

From Legacy to State-of-the-art; Architectural Refactoring

by *Gerrit Muller* University of South-Eastern Norway-NISE

e-mail: `gaudisite@gmail.com`

`www.gaudisite.nl`

Abstract

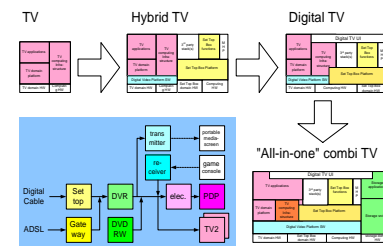
The market of electronic appliances shows a fast increasing diversity. Manufacturers must be able to combine existing functions and new applications in a short time frame. A large amount of accumulated SW code (legacy) has to be reused in new ways.

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.

September 9, 2018
status: finished
version: 1.3



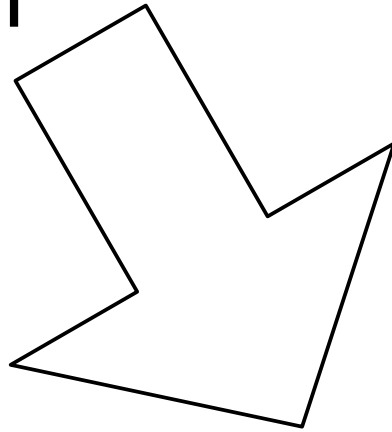
Today's Audio Video Consumer Products

From: COPA tutorial, Rob van Ommering

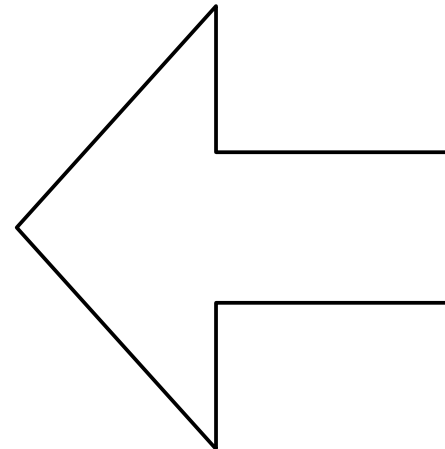
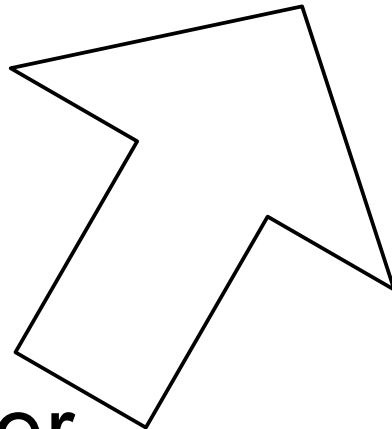


Trend: Convergence of separate worlds

Telecom



Consumer



Computer

Integration and Diversity



GSM phone



firewall



dvd



audio
microset



pda



watch



sailboat



surveillance
camera



cable
modem



set top box



headphone



pen



garment



car



camera



speech



mp3



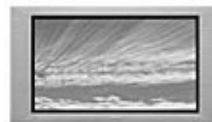
television



computer



games



flat display



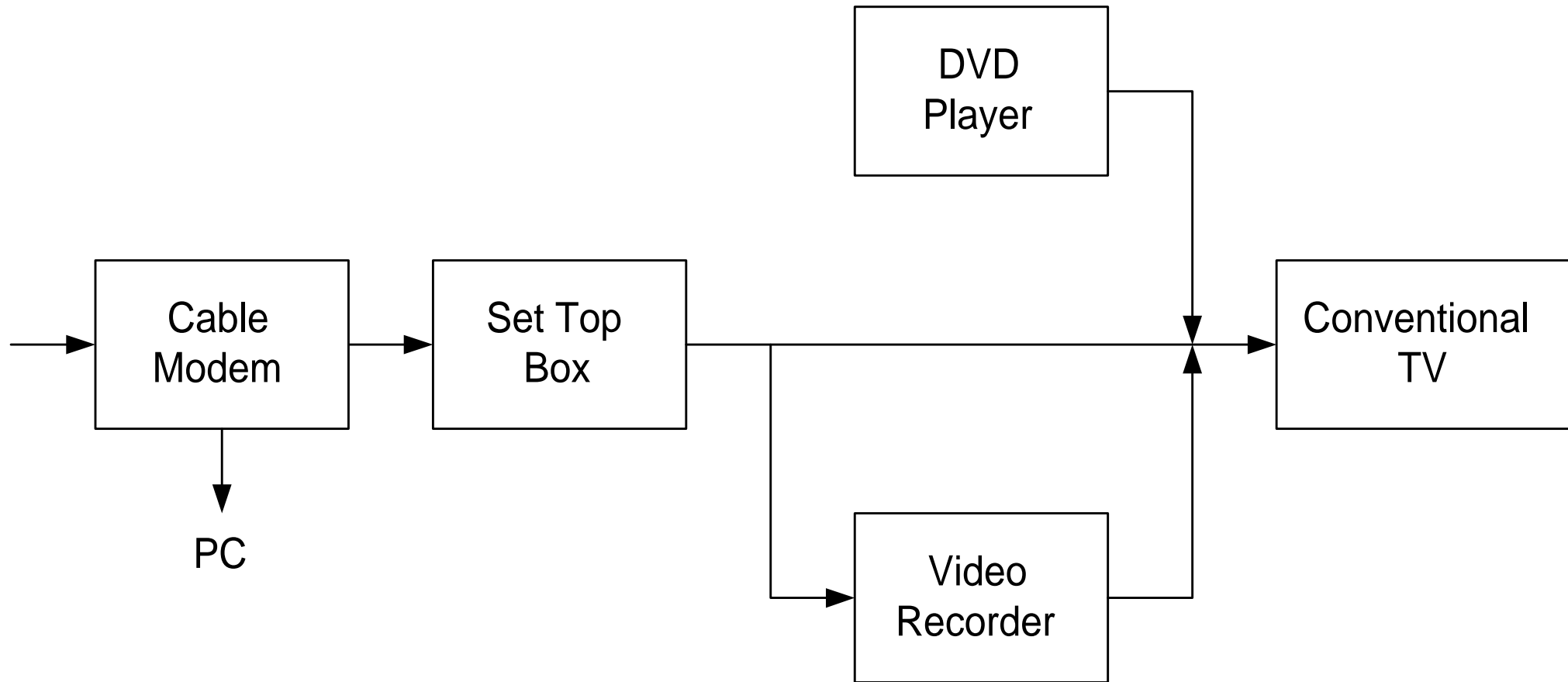
Communicator



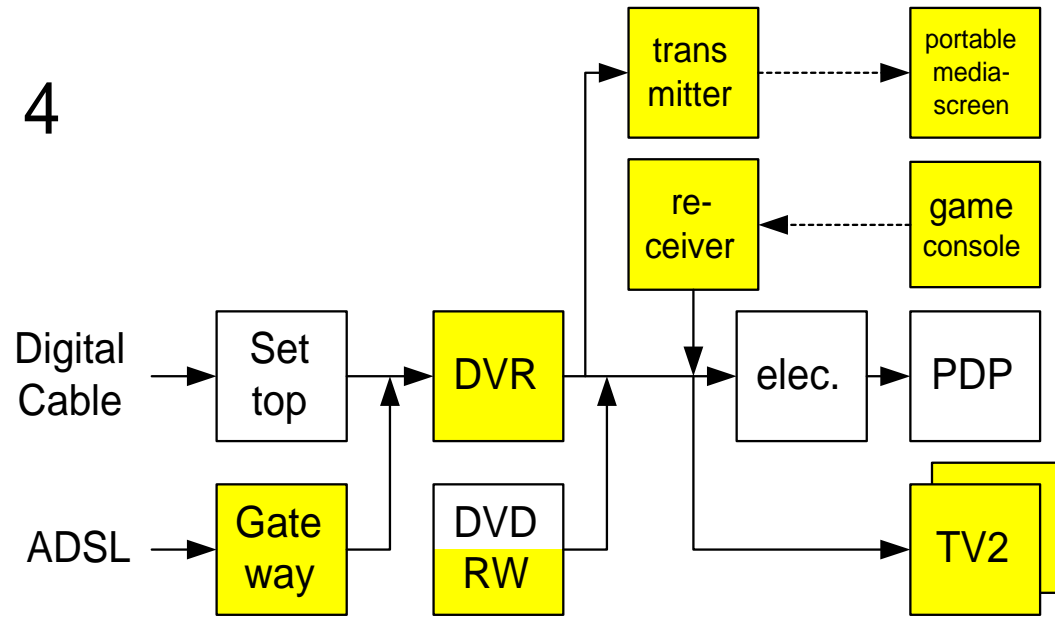
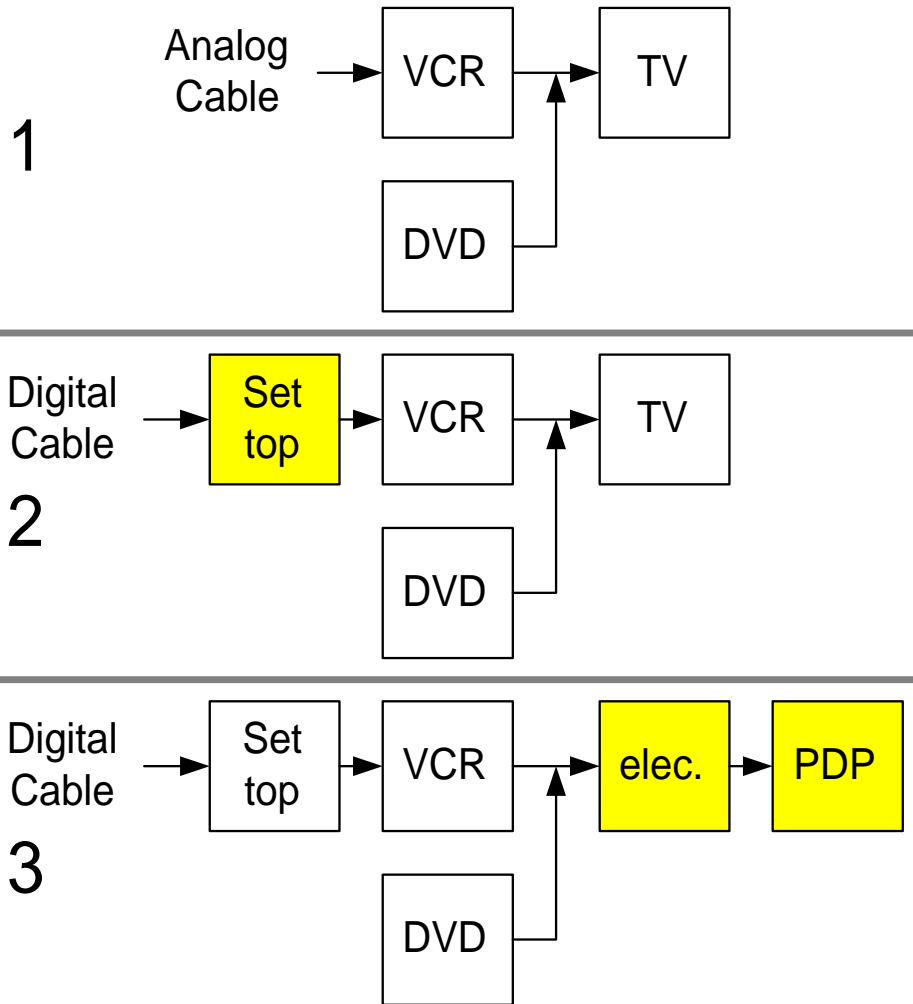
Ambient Intelligence
living room

car navigation

Today's Video Products



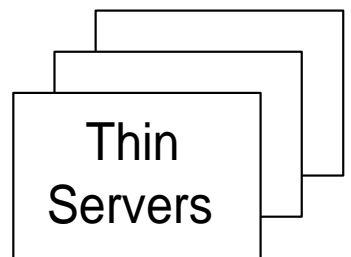
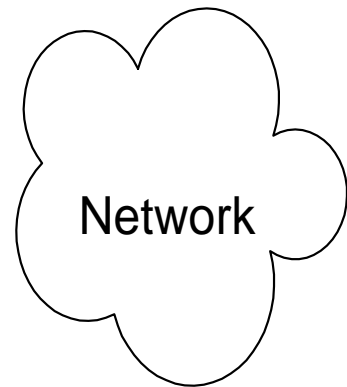
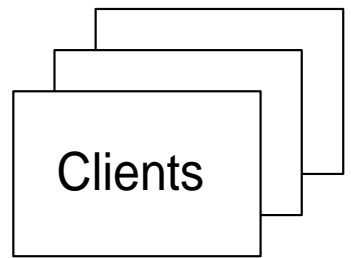
Evolution of Video Products



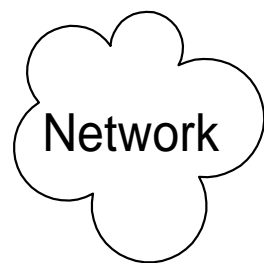
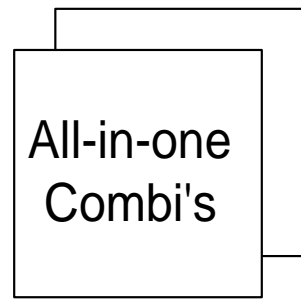
Multiple inputs
Multiple stores
Multiple displays
Multiple applications
Multiple formfactors

Distribution Scenario's

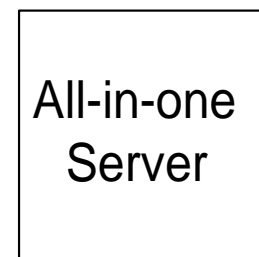
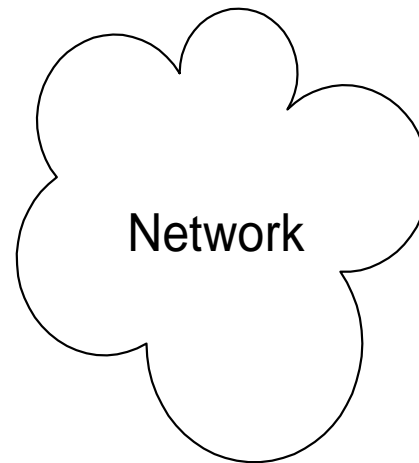
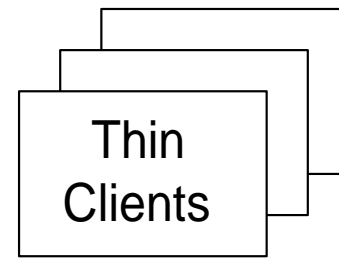
A "Thin Servers"



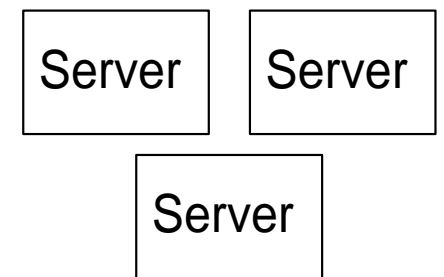
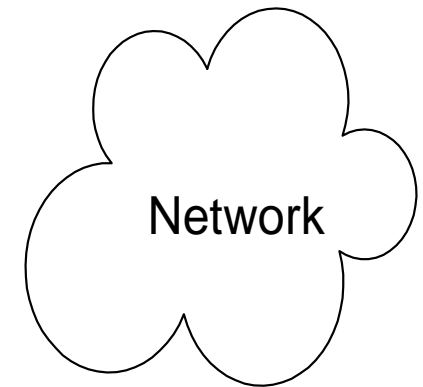
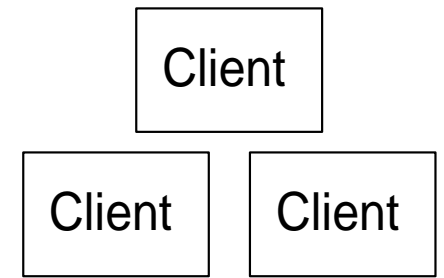
B "All-in-one" Combi's



C "All-in-one" Home server

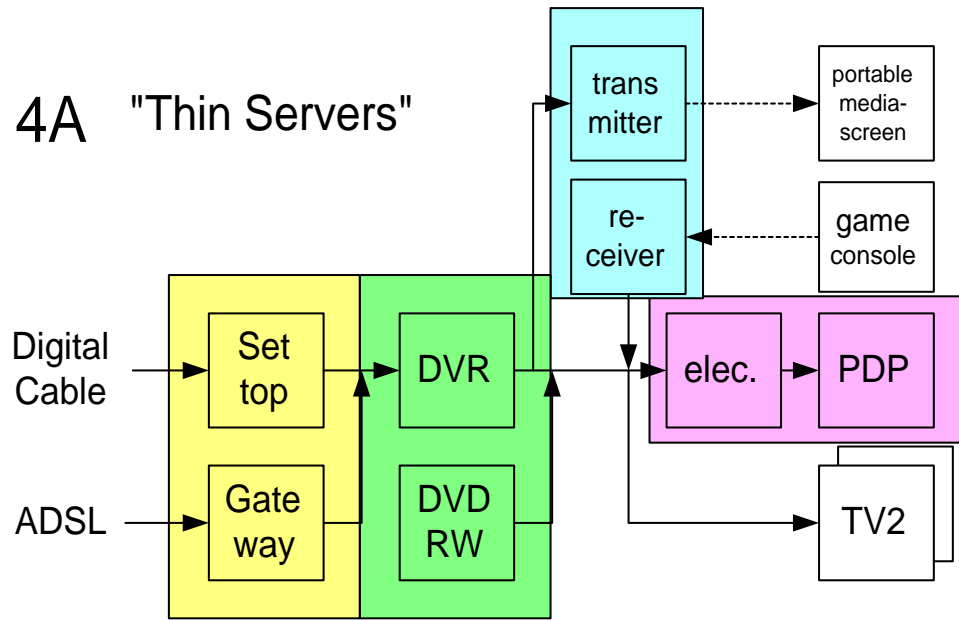


D "Modular" architecture

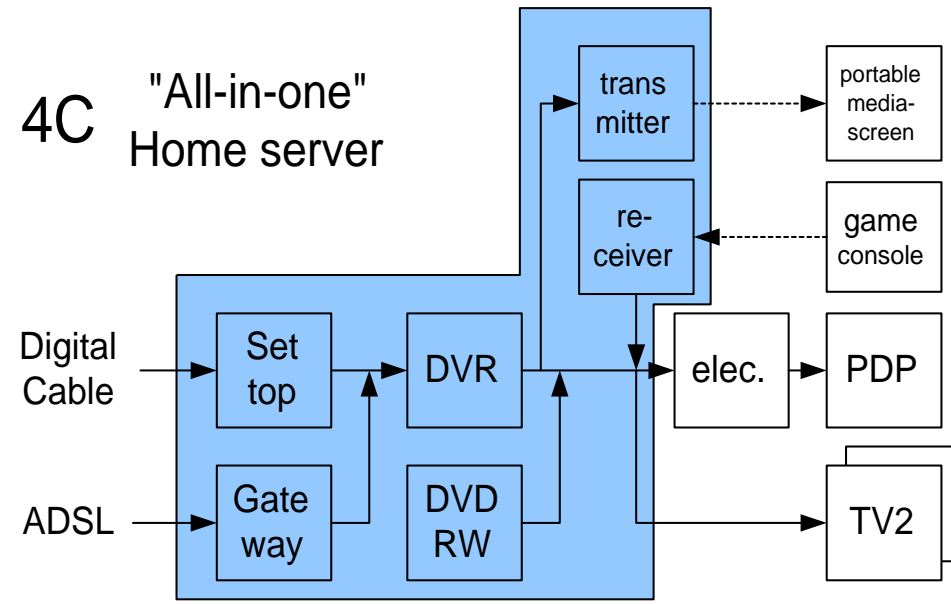


Product Packaging Options

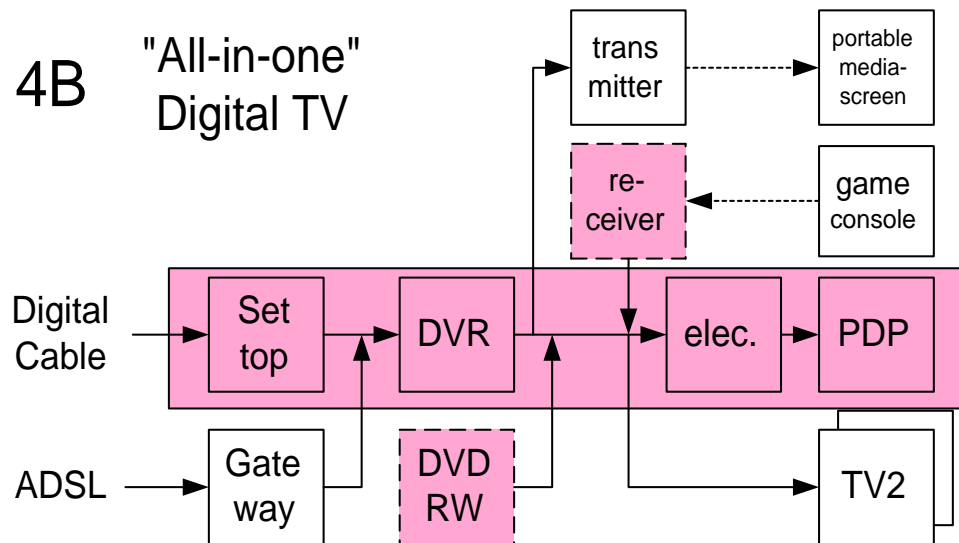
4A "Thin Servers"



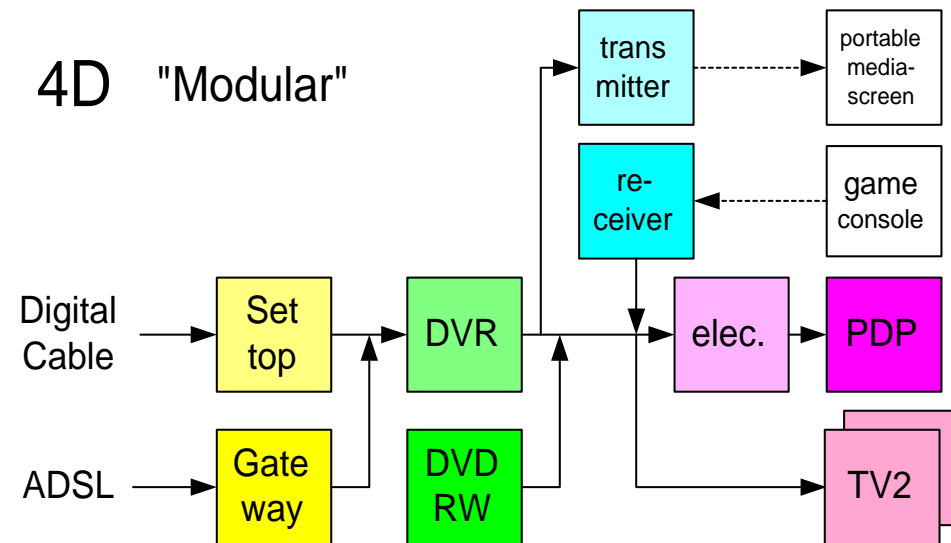
4C "All-in-one" Home server



4B "All-in-one" Digital TV



4D "Modular"

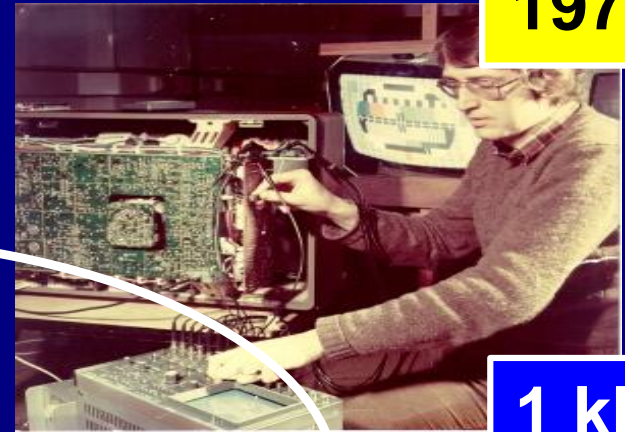


Moore's law

1965



1979



1 kB

Moore's law

1990



64 kB

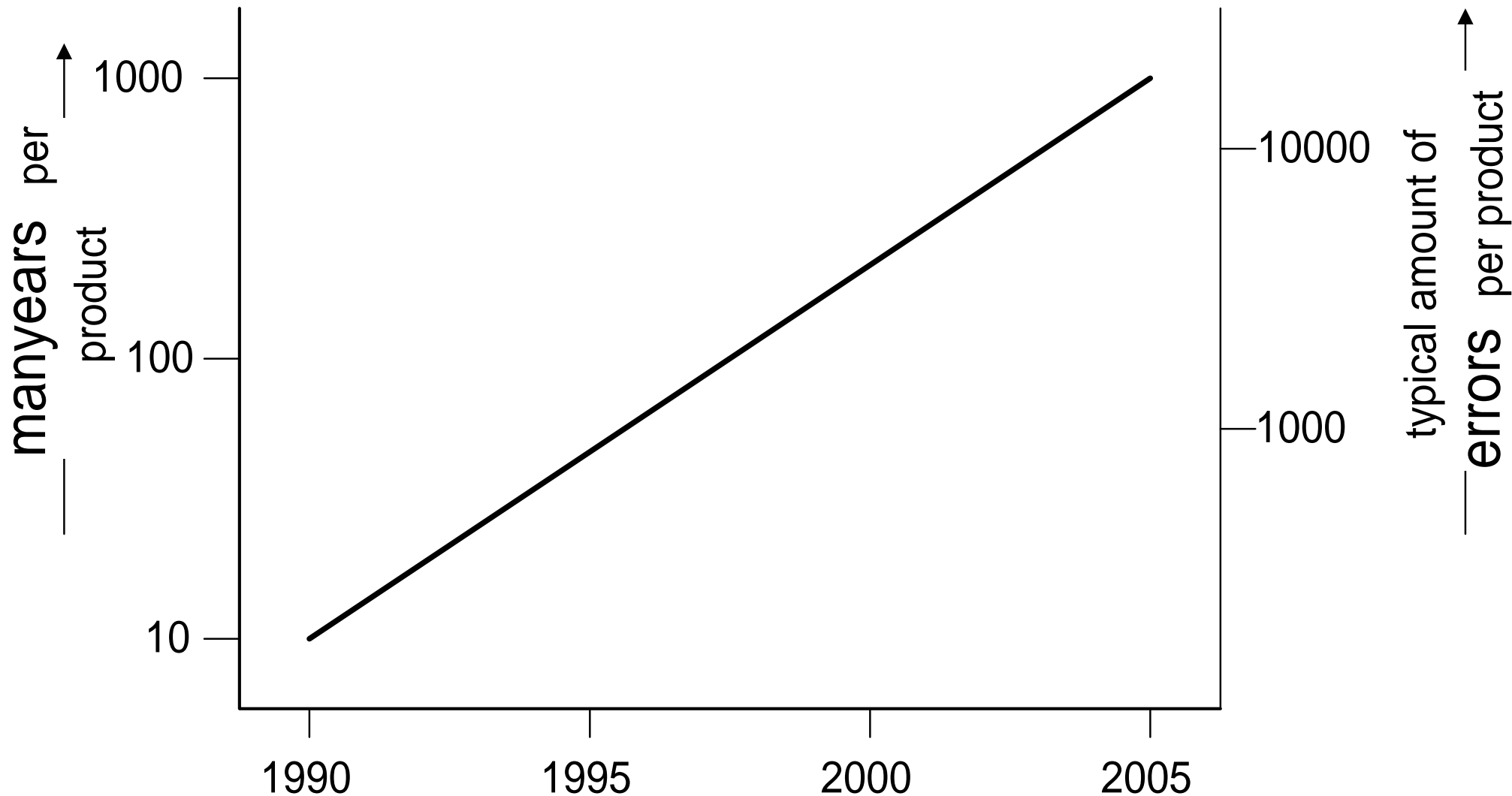
2000



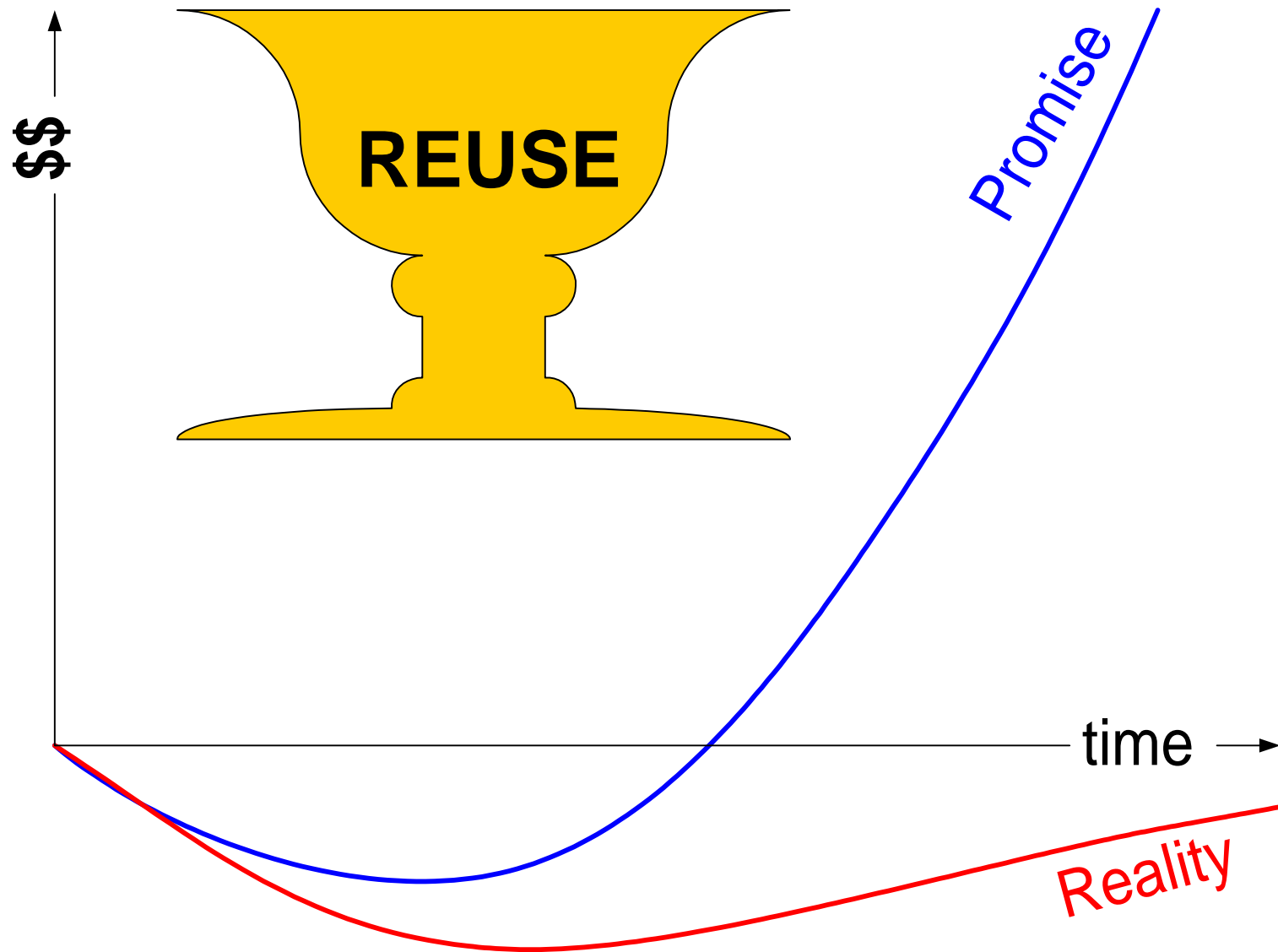
2 MB

From: COPA tutorial, Rob van Ommering

Problem: increasing SW size, decreasing reliability?

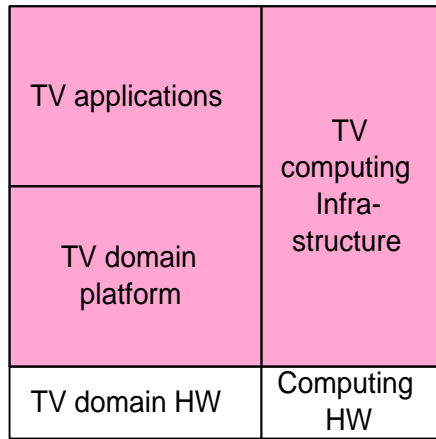


The Holy Grail: Reuse

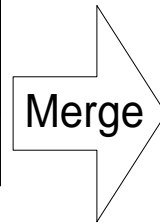
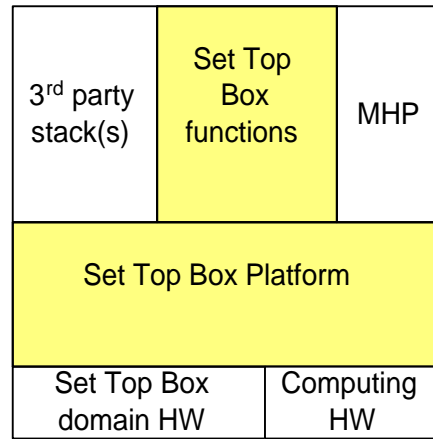


Simplistic Architecting: Digital TV

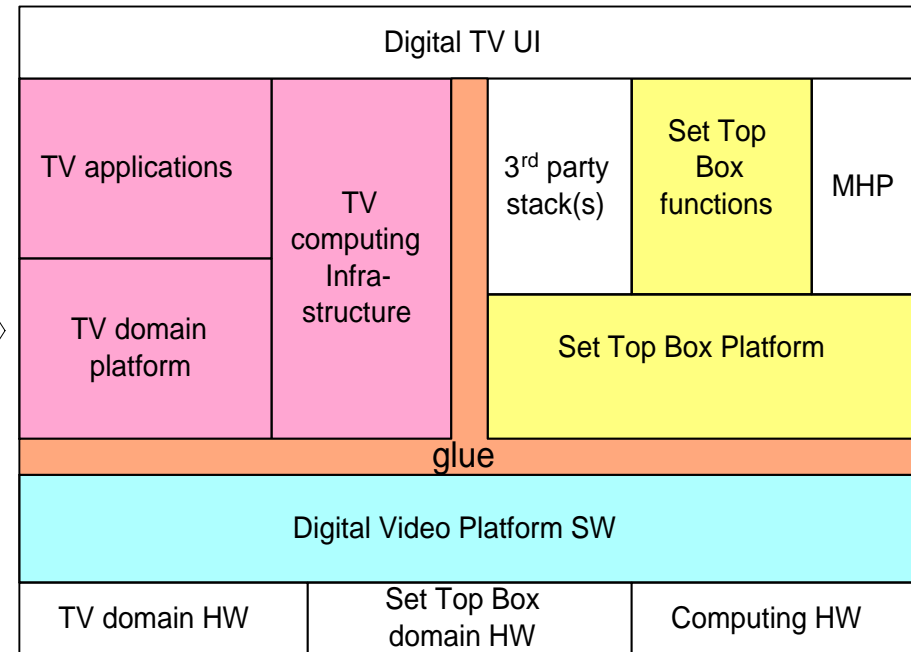
analog TV



Set top box

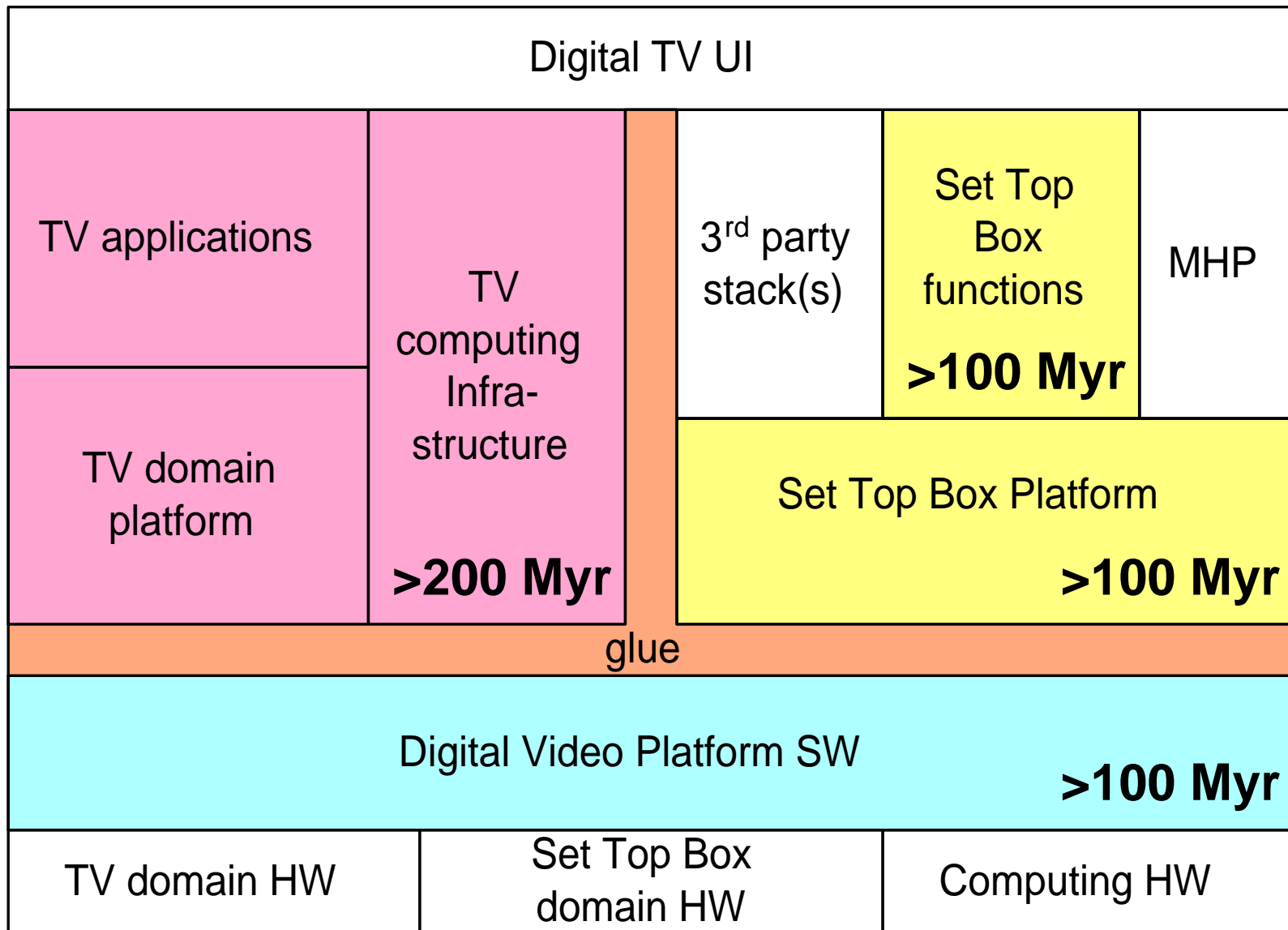


Digital TV



Digital Video Platform

Available Code Assets



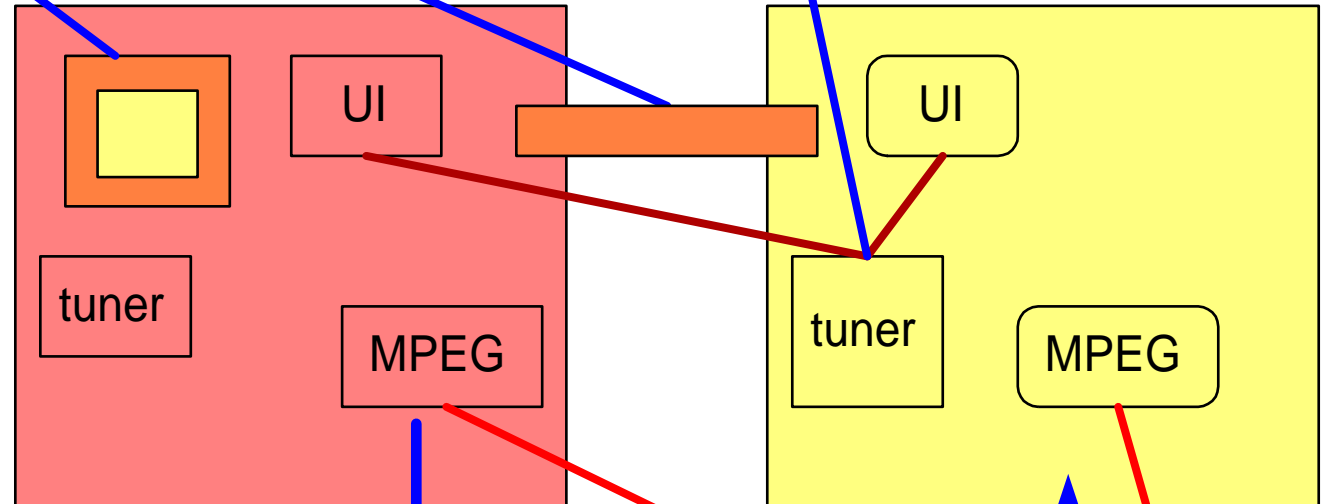
"Legacy" code > 500 Myr

Merge problems

Architectural mismatch :

wrappers, translators, conflicting controls

additional code
and complexity,
no added value

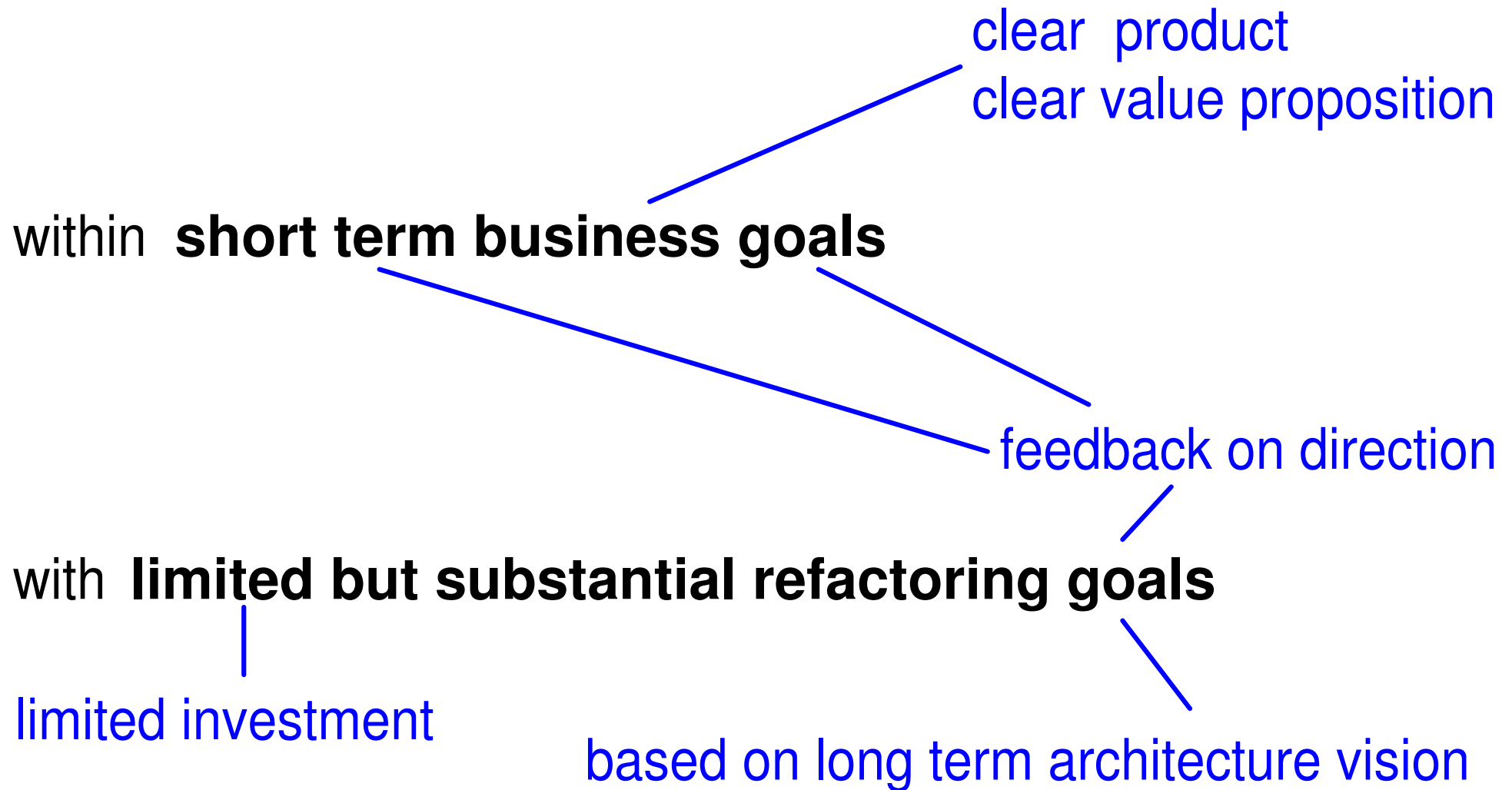


Poor performance;
additional resource usage

Duplication

Problems ← Architecture — Reuse → non problem

Refactoring



Example of Refactoring Goals

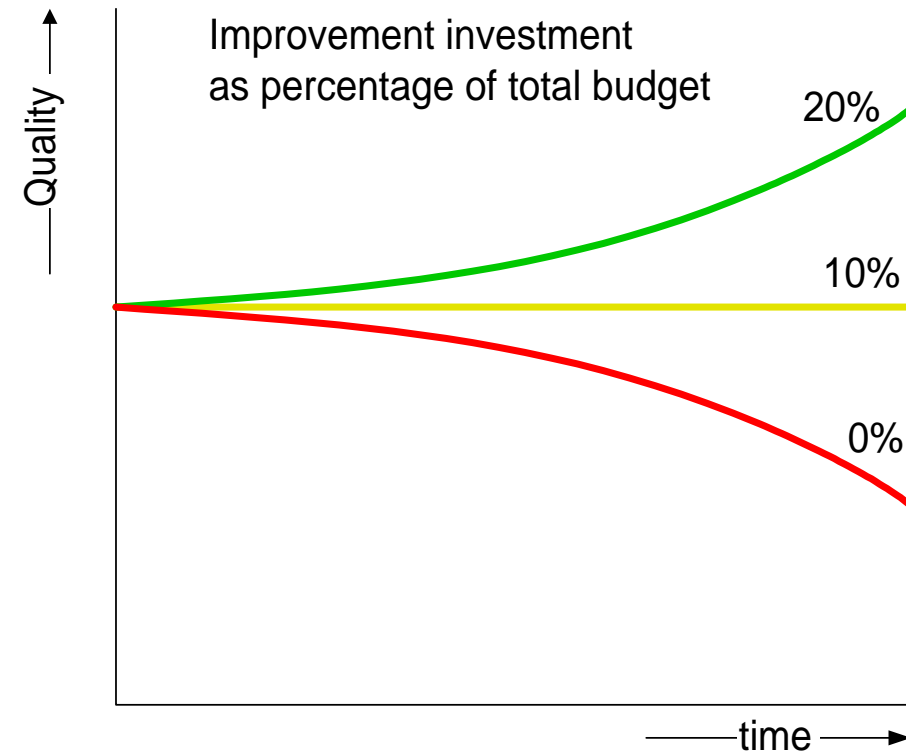
+ Decrease Code Size

+ Decrease Resource Usage

- * power
- * memory
- * silicon area

+ Increase Performance

- * response time
- * throughput



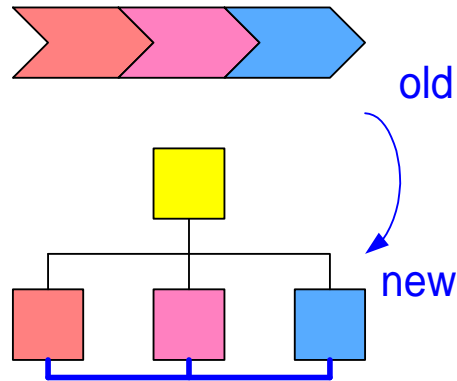
+ Increase quality

- * decrease fault density

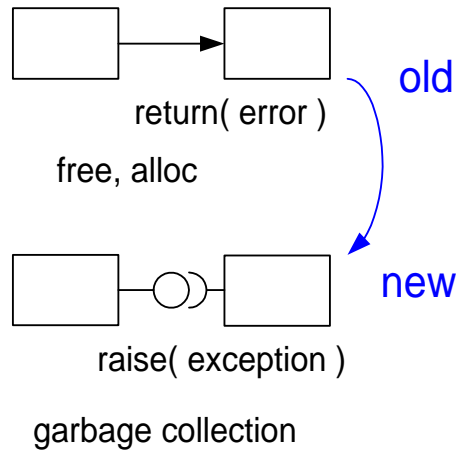
Architectural vs Code refactoring

Architectural Refactoring

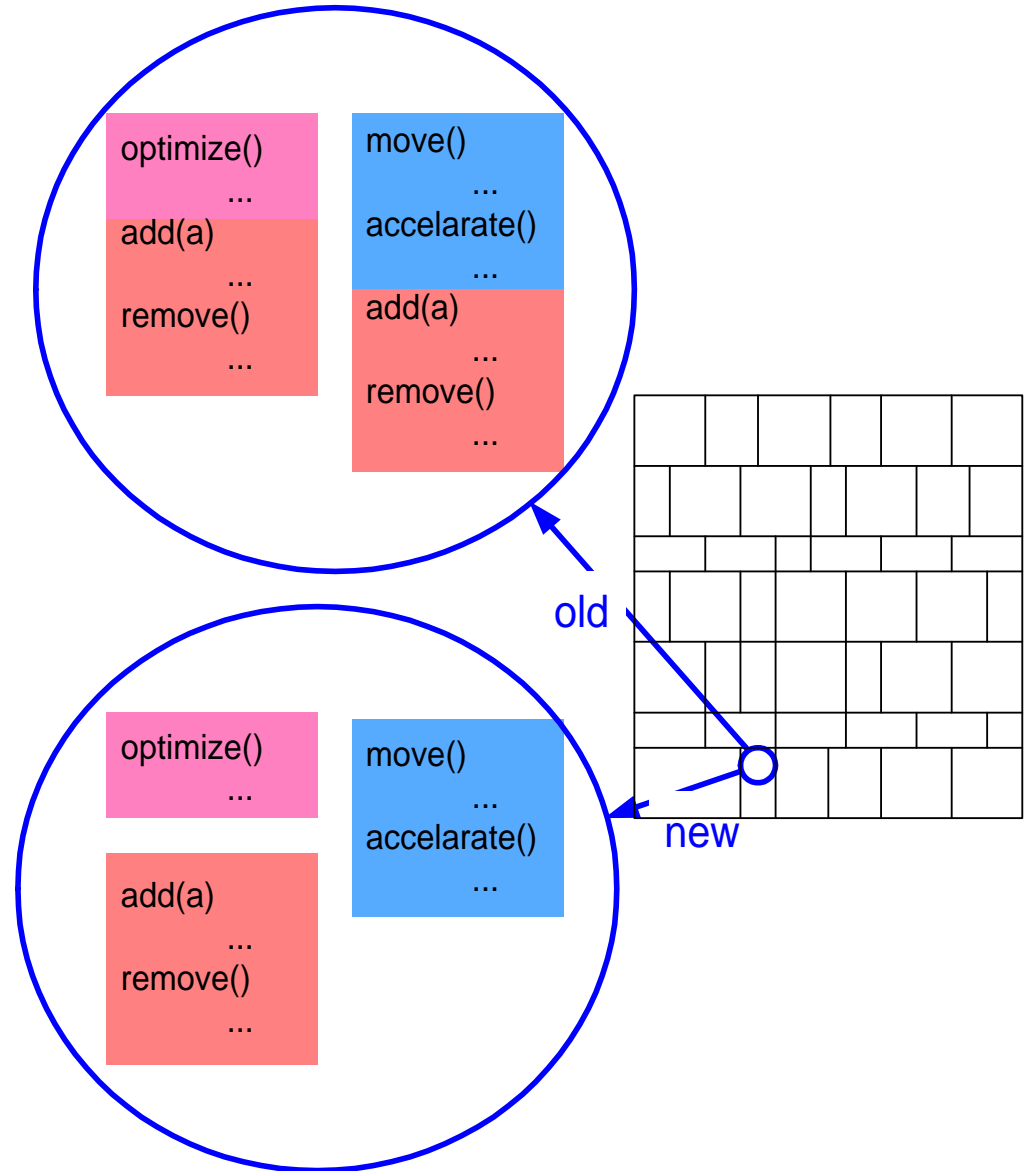
Function, Structure, Rationale



Mechanisms, Technologies



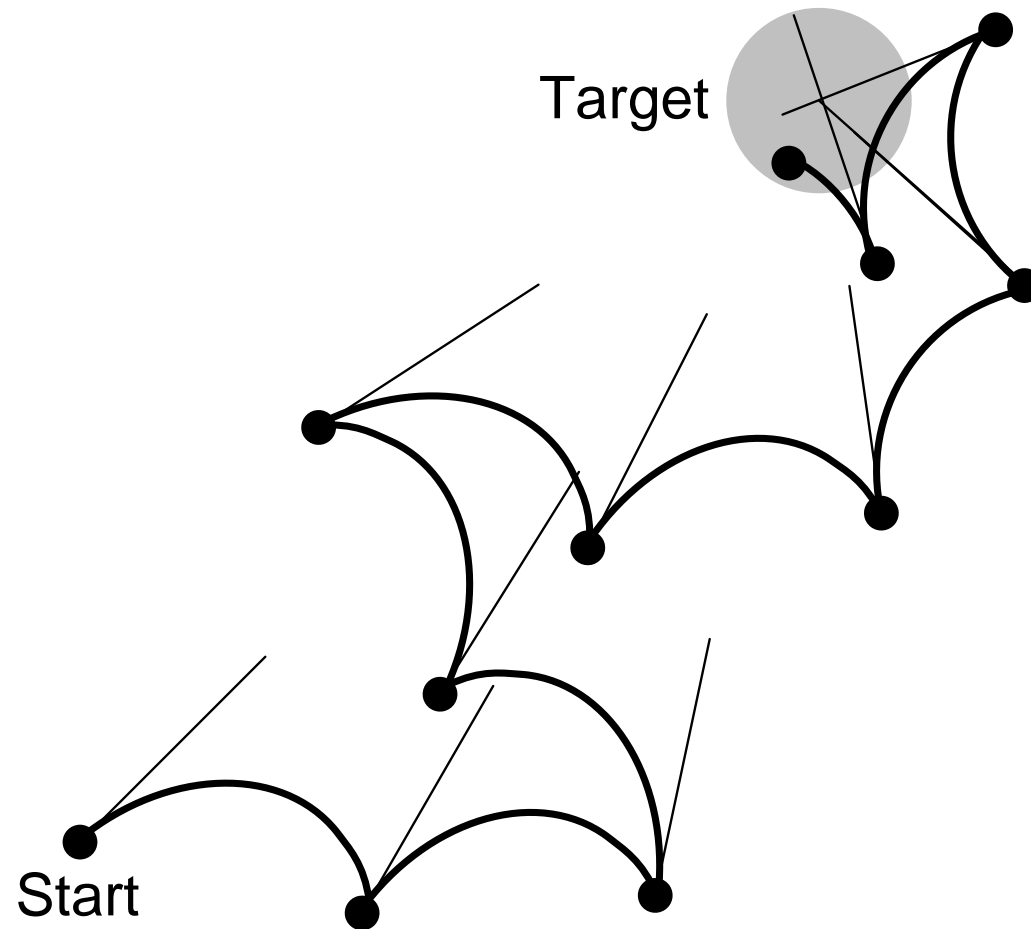
Code Refactoring



Frequent feedback

Feedback

stepsize: 3 months
elapsed time: 25 months

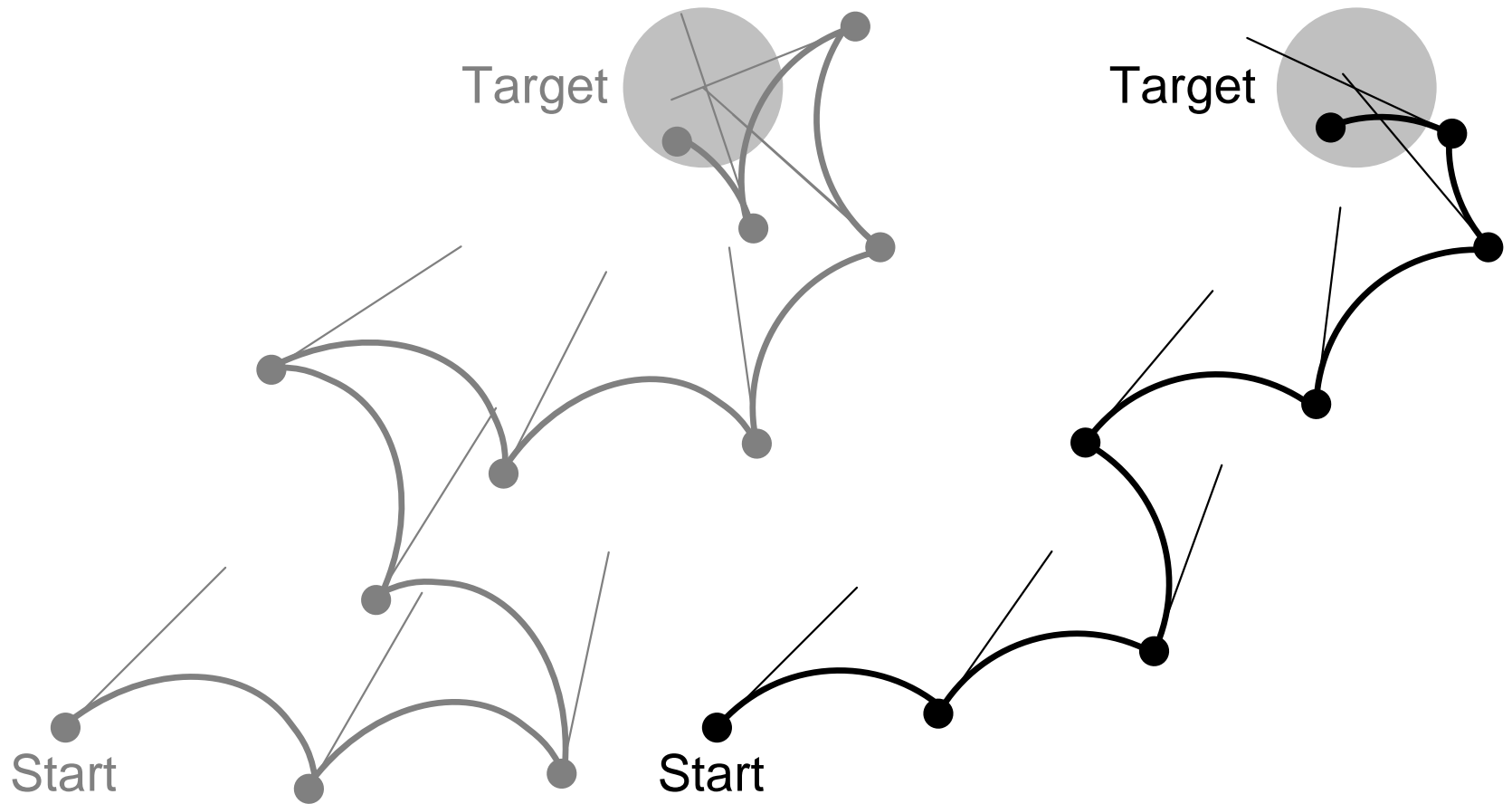


Feedback (2)

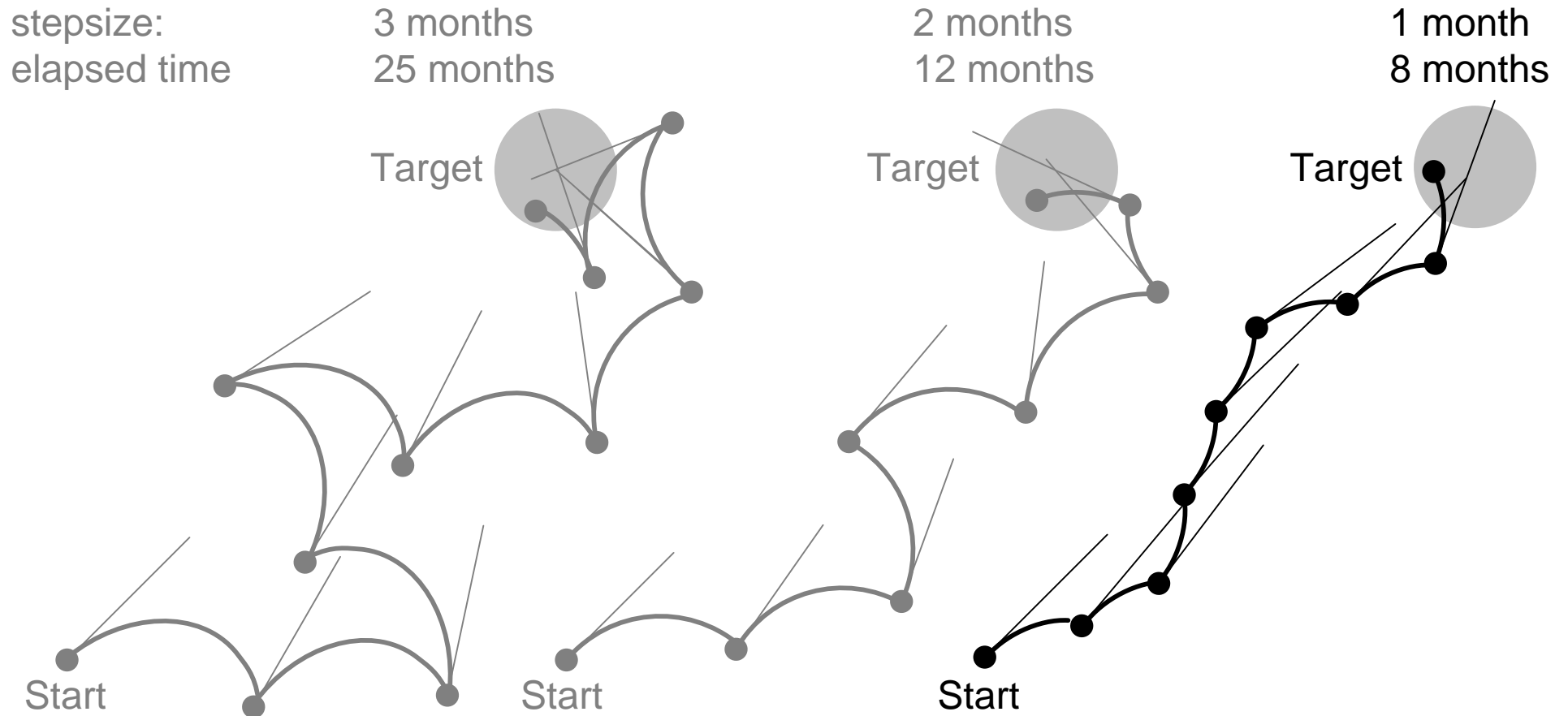
stepsize:
elapsed time

3 months
25 months

2 months
12 months



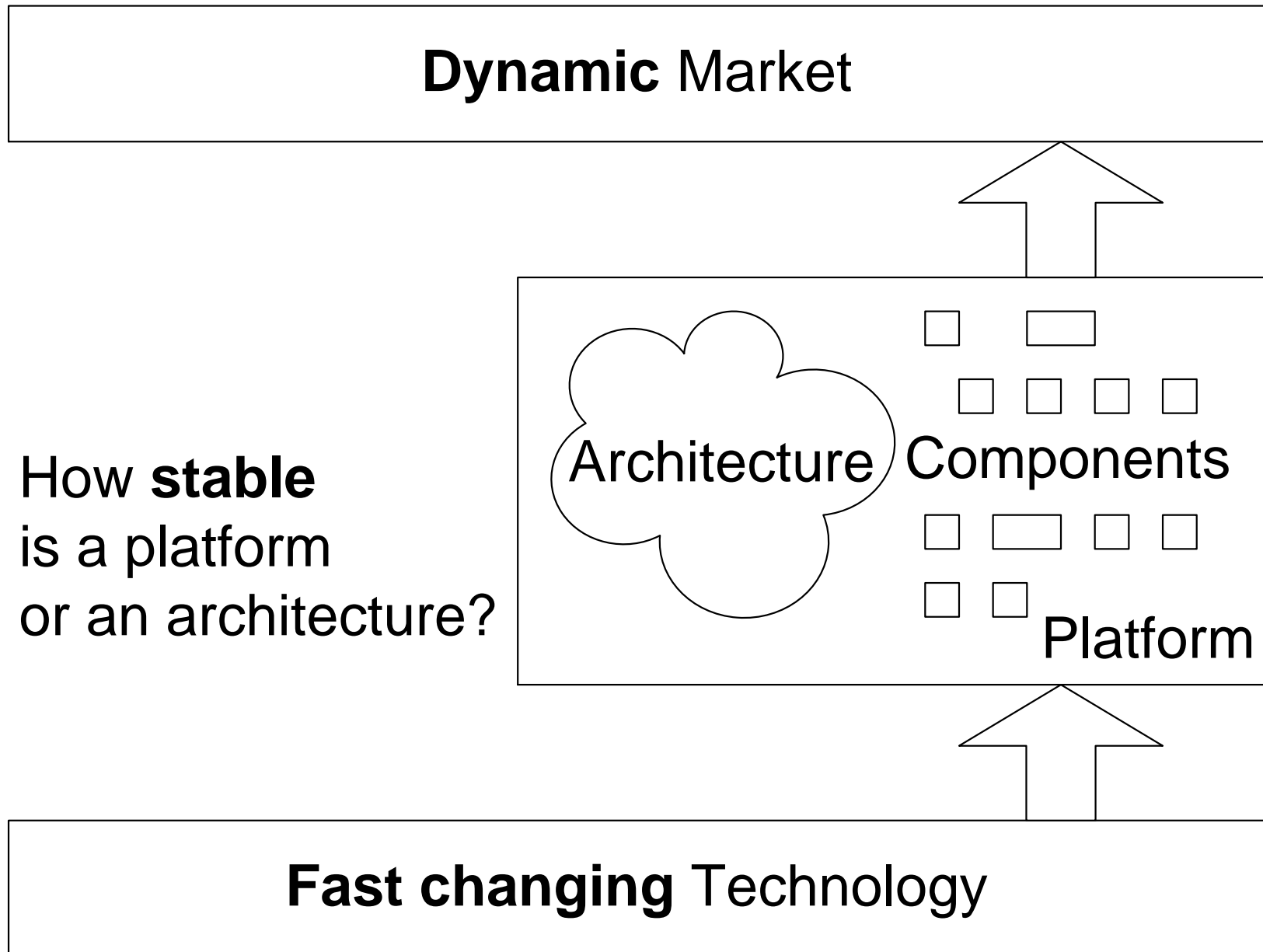
Feedback (3)



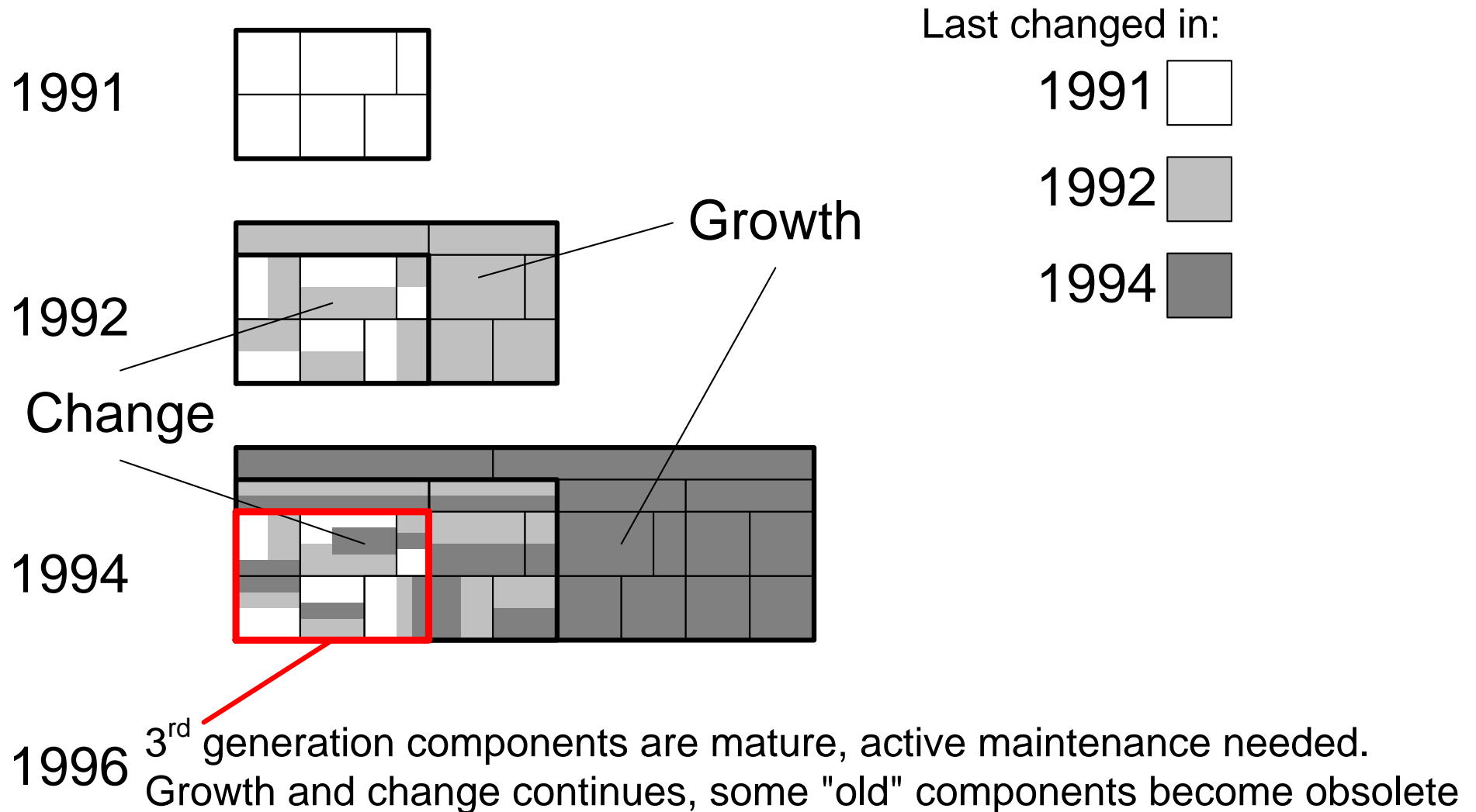
Small feedback cycles result in Faster Time to Market

Awareness of dynamics

Myth: Platforms are Stable

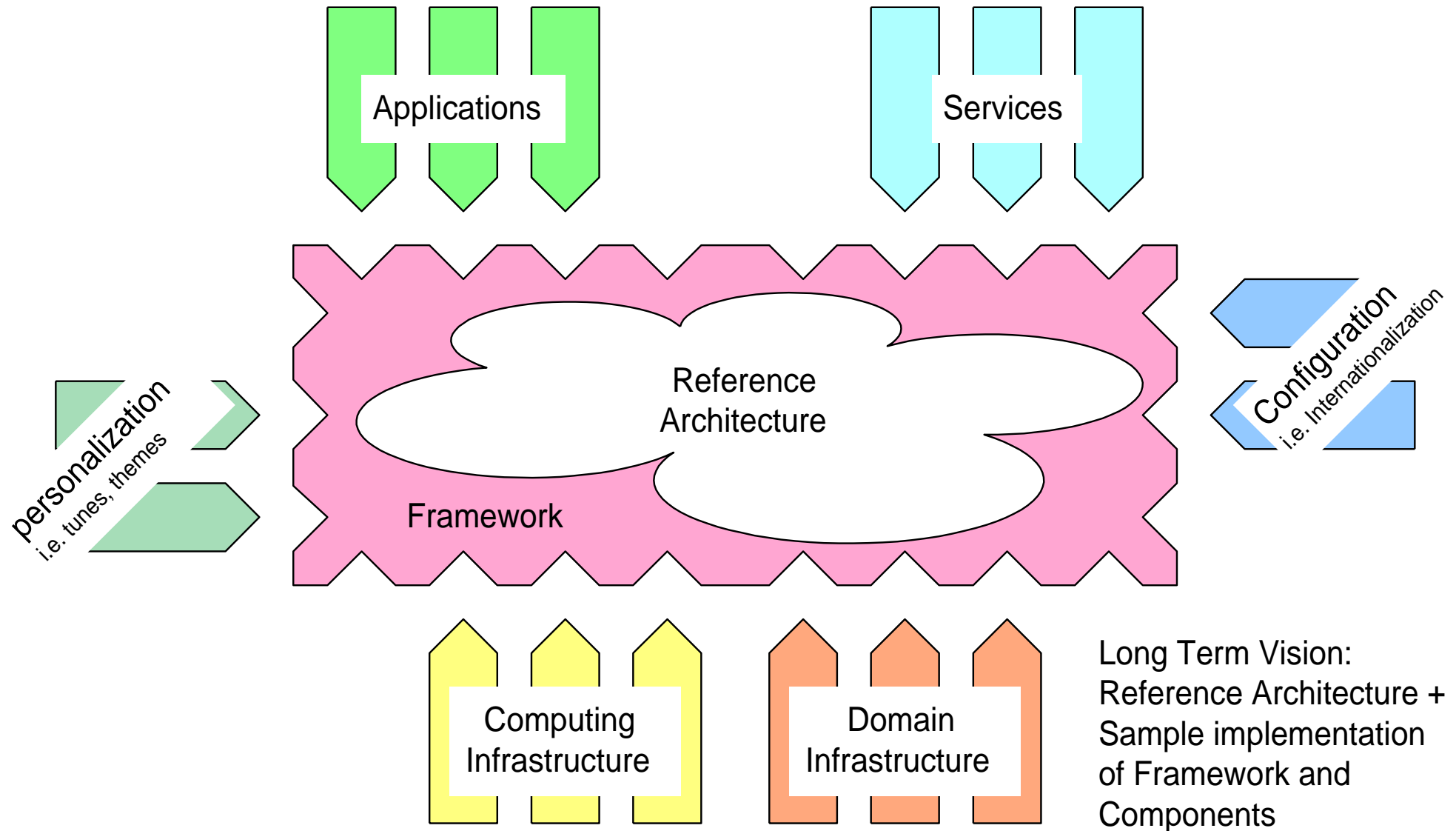


Platform Evolution (Easyvision 1991-1996)

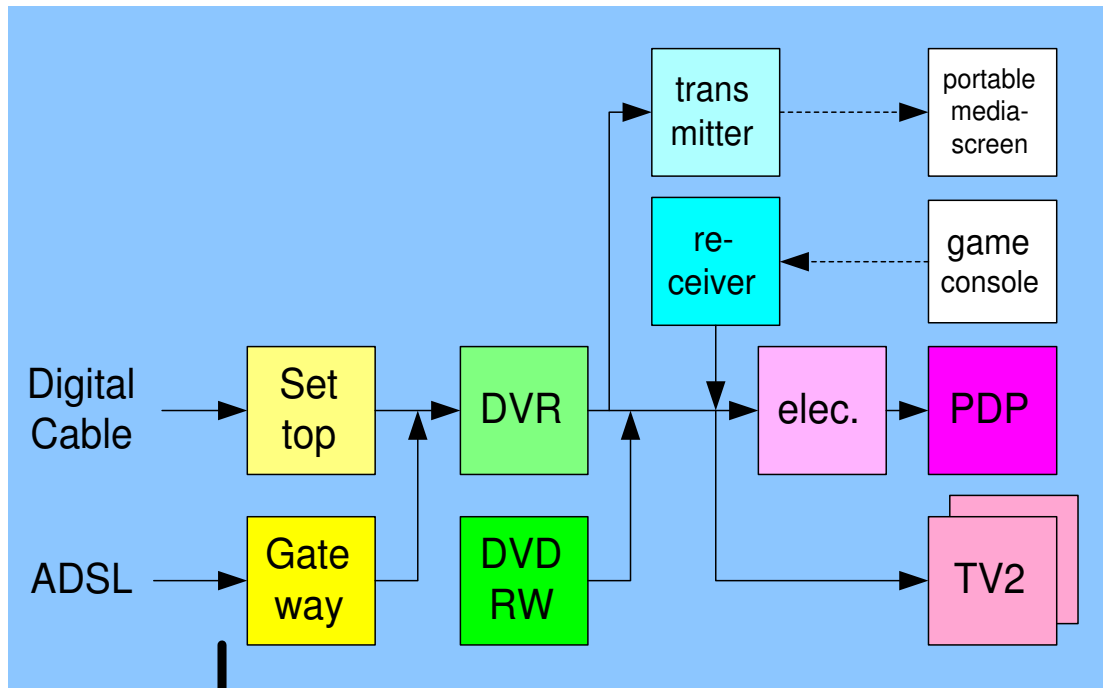


Long Term Vision

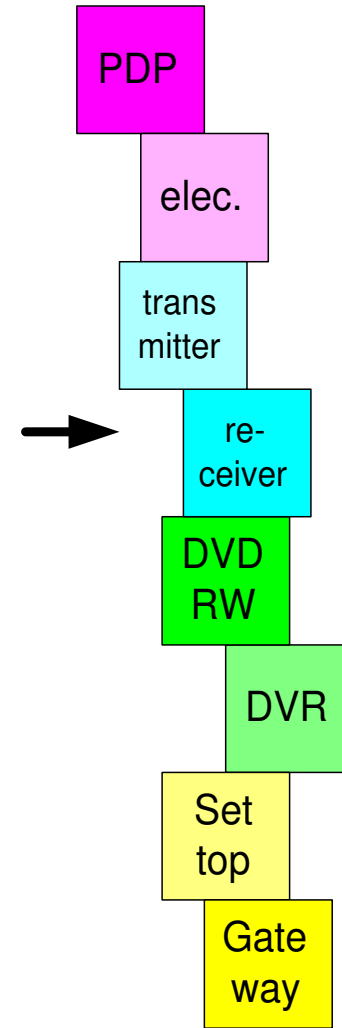
Example Long Term Vision



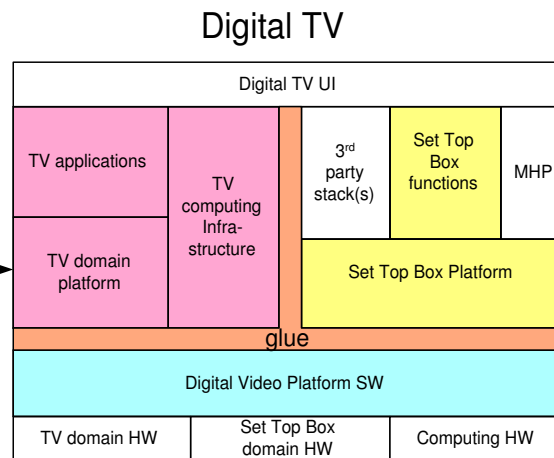
Don't do



Opportunistic
Legacy
Integration

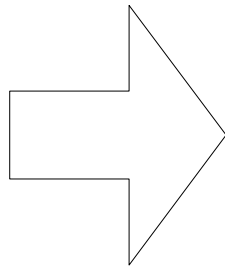
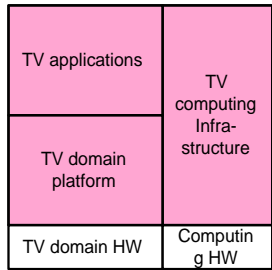


Proclaimed
reuse

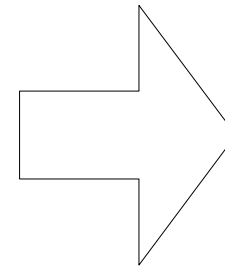
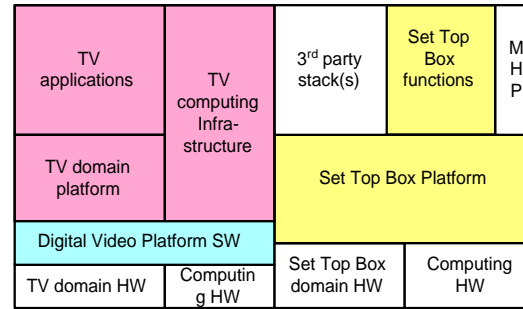


Conclusion: Refactoring the Architecture is a must

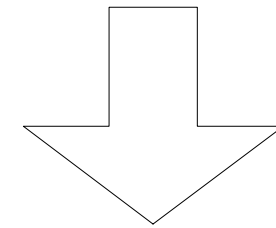
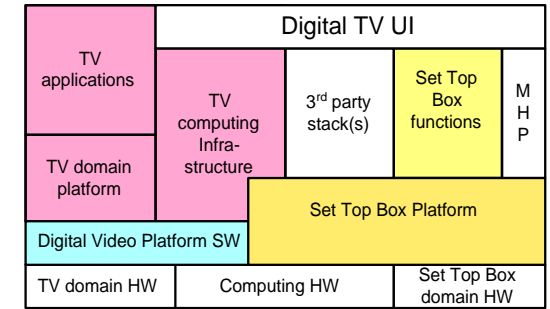
TV



Hybrid TV



Digital TV



"All-in-one" combi TV

