Emerging Architecture

Olve Maudal, Cisco Systems Norway

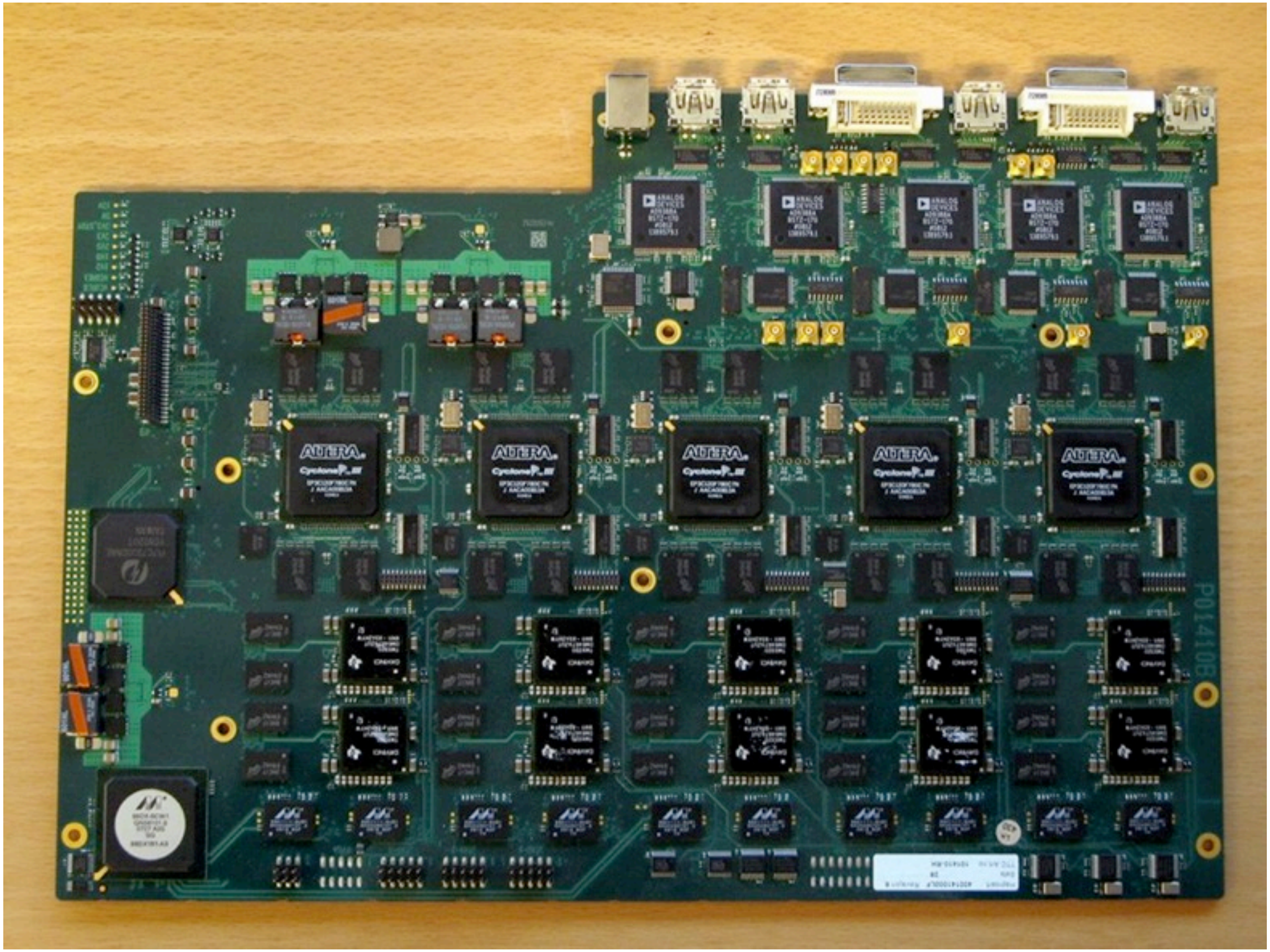
Det finnes mange teoretiske tilnærminger til hvordan god programvarearkitektur skal se ut. Men det viser seg at mange, kanskje de fleste, vellykkede kodebaser er et resultat av tilsynelatende tilfeldige valg og ustrukturert implementasjon. I denne lyntalen skal jeg diskutere litt om forutsetningene for å utvikle en solid kodebase og hva god programvarearkitektur *egentlig* handler om.

A 15 minute lightning talk
dagen@IFI, October 27, 2011

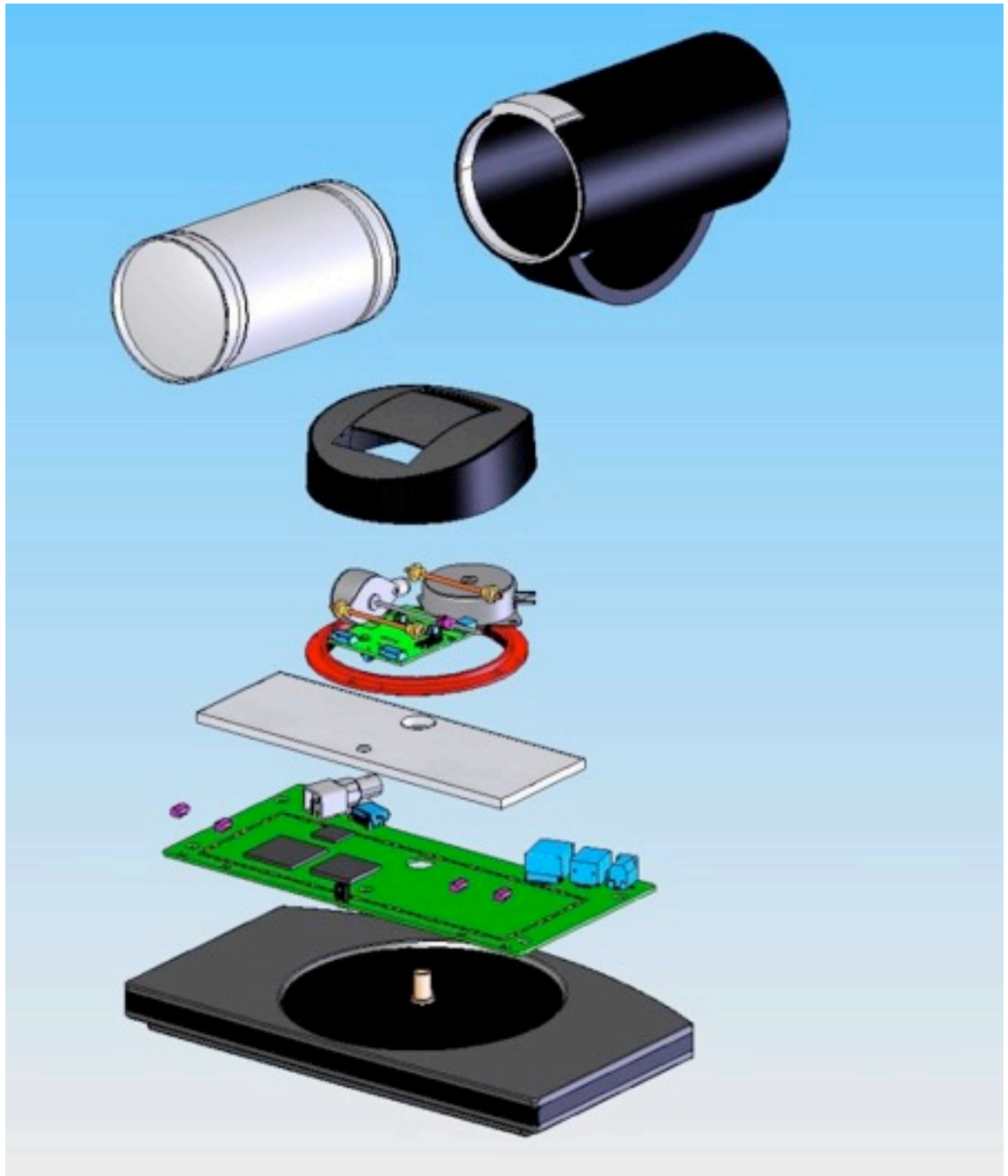
TelePresence Technology Group
Cisco Systems Norway

Telepresence systems and solutions

Electronics / Hardware



Mechanics



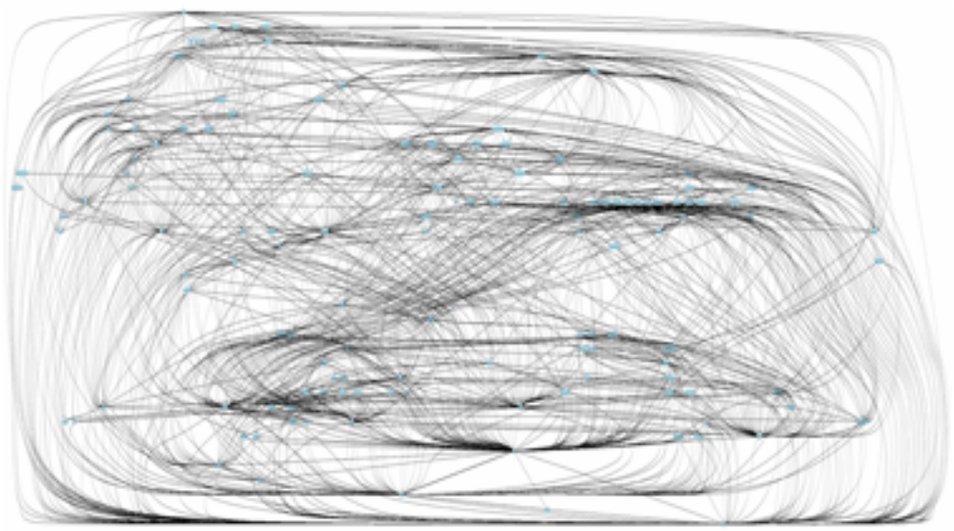
Industrial Design

1993

2011

Software Development

The Codebase



- C and C++
- a few million lines of code
- processor agnostic (TI, Phillips, PPC, ARM, Intel, and more...)
- currently developed and maintained by ~200 developers
- typically 50-200 commits per day
- very visible traces back to '80s and '90s

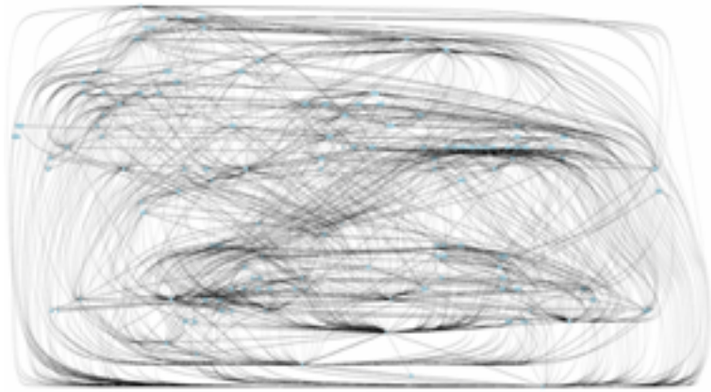
1987

1992

1997

2000

2003



2004

2006

2006

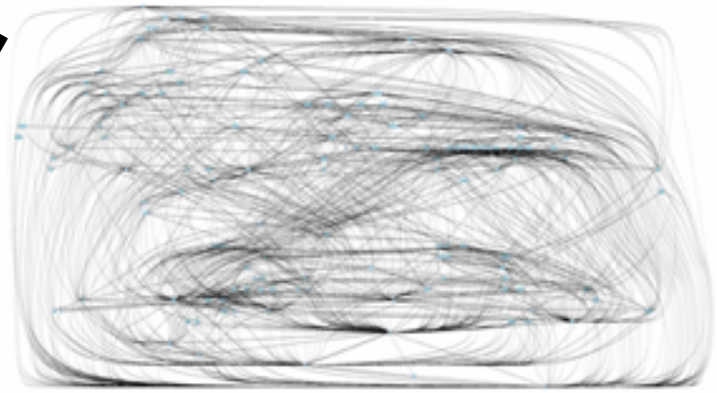
2005

2010

2008

2012

2009



2012

2008

2011

2009

- strict modularization

- strict modularization
- documented interfaces

- strict modularization
- documented interfaces
- well defined development processes

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications
- corporate coding standards

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications
- corporate coding standards
- change management

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications
- corporate coding standards
- change management
- code is documented properly

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications
- corporate coding standards
- change management
- code is documented properly
- strong and visible architecture

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications
- corporate coding standards
- change management
- code is documented properly
- strong and visible architecture

There are only two kinds of codebases: the ones people complain about and the ones nobody cares about anymore...

(inspired by a similar quote by Bjarne Stroustrup)

Unfortunately, by the time Rupert's mother had finished knitting his outfit, the war was over.

eFaktura

eFaktura

Analogies for software development

Analogies for software development

Analogies for software development

Analogies for software development

Analogies for software development


File
Speed
Options
Disasters
Windows
Newspaper
Oct 3575 <LAKELAND> $20,000
Centering Tool
SIGN
RCI

File Speed Options Disasters Windows Newspaper Sun 11:12:13

Dec 2048 <Volcano City> $20,000

SIGN

RCI

Query Tool
Water Shortage Reported

Typical characteristics of successful codebases:

- disposable code, modules and infrastructure
- low coupling and high cohesion
- duplication when needed
- transparent interfaces
- collective ownership
- rightsized modules and infrastructure
- focused on business needs
- working code *is* the documentation

This can evolve by:

- avoid *the* architect, rely on collective intelligence instead
- codebase should be fun to work with and easy to test
- continuous re-planning and re-working
- collective ownership
- build stuff only when needed
- reflect on status quo instead of dreaming
- share vision and communicate direction(not targets)
- beware of subliminal dependencies
- focus on what happens *between* modules
- establish multi-level feedback mechanisms
- prefer feature teams over component teams
- adapt your organization to support your architecture

Python's Flying Circus (page 40)

The Dark Ages of Software Development (mid 80's to mid 90')

- strict modularization
- documented interfaces
- well defined development processes
- detailed requirement specifications
- corporate coding standards
- change management
- code is documented properly
- strong and visible architecture

Problems in software development usually multiply and gets worse by exerting more control...

The more you tighten your grip, Tarkin, the more star systems will slip through your fingers.

(Princess Leia)

!