### Bloating, Waste, and Value

by Gerrit Muller TNO-ESI, University of South-Eastern Norway]

e-mail: gaudisite@gmail.com

www.gaudisite.nl

#### **Abstract**

A threat to performance is the combination of feature creep and technical debt. This combination causes bloating of the design. In Lean terms, the combination causes waste. A crucial question is where is the value, and is the value in balance with the potential degradation of performance.

Distribution

and unchanged.

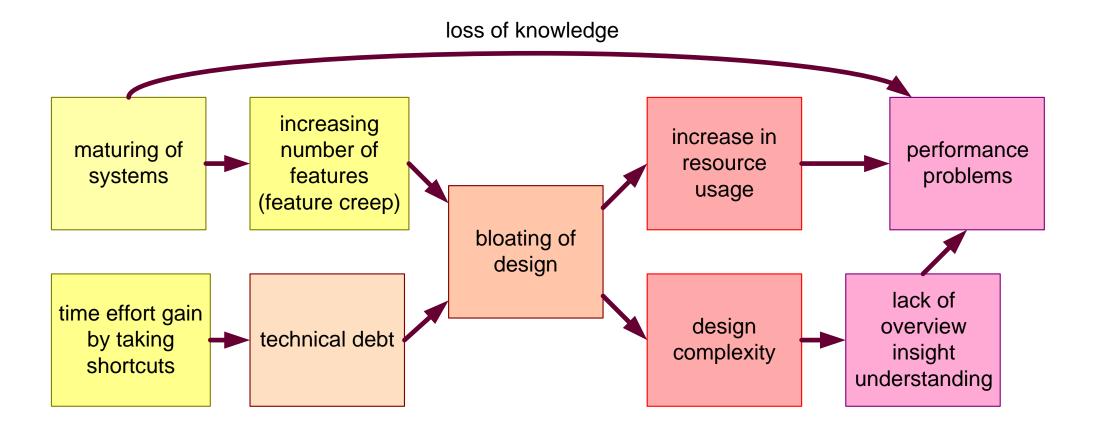
September 1, 2020 status: planned version: 0

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

logo

TBD

## From Feature Creep to Performance Problems





### **Technical Debt**

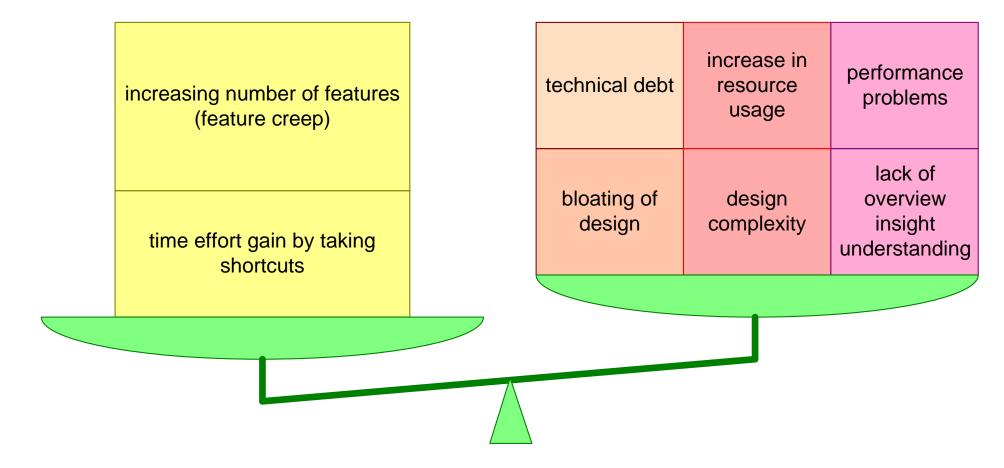
Technical Debt is a metaphor used within the software industry to communicate the consequences of pragmatic design decisions deviating from the intended design of a system

from: http://gaudisite.nl/INCOSE2016\_Callister\_Andersson\_SMARTtechnicalDebt.pdf based on Cunningham http://c2.com/doc/oopsla92.html



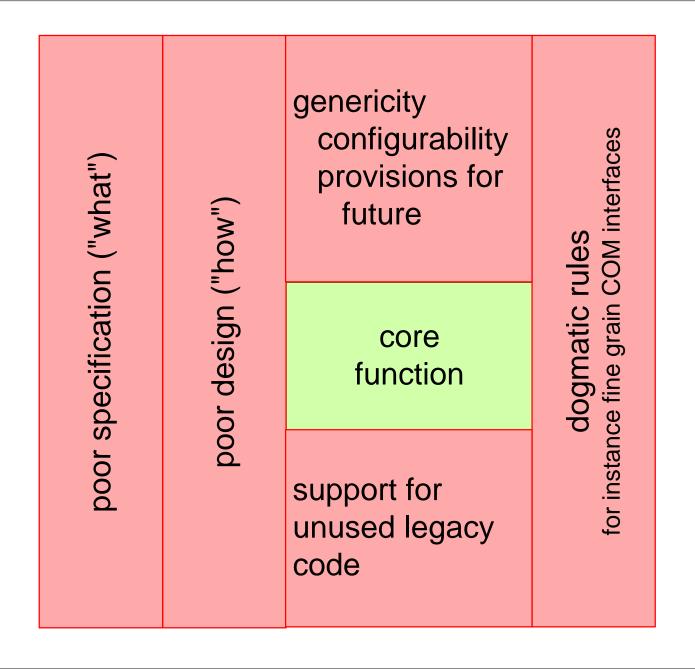
### Value versus Performance Degradation

Are benefits (value) in balance with the costs (such as performance degradation)?





### Exploring bloating: main causes



legenda

overhead

value



### Necessary functionality ≫ the intended regular function

# testing

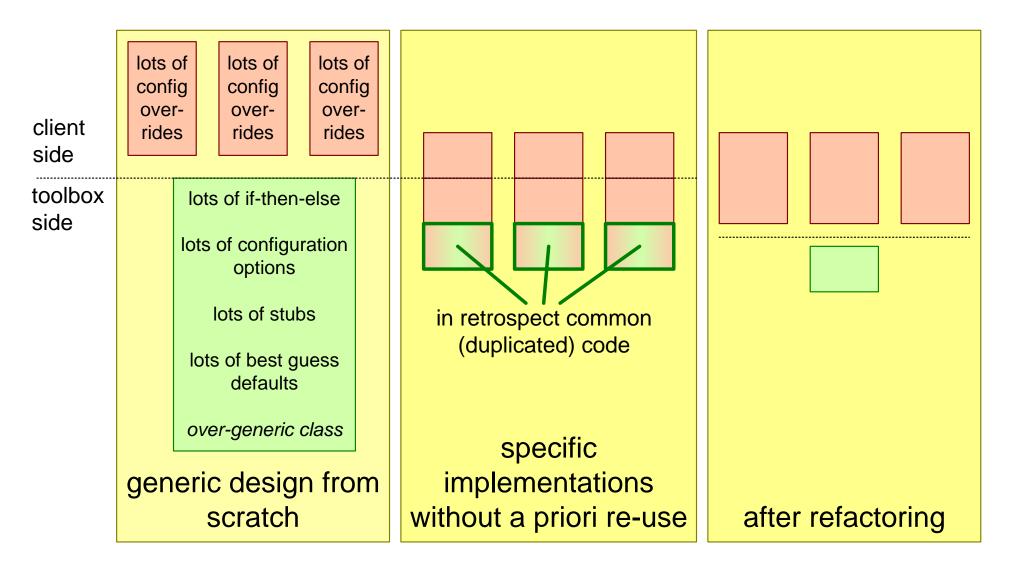
regular functionality

instrumentation diagnostics tracing asserts

boundary behavior: exceptional cases error handling



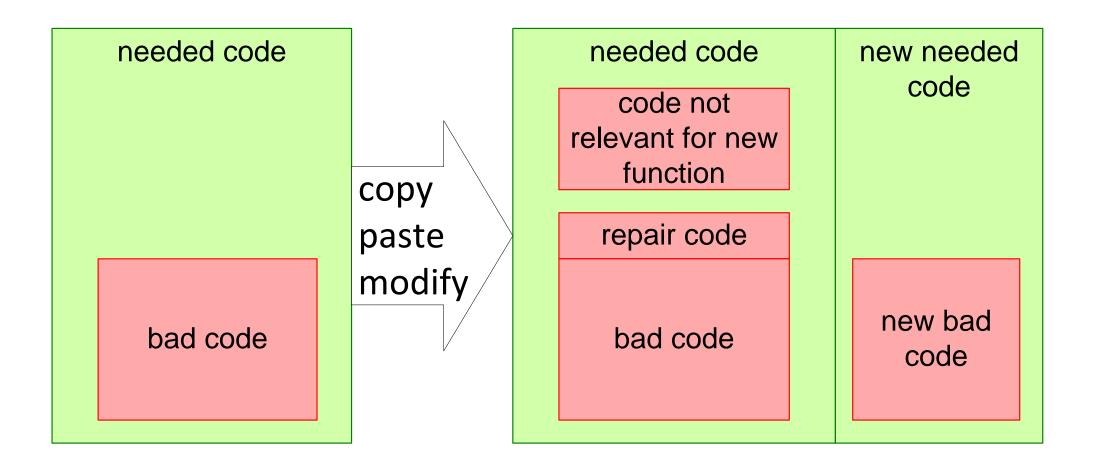
### The danger of being generic: bloating



"Real-life" example: redesigned Tool super-class and descendants, ca 1994



### Shit propagation via copy paste





### Example of shit propagation

copy

paste

```
Class Old:
    capacity = startCapacity
    values = int(capacity)
    size = 0

def insert(val):
    values[size]=val
    size+=1
    if size>capacity:
        capacity*=2
    relocate(values,
        capacity)
```

```
Class New:

capacity = 1

values = int(capacity)

size = 0

def insert(val):

values[size]=val

size+=1

capacity+=1

relocate(values,

capacity)

capacity)
```

```
Class DoubleNew:

capacity = 1

values = int(capacity)

size = 0

def insert(val):

values[size]=val

size+=1

capacity+=1

relocate(values,

capacity)

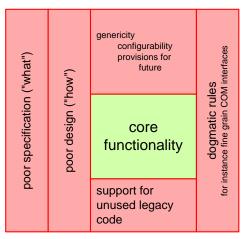
def insertBlock(v,len):

for i=1 to len:

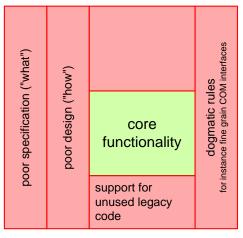
insert(v[i])
```

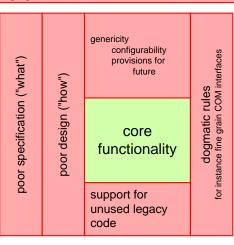


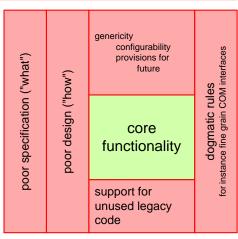
### Bloating causes more bloating











legenda

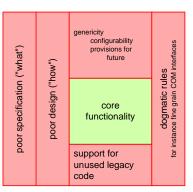
overhead

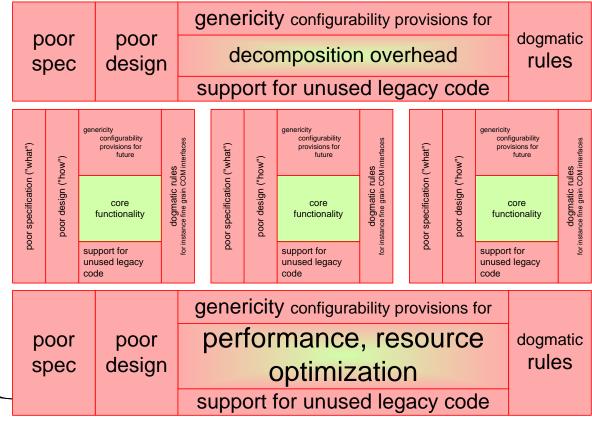
value



### Causes even more bloating...

Bloating causes performance and resource problems.
Solution: special measures: memory pools, shortcuts, ...





legenda

overhead

value

