

# Architectural Refactoring; illustrated by MR

by *Gerrit Muller* Embedded Systems Institute

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

`www.gaudisite.nl`

## Abstract

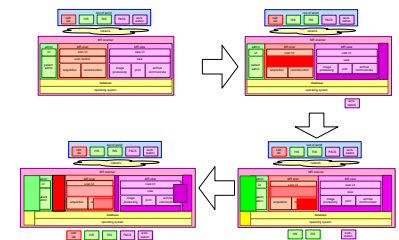
The market of medical appliances shows a fast increasing diversity. Manufacturers must be able to combine existing functions and new applications in a short time frame. A large amount of accumulated SW code (legacy) has to be reused in new ways.

The architecture(s) must be adapted to these new ways of working. Revolutionary adaptations have proven to be extremely risky. Opportunistic extension and integration decrease the quality of the code base, making it increasingly more difficult to continue. Architectural refactoring is a feedback based method to evolve an architecture.

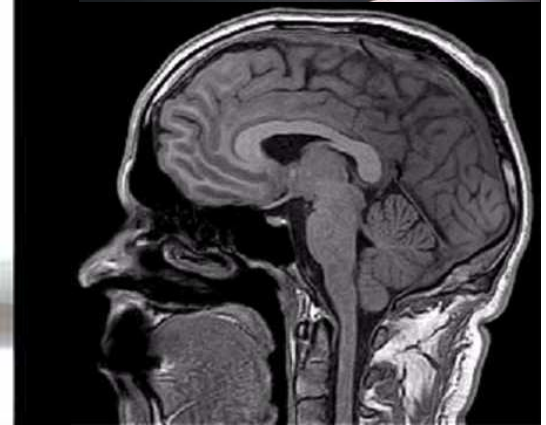
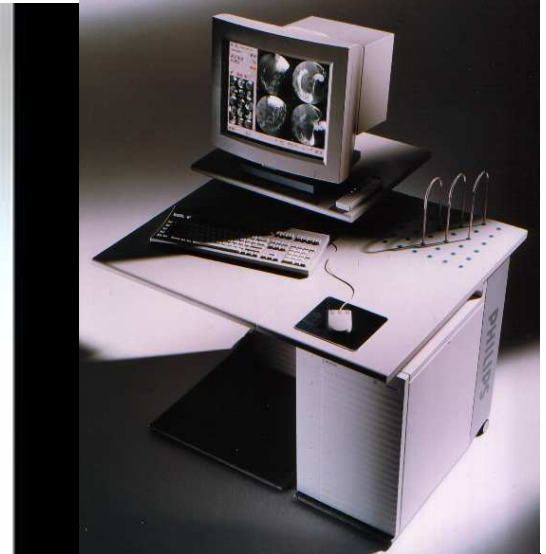
### Distribution

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: 0.1



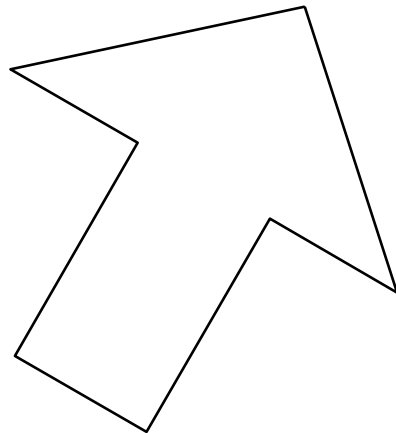
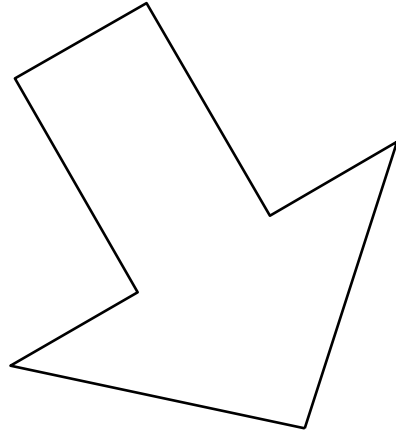
# Today's Medical Products



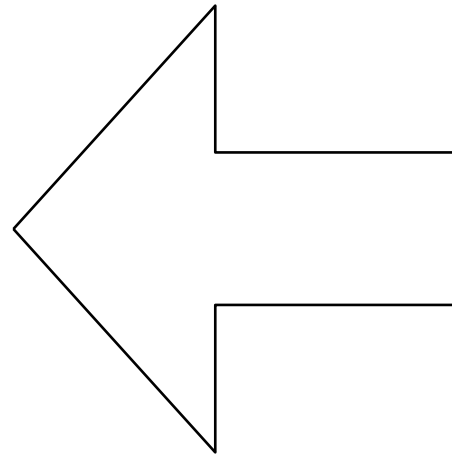
# Trend: Convergence of separate worlds

---

Telecom

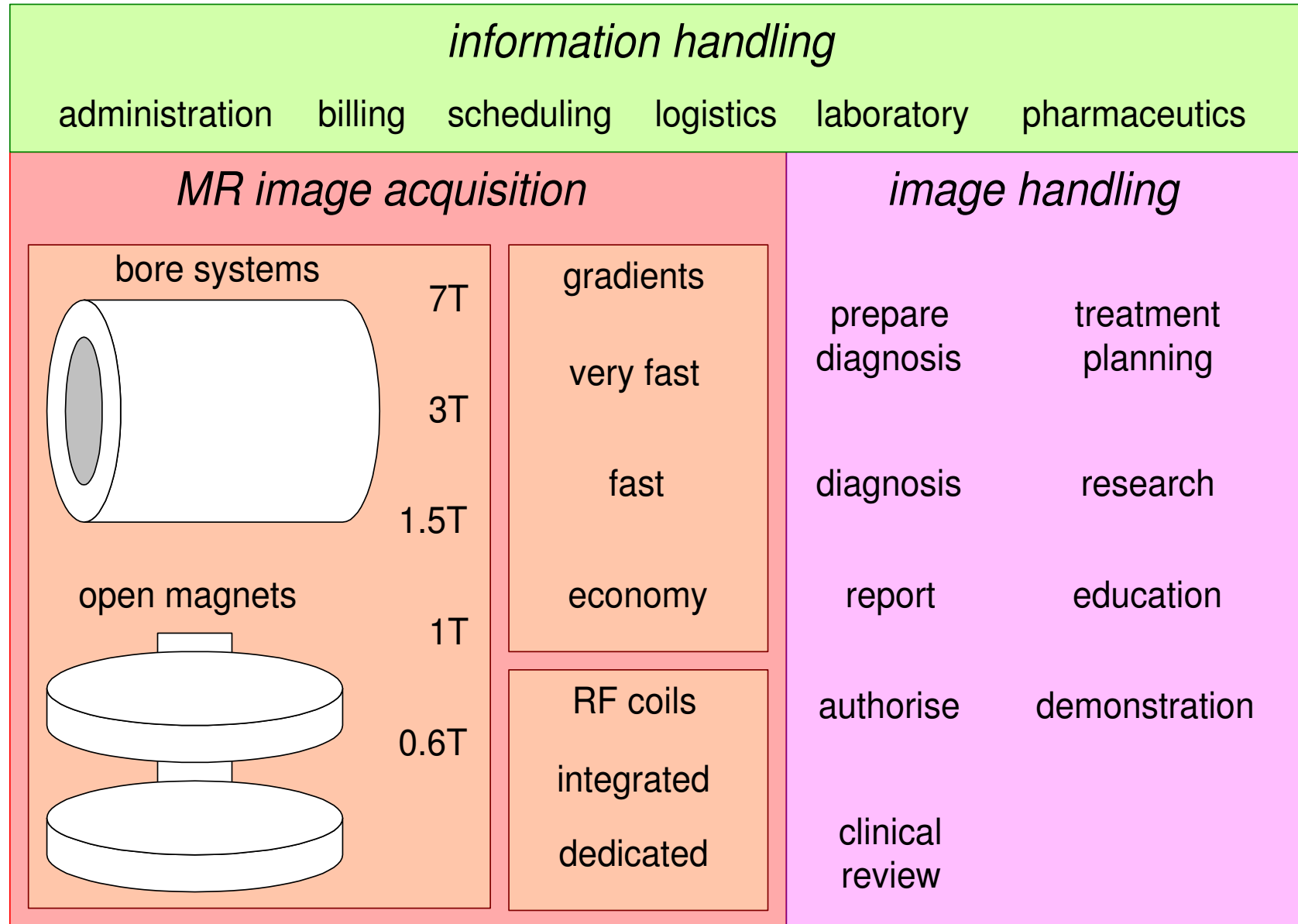


Medical

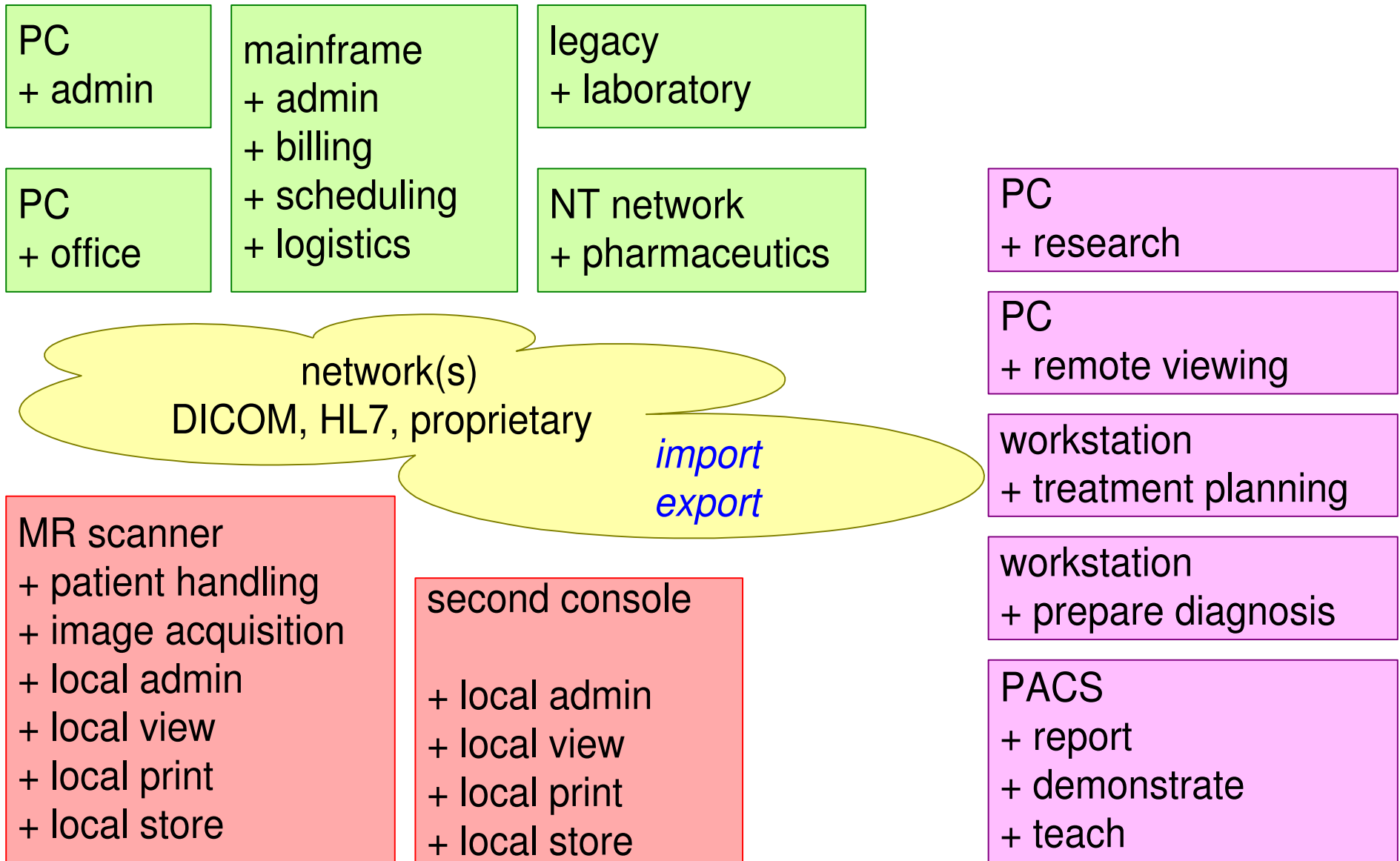


Computer

# Integration and Diversity

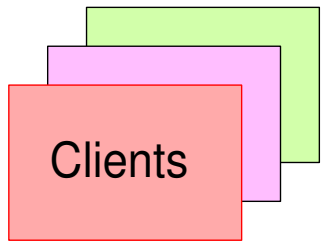


# Today's Medical Products

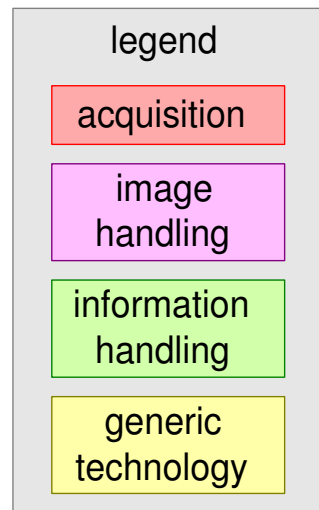
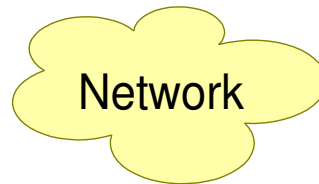


# Distribution Scenario's

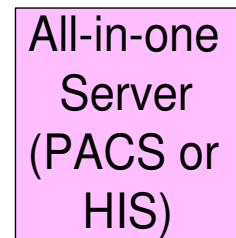
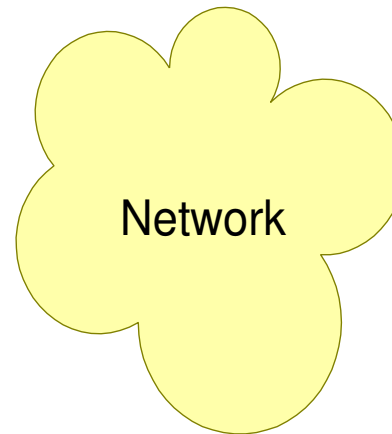
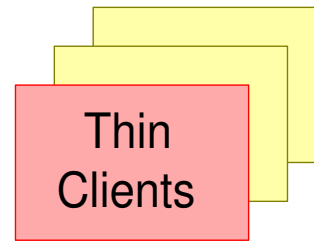
A "Thin Servers"



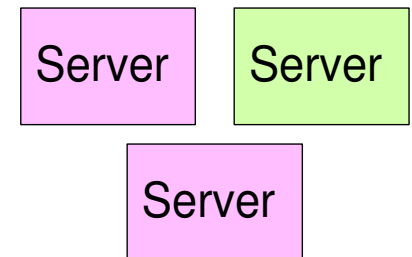
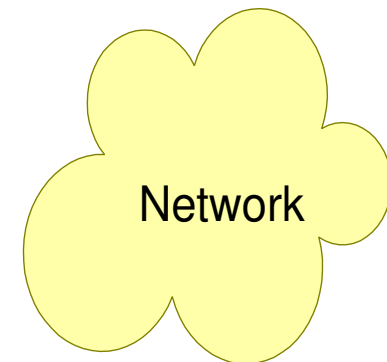
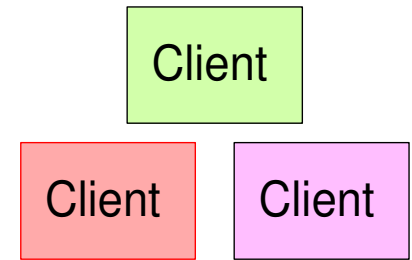
B "All-in-one" Combi's



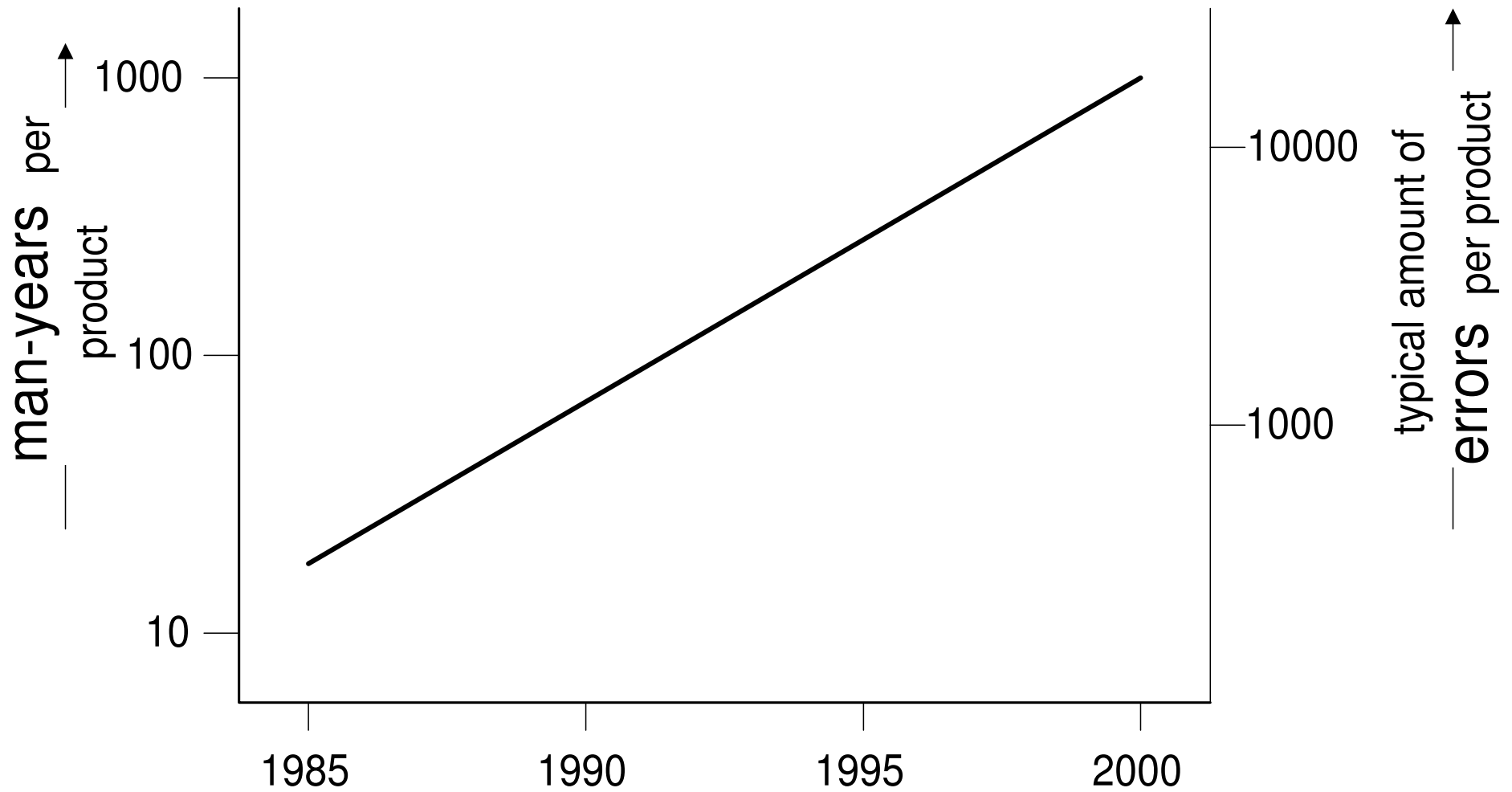
C "All-in-one" server



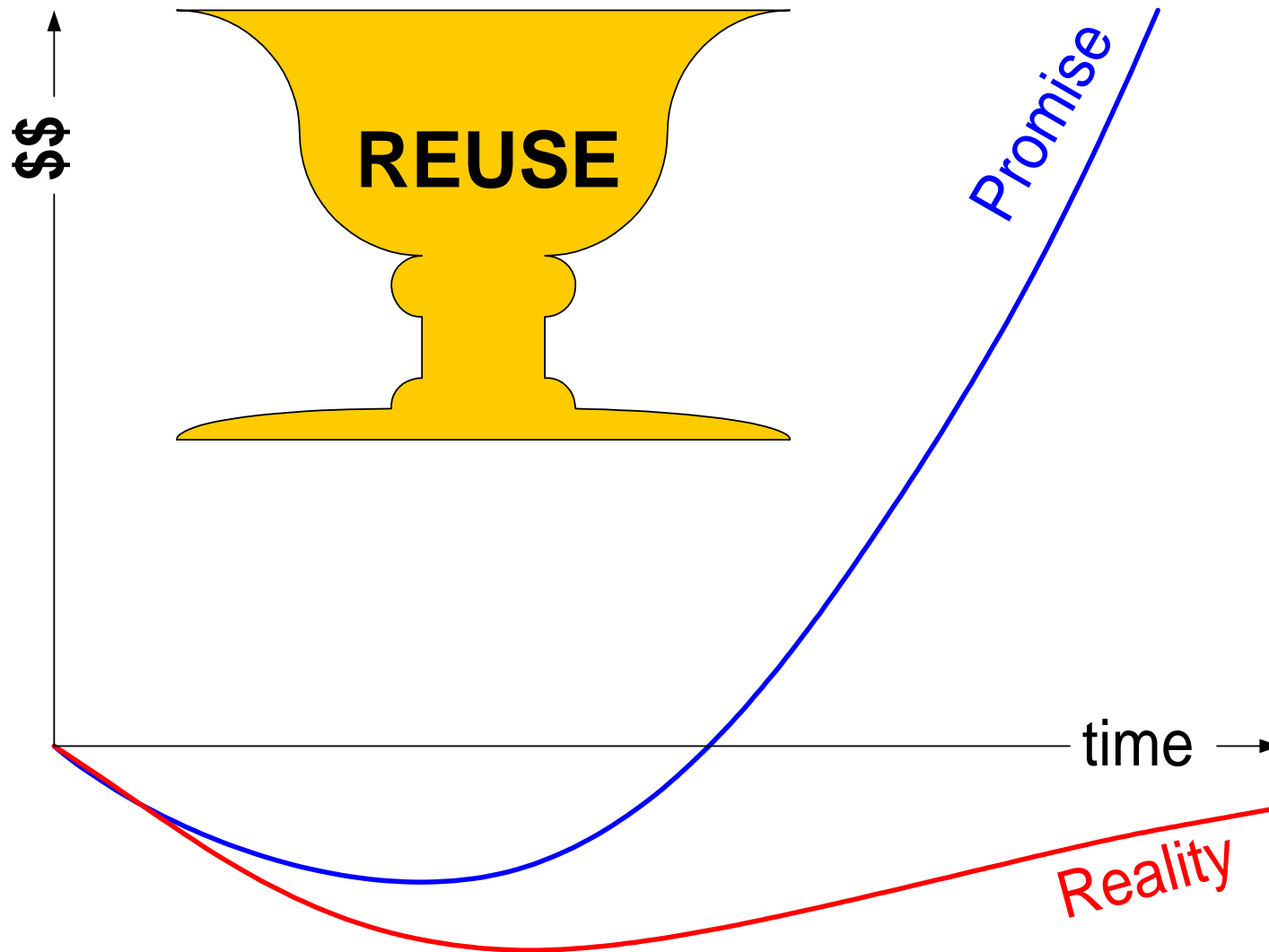
D "Modular" architecture



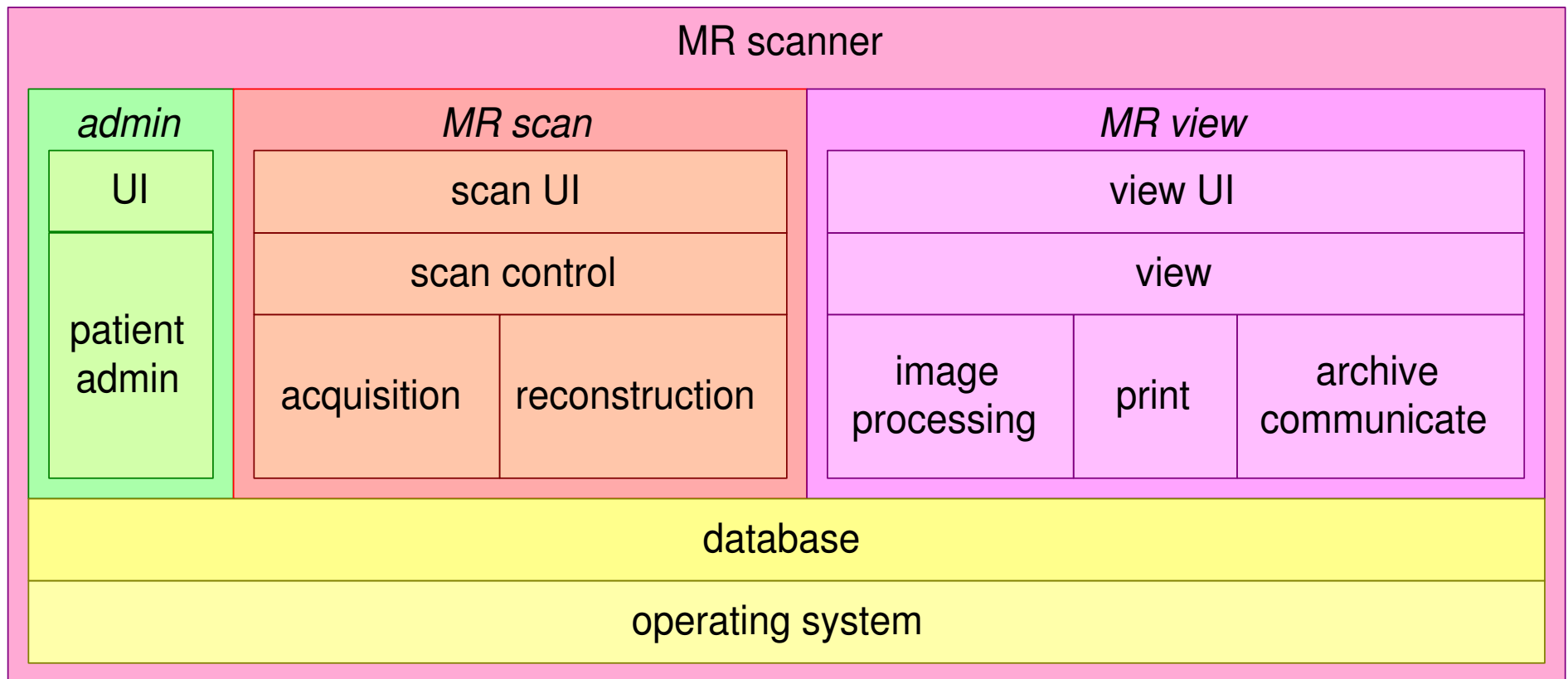
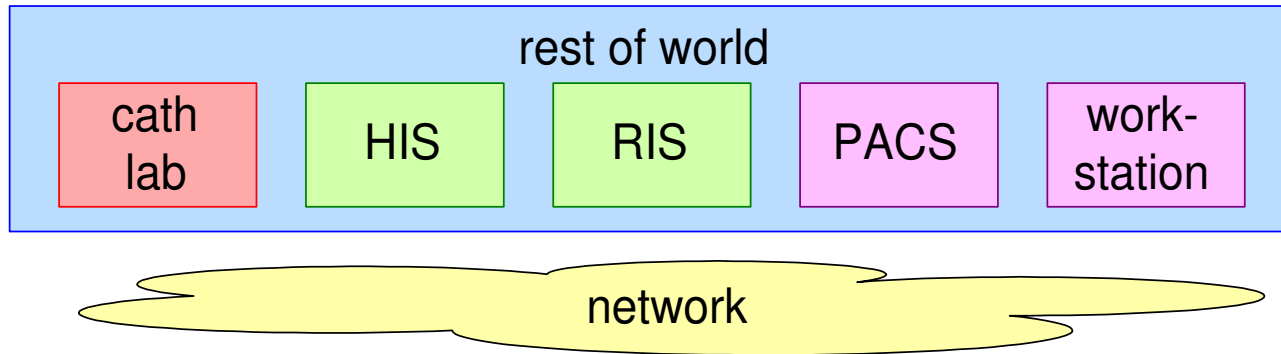
# Problem: increasing SW size, decreasing reliability?



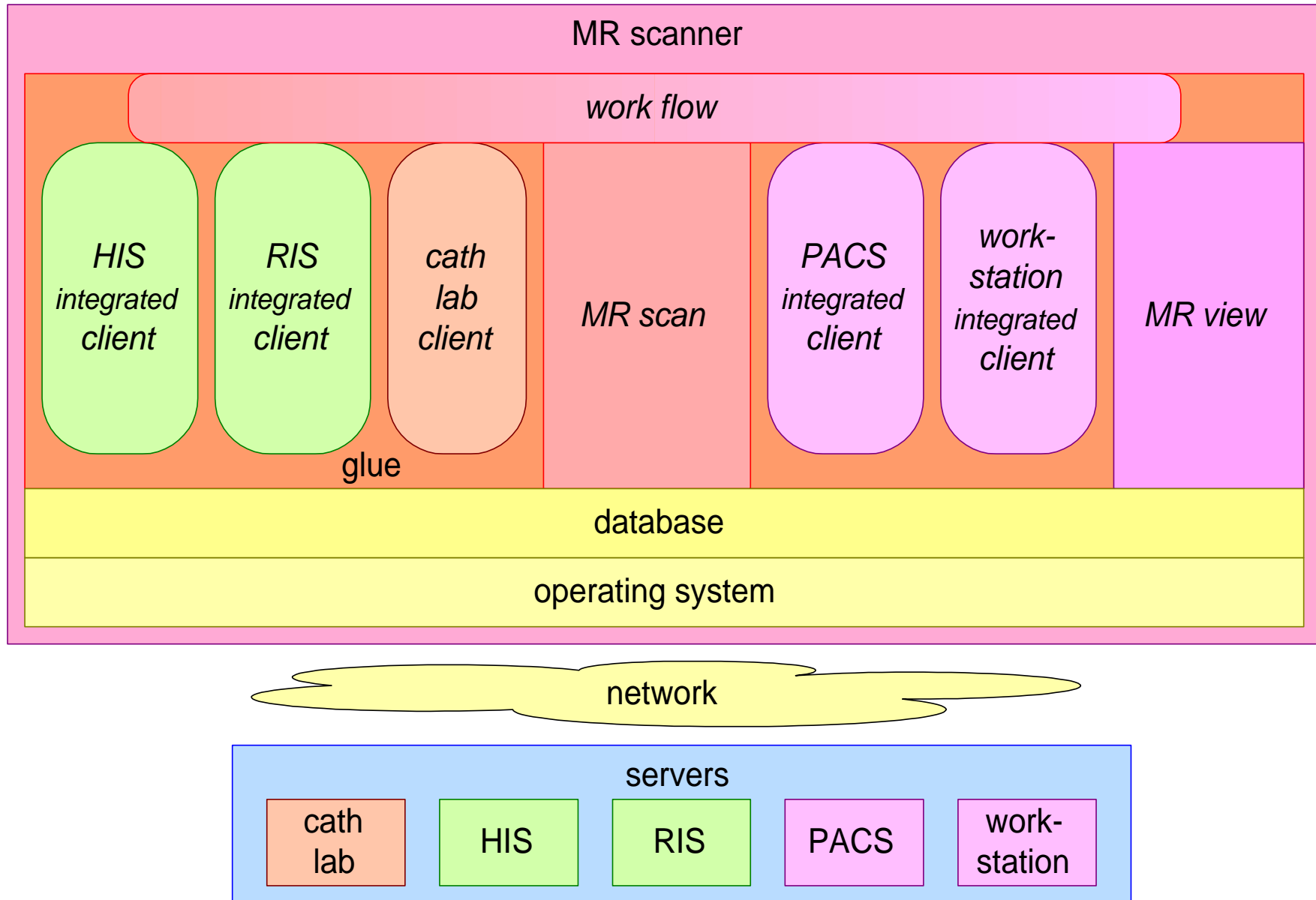
# The Holy Grail: Reuse



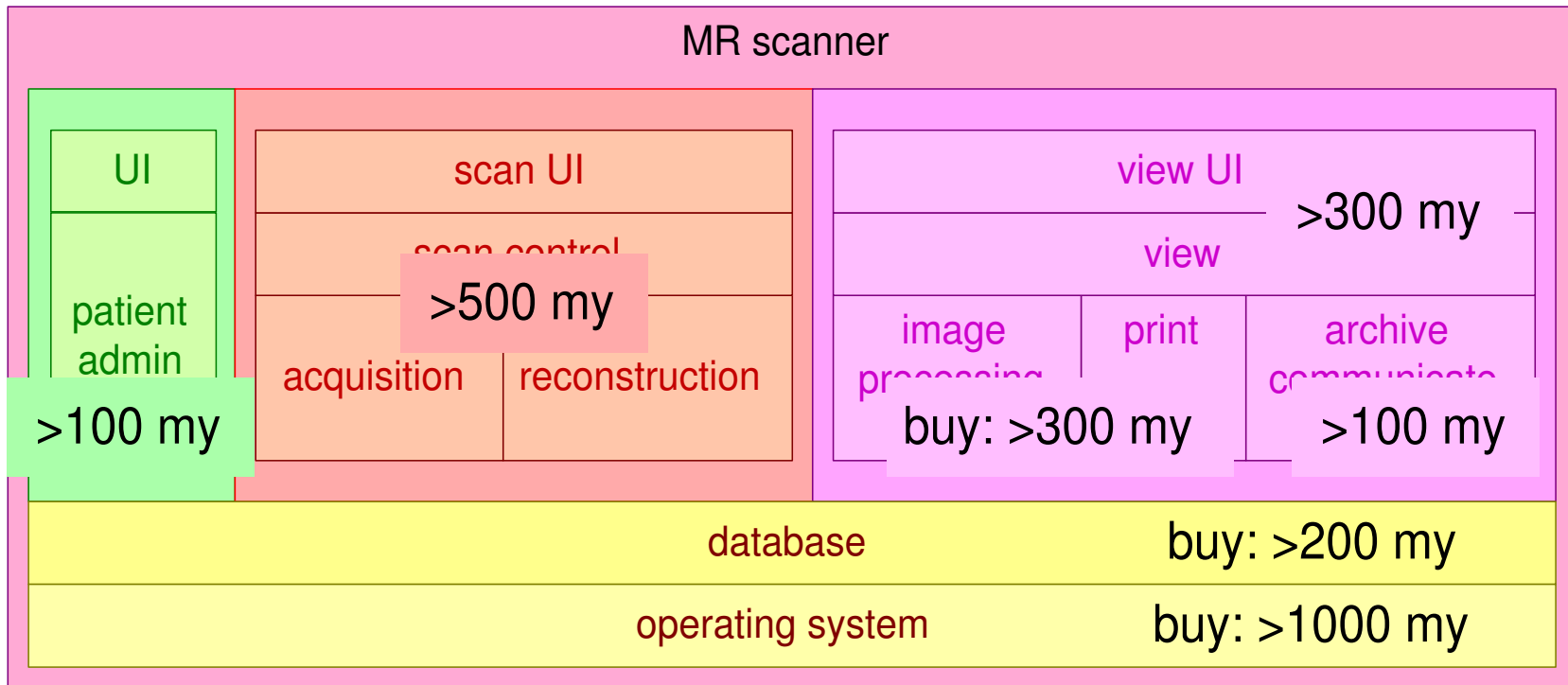
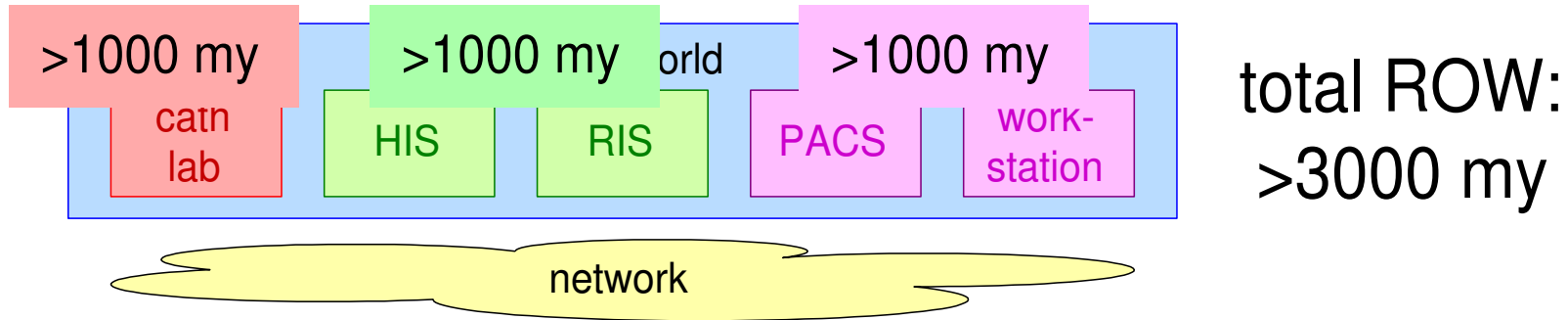
# Simplistic Architecture



# Future Simplistic Architecture



# Available Code Assets

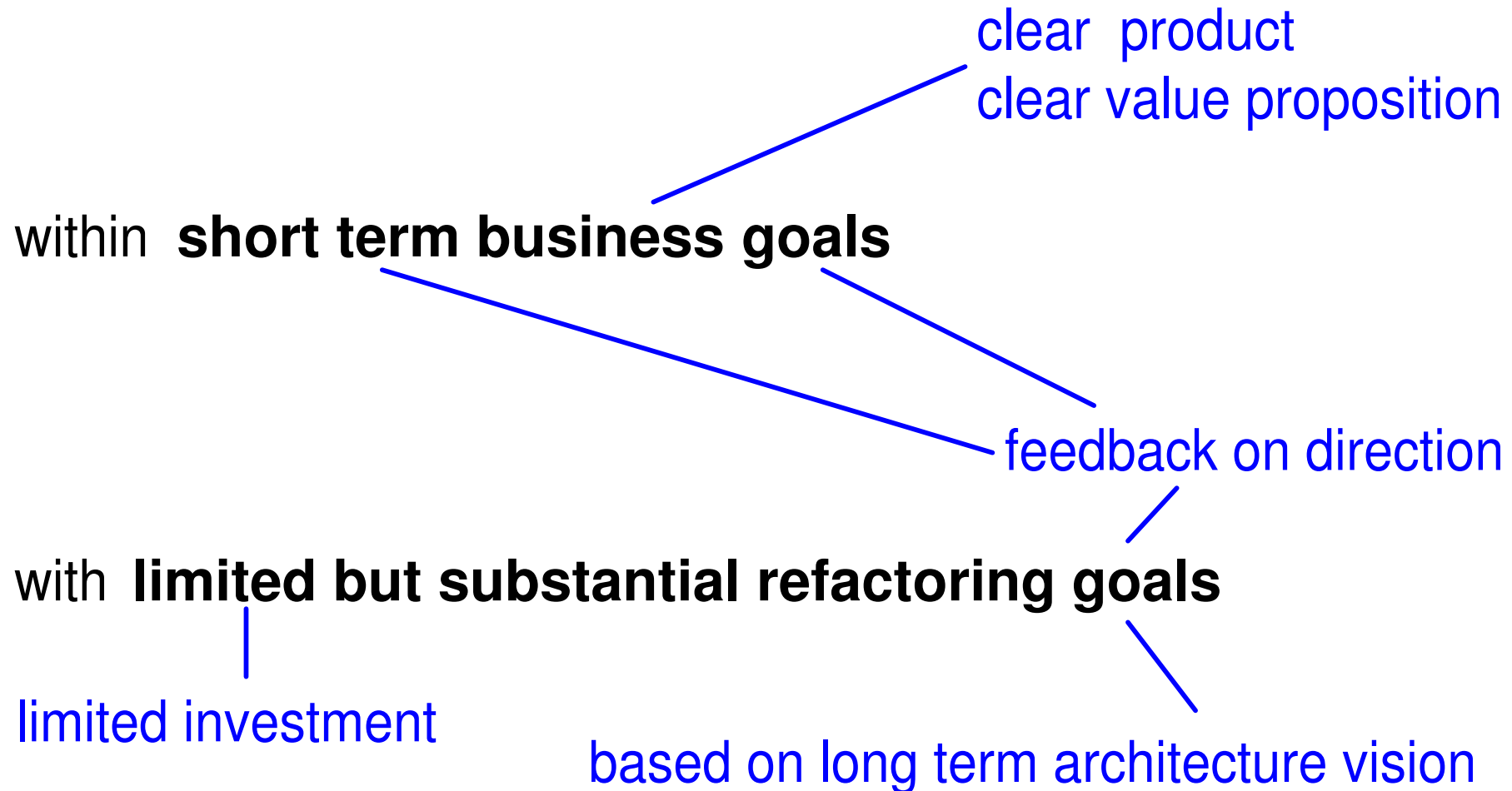


total make: >1000 my

total buy: >1500 my



## Refactoring



# Example of Refactoring Goals

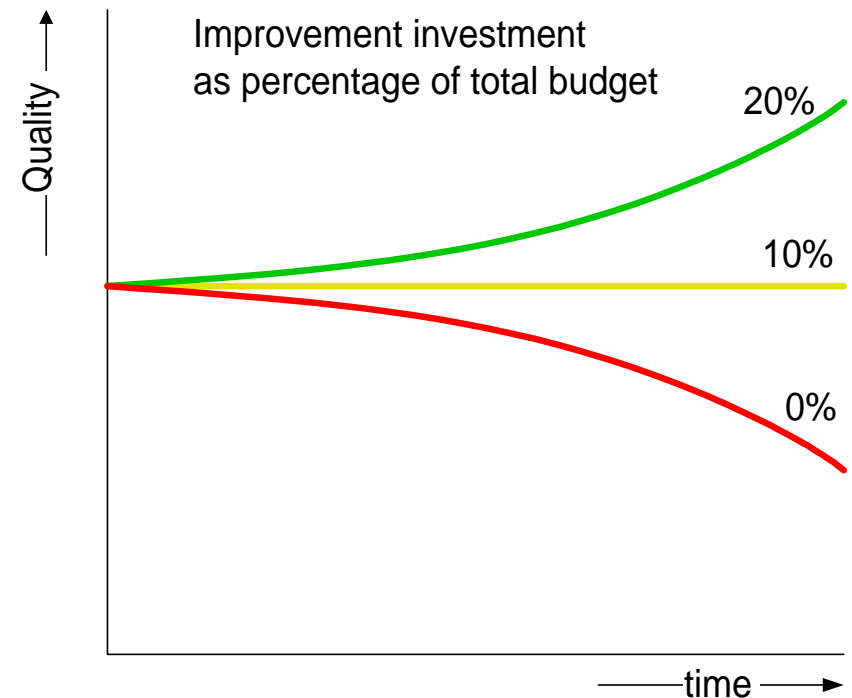
+ Decrease Code Size

+ Decrease Resource Usage

- \* power
- \* memory
- \* silicon area

+ Increase Performance

- \* response time
- \* throughput

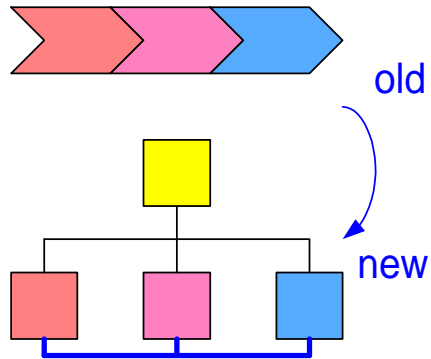


+ Increase quality  
\* decrease fault density

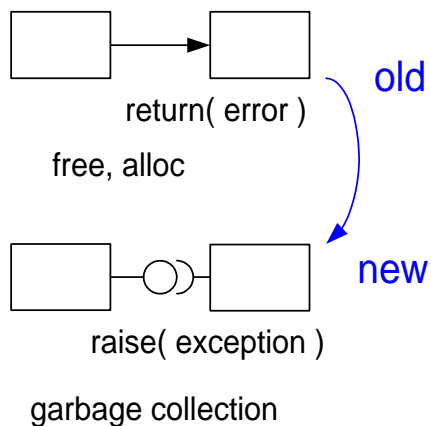
# Architectural vs Code refactoring

## Architectural Refactoring

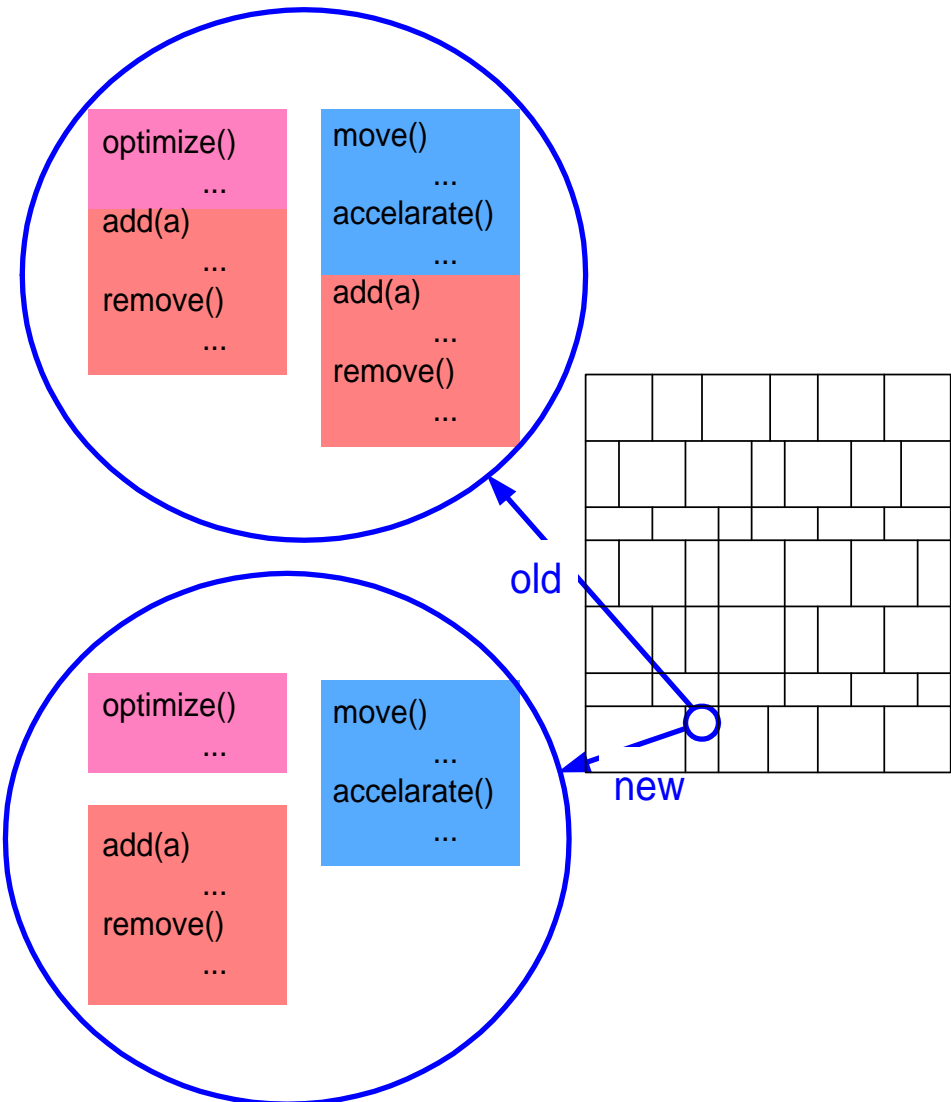
Function, Structure, Rationale



Mechanisms, Technologies



## Code Refactoring

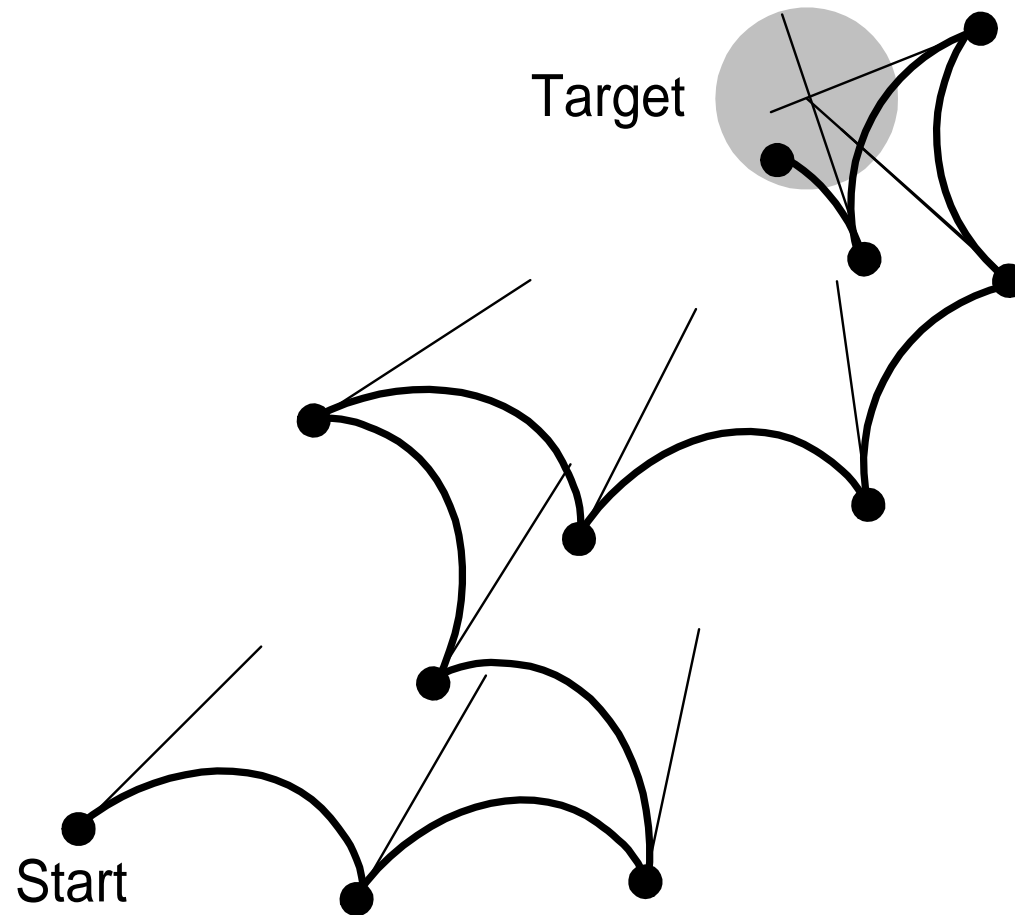


## Frequent feedback

# Feedback

---

stepsize: 3 months  
elapsed time: 25 months

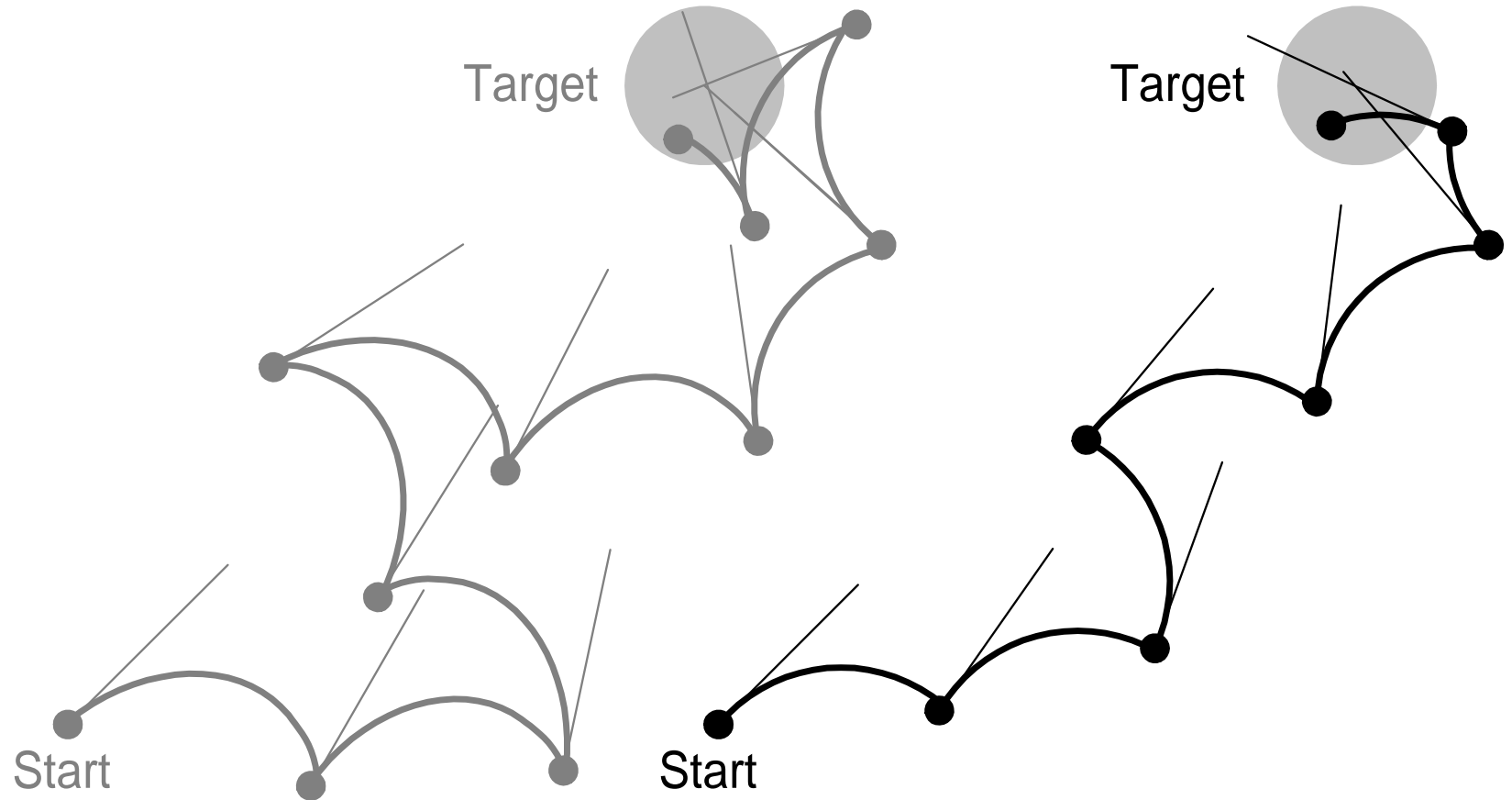


# Feedback (2)

stepsize:  
elapsed time

3 months  
25 months

2 months  
12 months



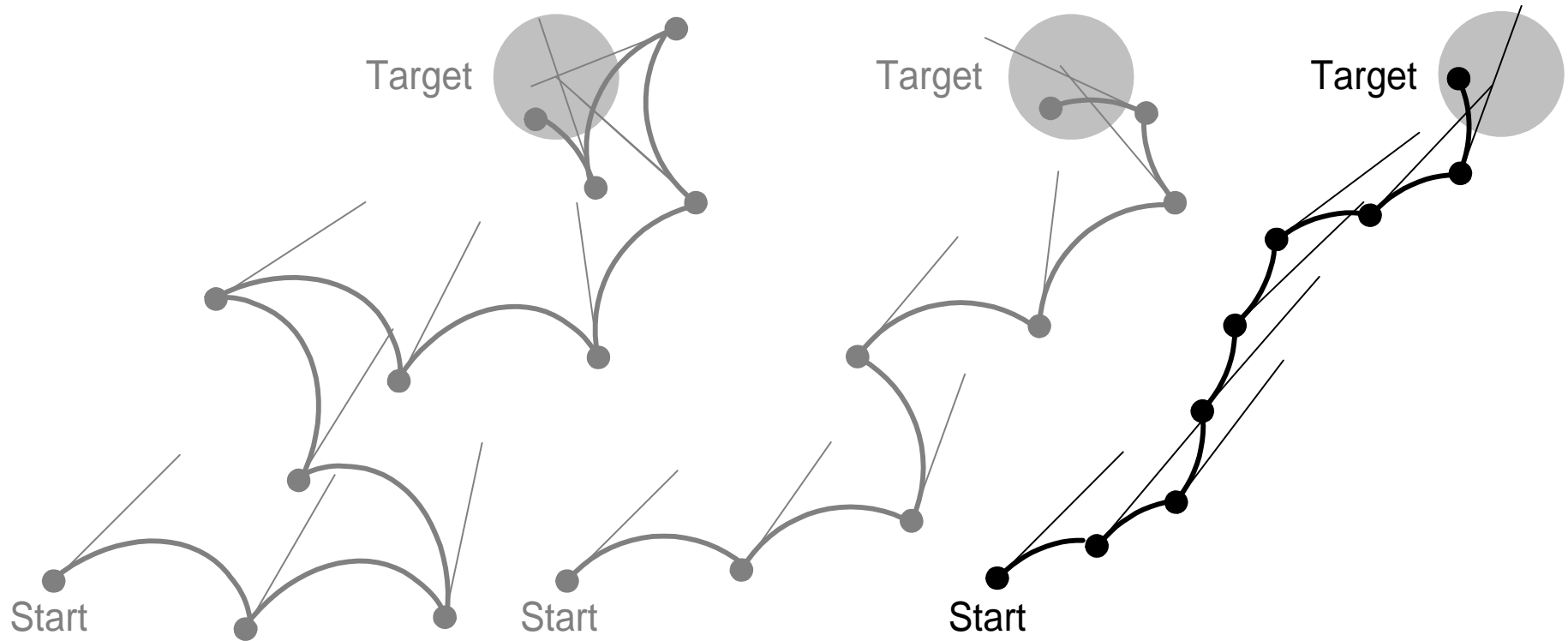
# Feedback (3)

stepsize:  
elapsed time

3 months  
25 months

2 months  
12 months

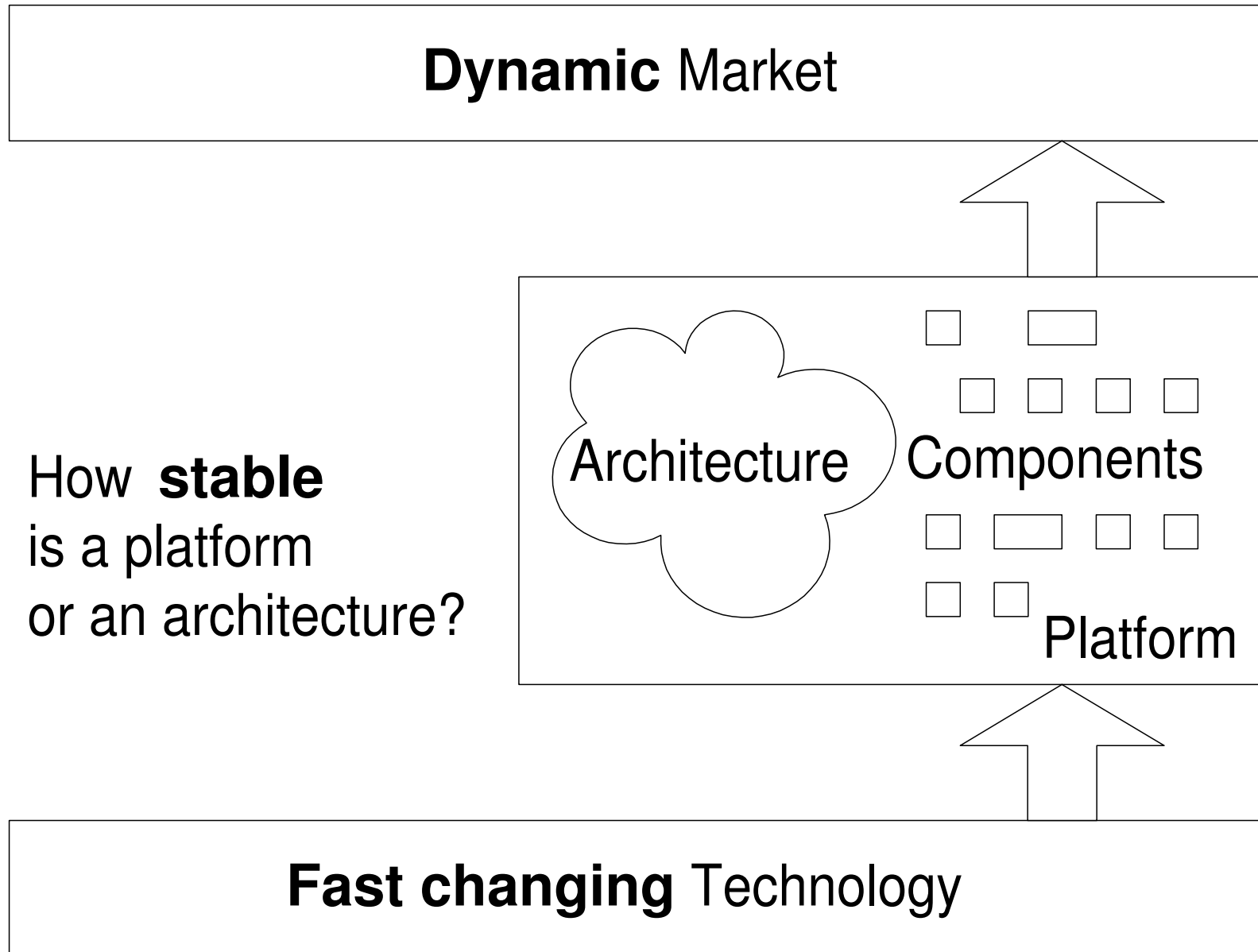
1 month  
8 months



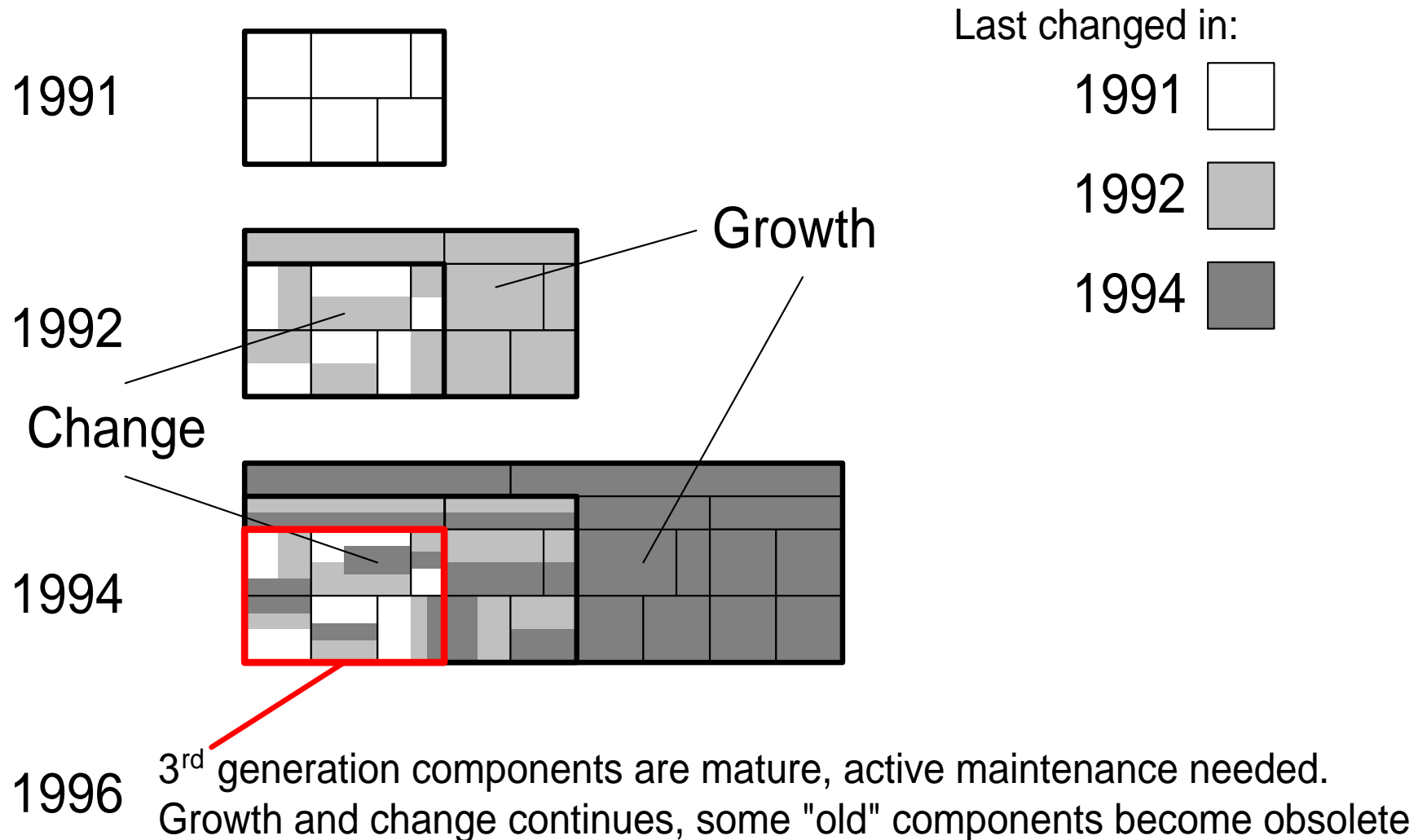
Small feedback cycles result in Faster Time to Market

## Awareness of dynamics

# Myth: Platforms are Stable

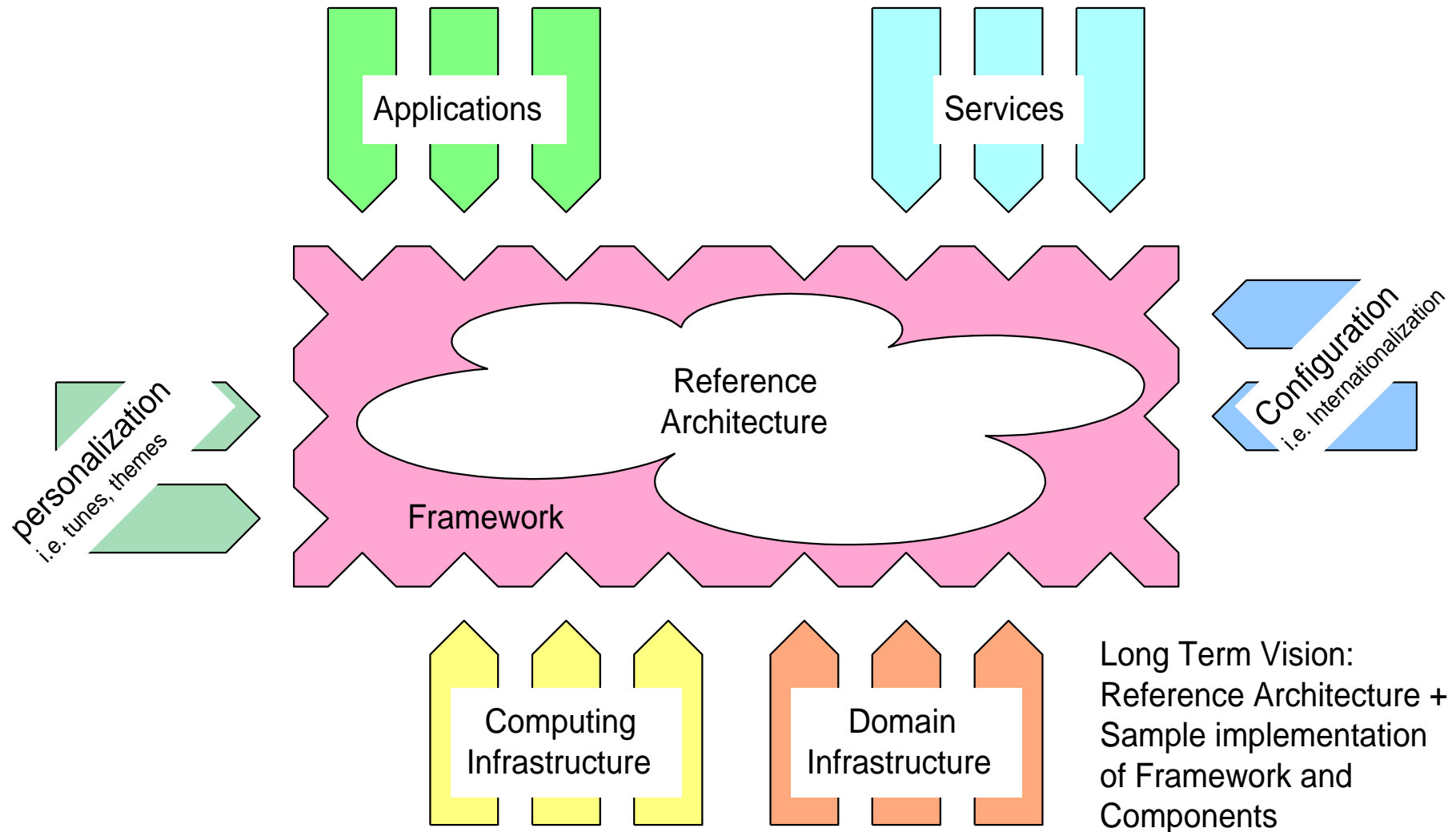


# Platform Evolution (Easyvision 1991-1996)

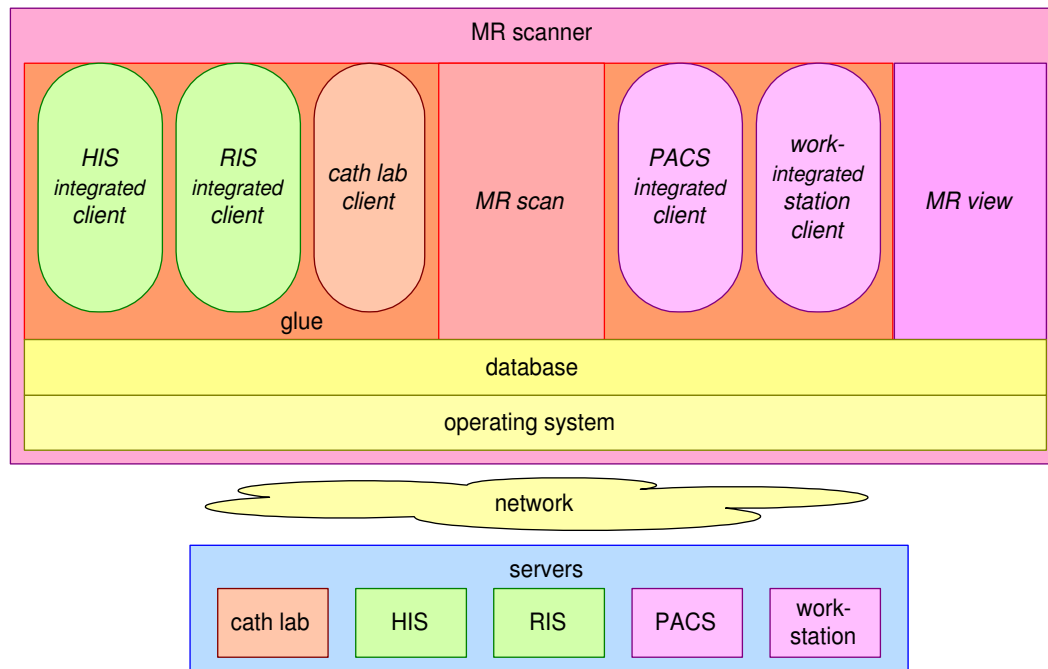


## Long Term Vision

# Example Long Term Vision

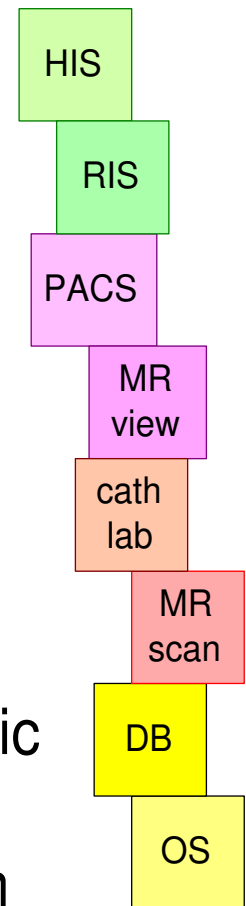


# Don't do



Proclaimed  
reuse

Opportunistic  
Legacy  
Integration



# Conclusion: Refactoring the Architecture is a must

