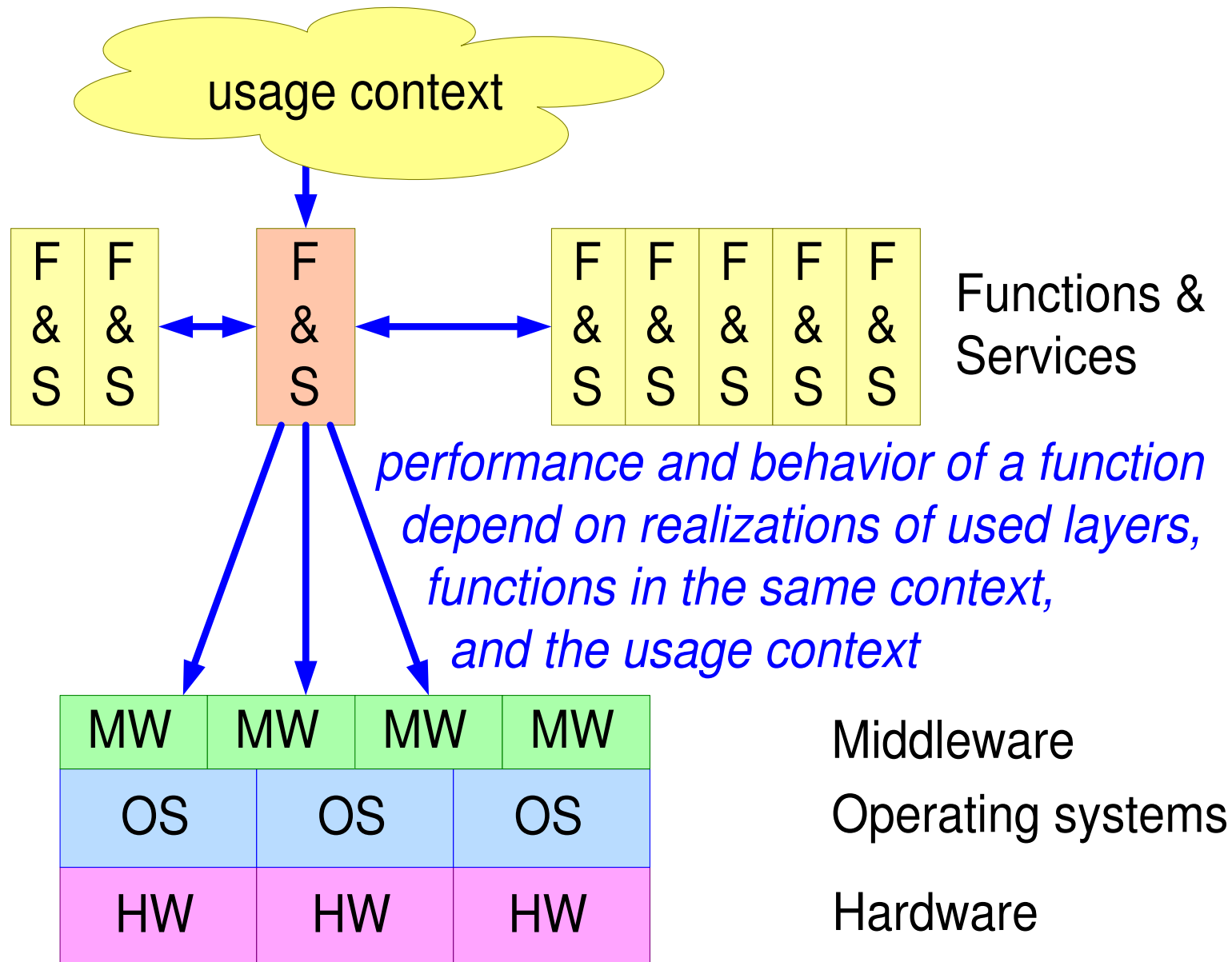
fast, but very local  
slow, but very generic  
slow, but very robust  
fast and robust

...

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

*Underlying platform and neighbouring functions determine emerging properties mostly.*

# Function in System Context



# Challenge

F	F	F	F	F	F	F	F
&	&	&	&	&	&	&	&
S	S	S	S	S	S	S	S
MW		MW		MW		MW	
OS			OS			OS	
HW		HW		HW			

Functions & Services

Middleware

Operating systems

Hardware

Performance = Function (F&S, other F&S, MW, OS, HW)

MW, OS, HW >> 100 Manyear : very complex

Challenge: How to understand MW, OS, HW  
with only a few parameters

## *Summary of Introduction to Problem*

Resulting System Characteristics cannot be deduced from local code.

Underlying platform, neighboring applications and user context:

- have a big impact on system characteristics

- are big and complex

Models require decomposition, relations and representations to analyse.

The ASP <sup>TM</sup> course is partially derived from the EXARCH course developed at *Philips CTT* by *Ton Kostelijk* and *Gerrit Muller* .

Extensions and additional slides have been developed at *ESI* by *Teun Hendriks* , *Roland Mathijssen* and *Gerrit Muller* .