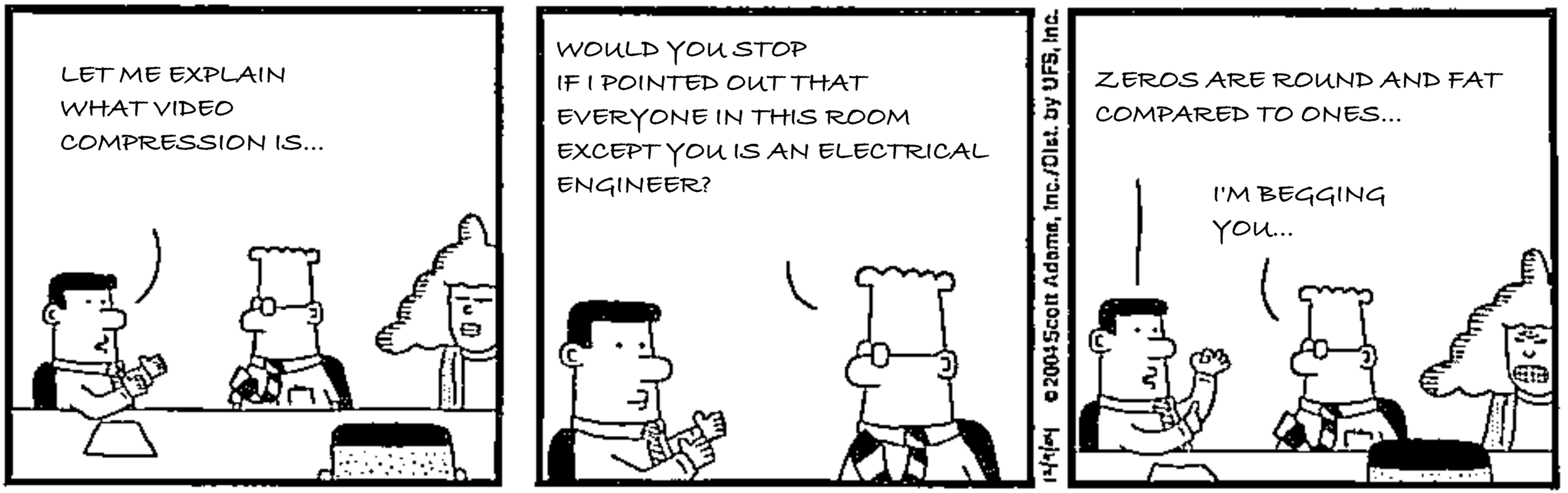
Advanced Feedback Driven Development
and Globally Distributed Projects

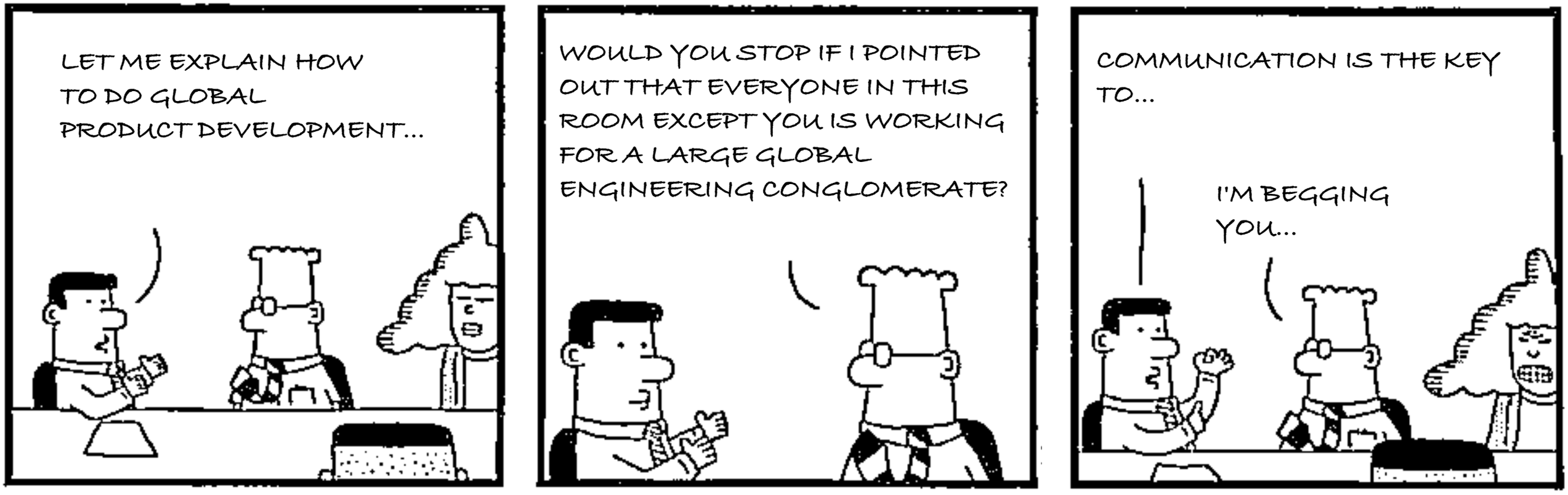
The keynote at
SIEMENS Global Development Experience Day
Erlangen, May 4, 2010

Olve Maudal
olve.maudal@tandberg.com



Outline

- About me, Tandberg and what we do
- A case study - Advanced Feedback Driven Development
- Observations from Tandberg R&D
- Scaling into global development
- Some final thoughts about software development



About me

1992-1995 Software Engineering, UMIST, Manchester
1995-1996 Intelligent Robotics, Dept of Artificial Intelligence, Edinburgh
1996-1996 Data Mining and Knowledge Discovery, NTNU, Trondheim
1996-2000 Schlumberger, developing systems for finding oil
2000-2004 BBS, developing systems for electronically moving money
2004-now TANDBERG, developing systems for effective communication between people

Active member of the vibrant geek community in Oslo. Eg, JavaPils, Smidig, JavaZone, XP Meetup, Cantara, Lean Meetup, Rubberducks and Oslo C++ Users Group, and a lot of other things. Also an active member of ACCU.

Blogs regularly on http://olvemaudal.wordpress.com/ and Twitter @olvemaudal

About TANDBERG

TANDBERG is the leading provider of telepresence, high-definition video conferencing and mobile video products and services. TANDBERG designs, develops and markets systems and software for video, voice and data. The company provides sales, support and value-added services in more than 90 countries worldwide.

TANDBERG shipped it's first product, a picture telephone for ISDN in 1993. Since then TANDBERG has grown from a small startup based in Norway into an international company with ~1600 employees and a revenue of 800 MUSD in 2008. ~450 engineers works in R&D with product development.

Dual headquarters in New York and Oslo. R&D centres at Lysaker (NO), Langley (UK), Ruscombe (UK), Bangalore (IN) and Hamilton (NZ).

www.tandberg.com

About TANDBERG

TANDBERG is the leading provider of telepresence, high-definition video conferencing and mobile video products and services. TANDBERG designs, develops and markets systems and software for video, voice and data. The company provides sales, support and value-added services in more than 90 countries worldwide.

TANDBERG shipped it's first product, a picture telephone for ISDN in 1993. Since then TANDBERG has grown from a small startup based in Norway into an international company with ~1600 employees and a revenue of 800 MUSD in 2008. ~450 engineers works in R&D with product development.

Dual headquarters in New York and Oslo. R&D centres at Lysaker (NO), Langley (UK), Ruscombe (UK), Bangalore (IN) and Hamilton (NZ). And San Jose (US)

www.tandberg.com

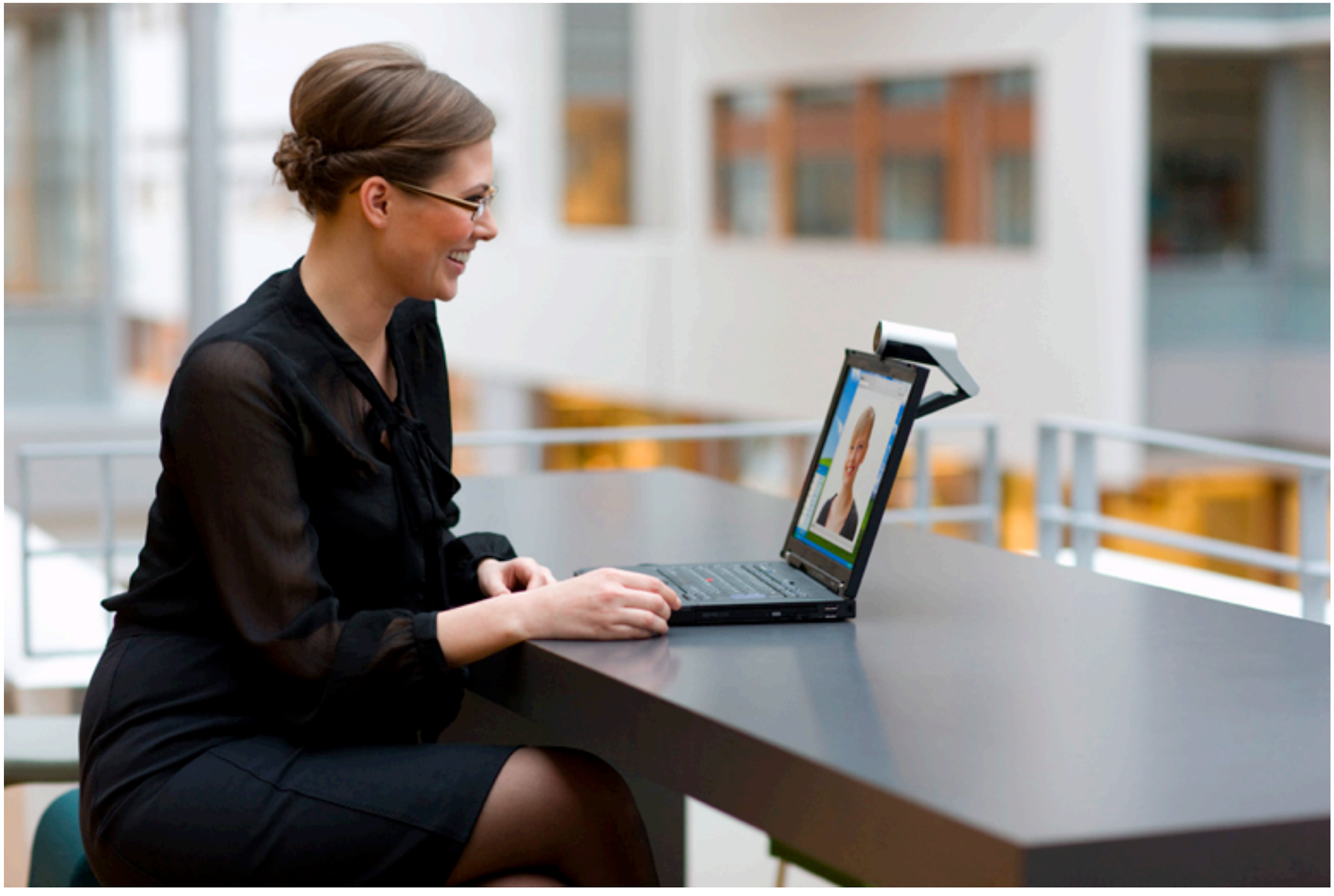
Breaking news (April 18, 2010):
After a successful $3.8 billion deal, TANDBERG is now a part of Cisco

We develop and sell...

Meeting room systems

Telepresence systems

Personal systems



PC based solutions

Networking products

And a lot of other stuff


TANDBERG

Video: The new way of working (2:00)

http://www.tandberg.com/media/index.jsp?id=1373

A case study:

TANDBERG Codec C90 - “The Saturn Project”

How did we do it?

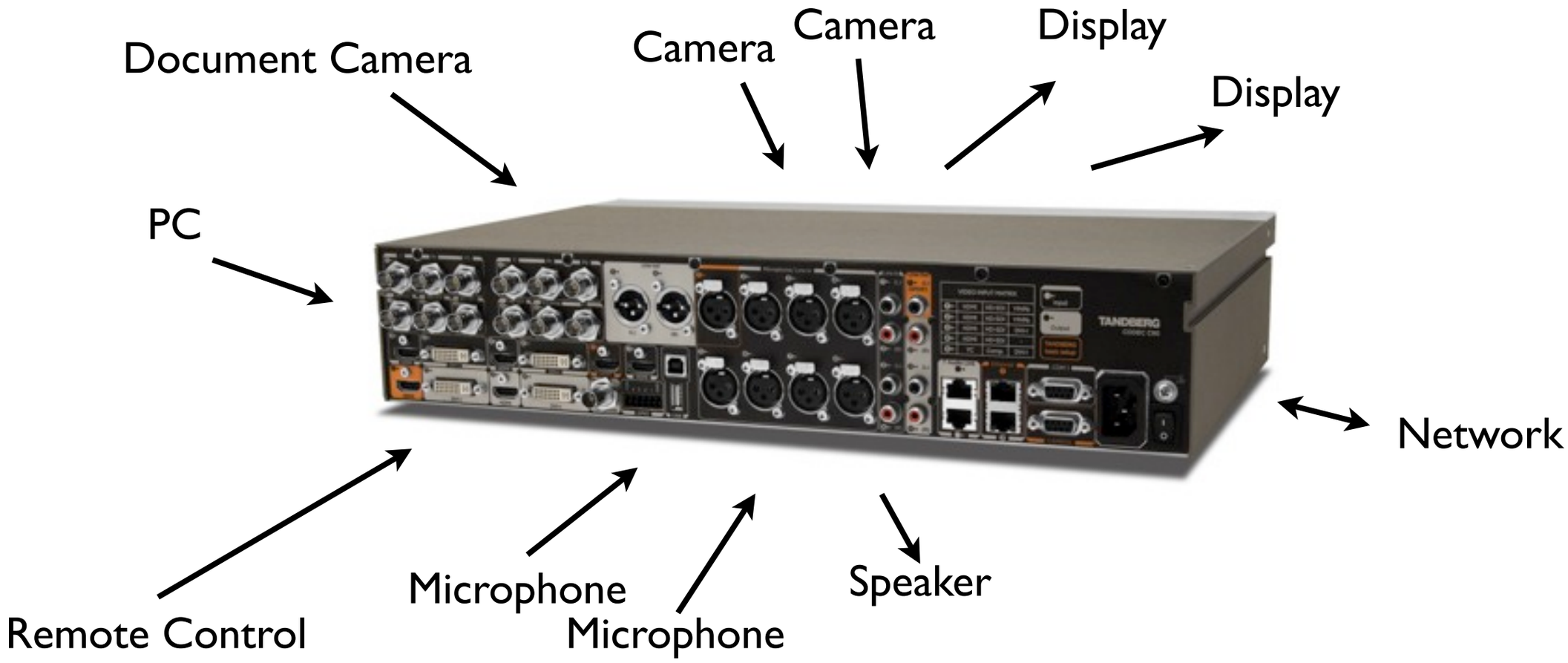
Disclaimer:

The following description does not show how projects in Tandberg are typically developed, it is just an example of how a particular project actually did it. We think about every project, product and team as something unique, thus it does not make sense to create a particular procedure to follow.

Indeed, when it comes to product development, TANDBERG is "allergic" to corporate procedures. It is "unthinkable" that anyone outside a project or a team should impose a certain way of doing things, so we can not say "This is the way we do it", but you may look at a particular project and say "This is the way we did it".

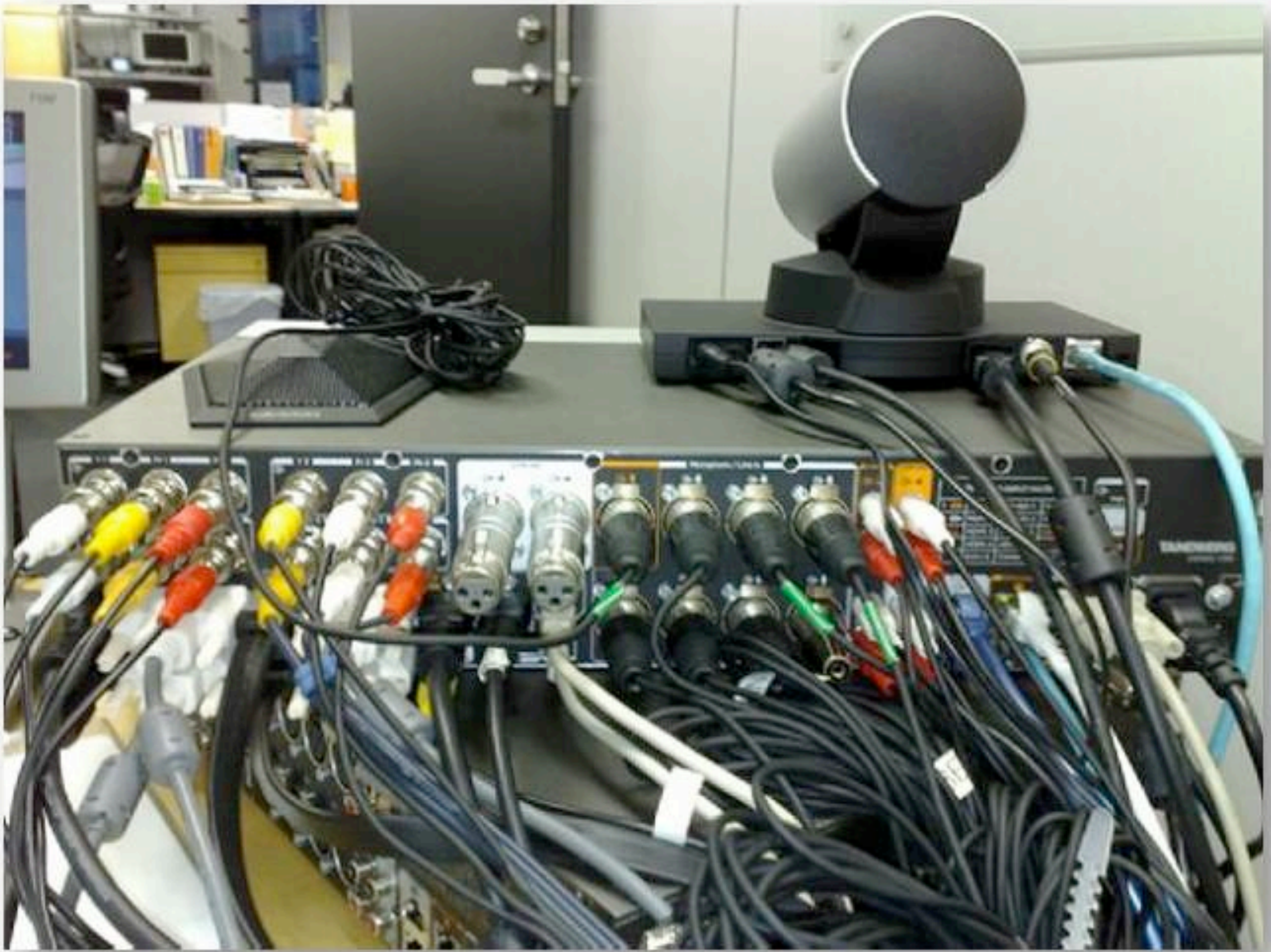
C90 video (1:19)

http://www.tandberg.com/media/index.jsp?id=1312



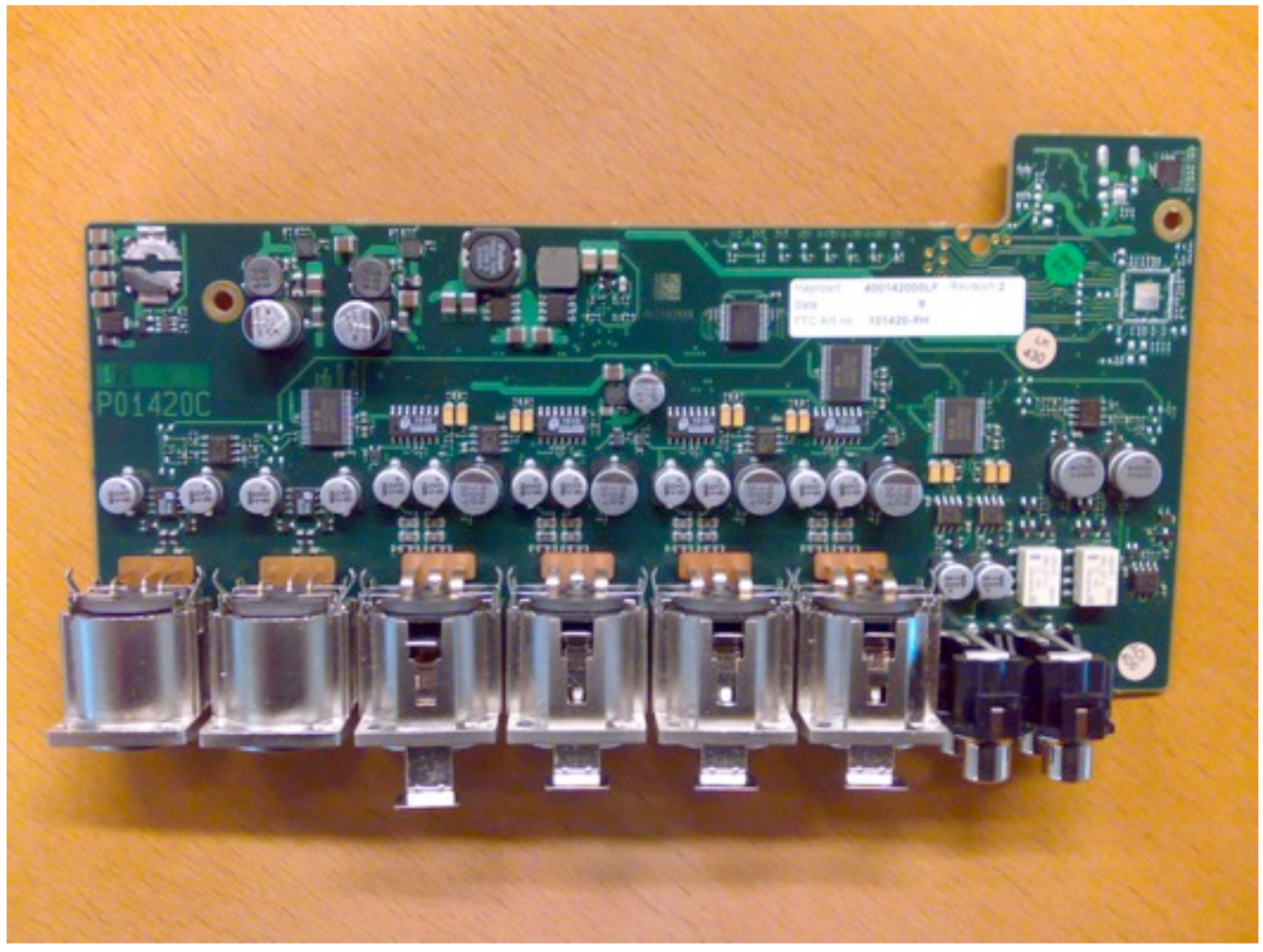
C90 Features:

- realtime H.264 encoding/decoding
- full HD 1080p30, (4+4) concurrent streams
- 12 high definition video sources
- 8 high quality audio sources
- support for many-to-many communication
- Interoperability through H323 and SIP
- API for integration and remote control



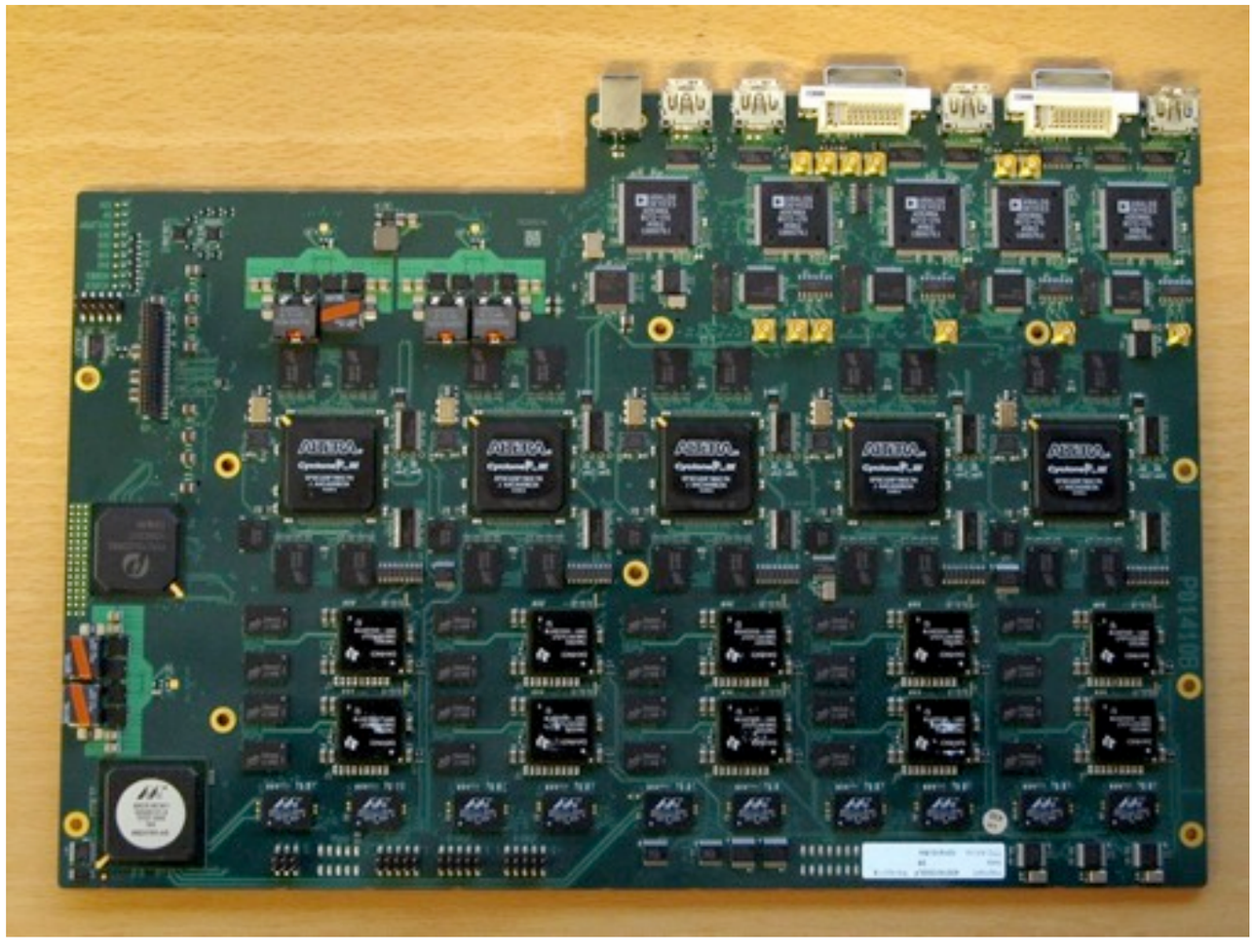
C90 AUDIO EXTENSION BOARD

- analog amplification
- high quality AD and DA converters
- pure electronics, no processor/SW
- 717 components
- 6 layers



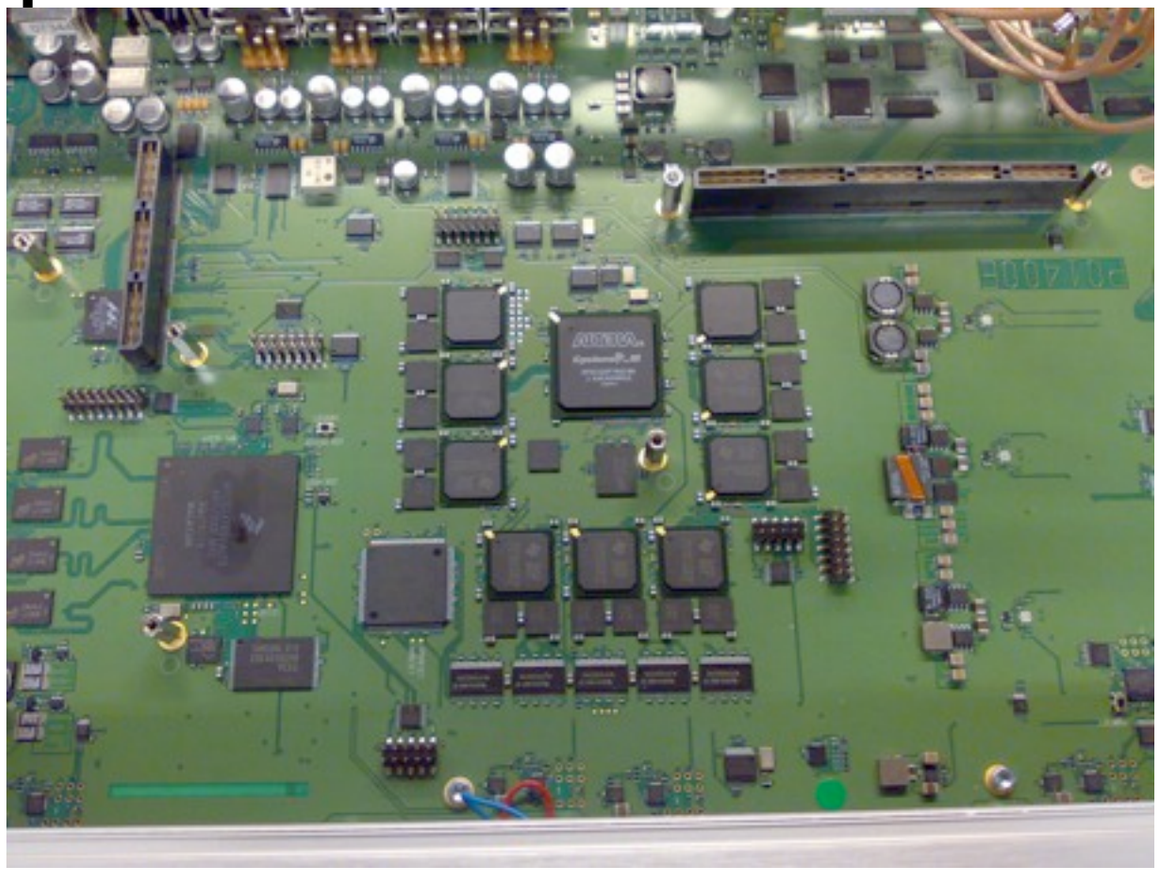
C90 VIDEO BOARD

- 10 Da Vinci DM6467 for video compression/ decompresion(1 ARM, 1 dsp, 2 coprocessors),
- 5 Altera Cyclone III 120 for video scaling & composing(Nios II softcore 50 MHz)
- 15 Gbps video backplane
- 3.8 GByte DDR2 RAM
- 128 mbit x5 SDRAM
- 6097 components
- 30520 pins
- 22 layers
- 6490 nets



C90 MAIN BOARD

- 1 Altera Cyclone III 120 for Audio switching (Nios II softcore 50 MHz)
- 9 TI 6727, audio dsp for echo control, compression, decompression, +++
- PowerPC 8347, main processor, application software, networking, user interface
- 3543 components / 15659 pins
- 16 layers
- 3264 nets



C90 - from a geek point of view

- 10000+ components
- 44 (6+22+16) layers
- 56 processor cores
- several million lines of code (C and C++)

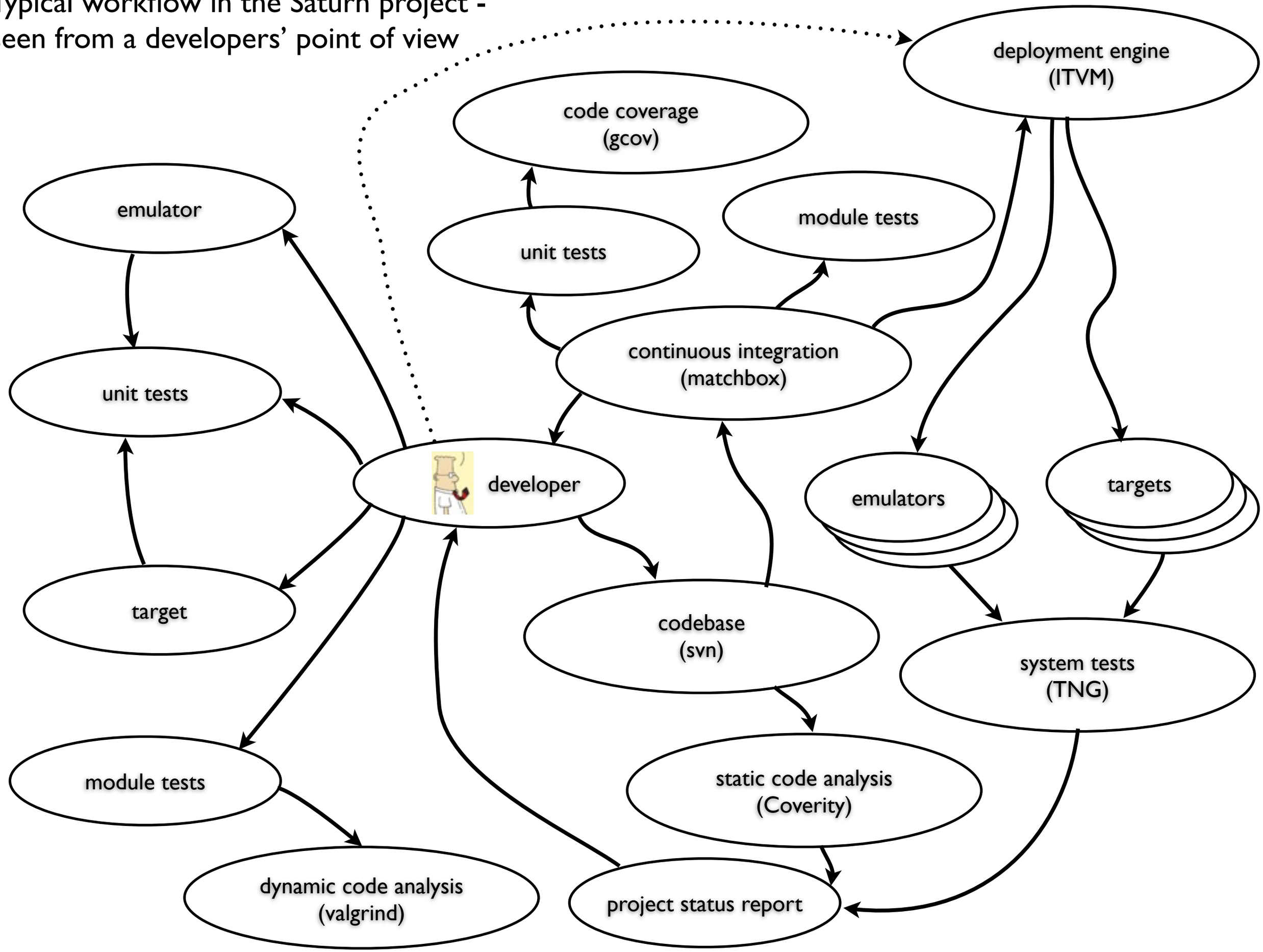
TANDBERG Codec C90

- Developed at Lysaker
- Started spring 2007
- First HW prototype arrived summer 2008
- Released late 2008 (~20 months of development)
- 2-3 people working with mechanics/design
- 4-5 people working with electronics/hardware
- 5-6 people working with FPGA development
- 40-50 people working with software development
- 4 people working with test developers
- 1 person working with approvals

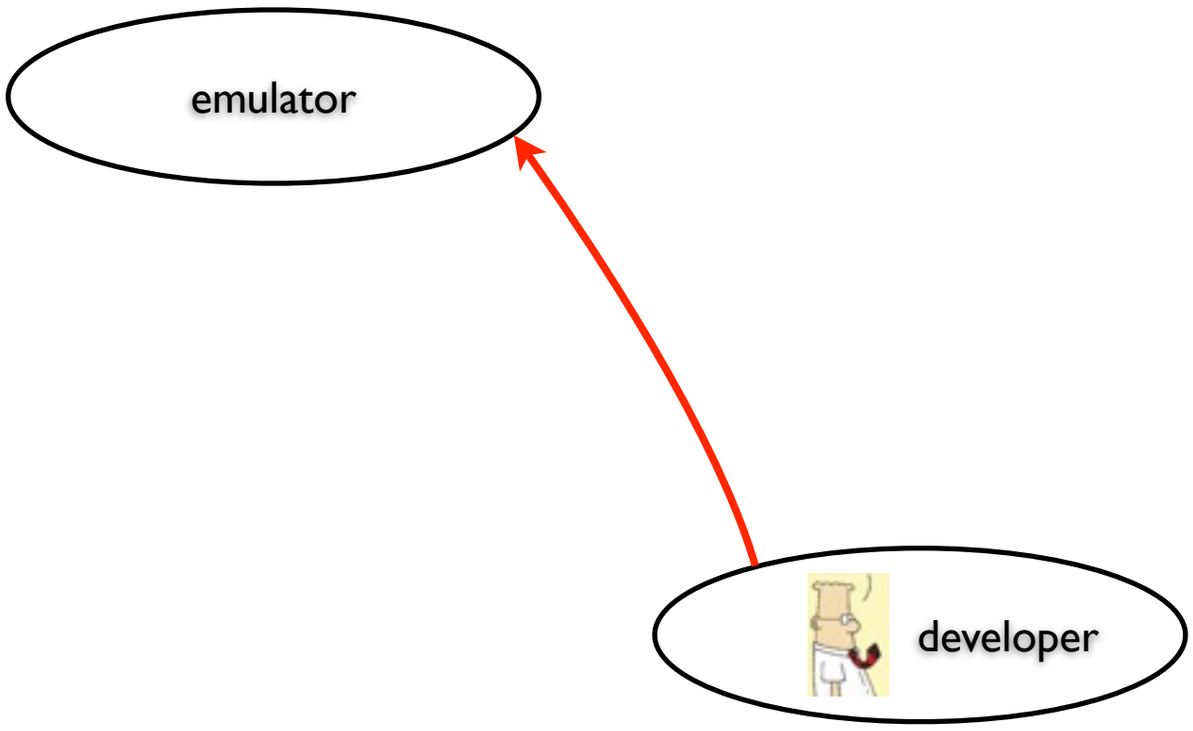
Development Practices in the Saturn project

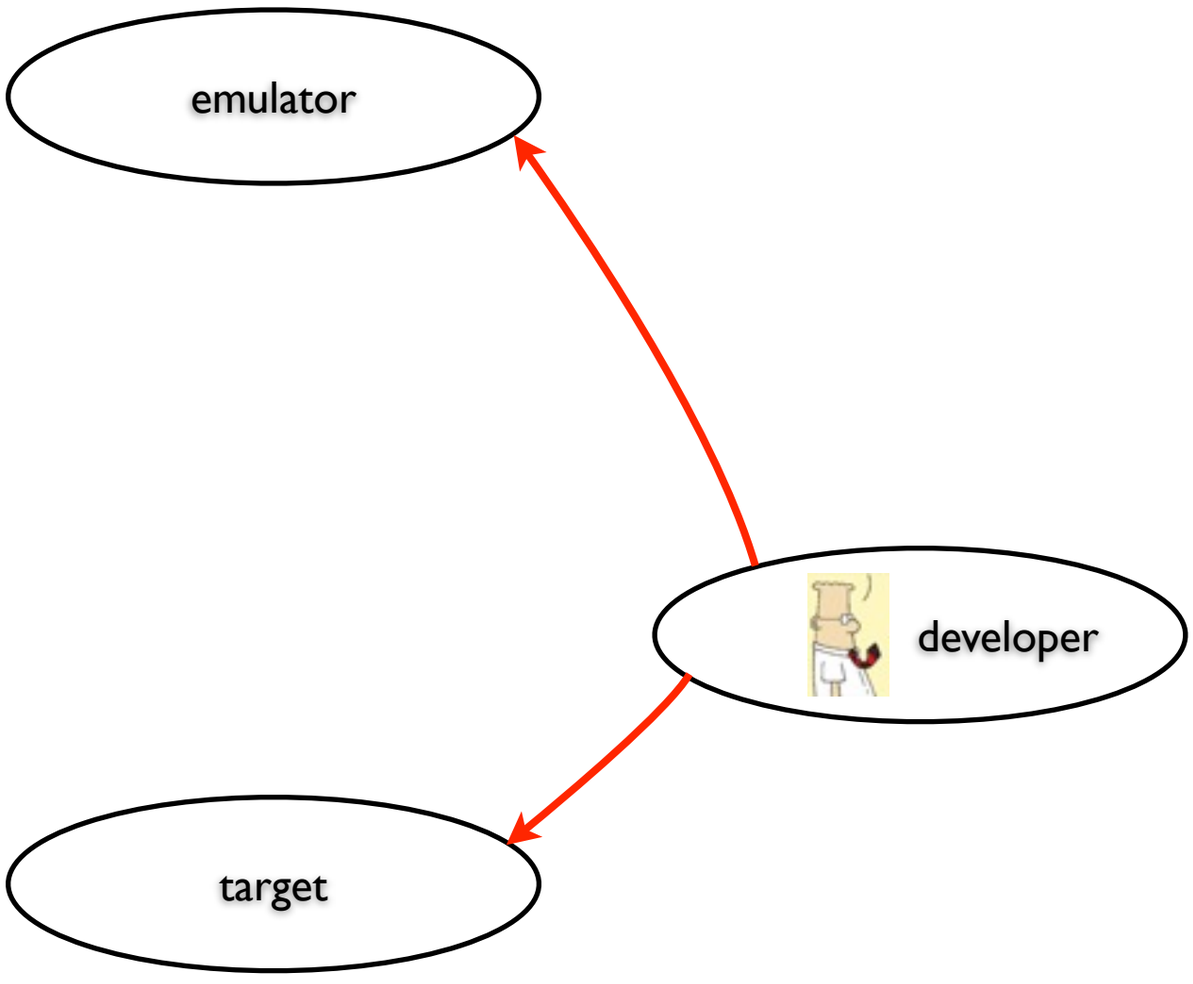
- Continuous planning
- Always attack high risks first
- Heavy focus on effective feedback mechanisms
- Visualization of actual status throughout project
- Teams: GUI, App, Protocol, Video, Audio, FPGA, Platform, QA, Support
- Parallel development
- Iterations and time-boxing
- Daily meetings of the elderly
- Weekly rendezvous meetings
- Early and many prototypes

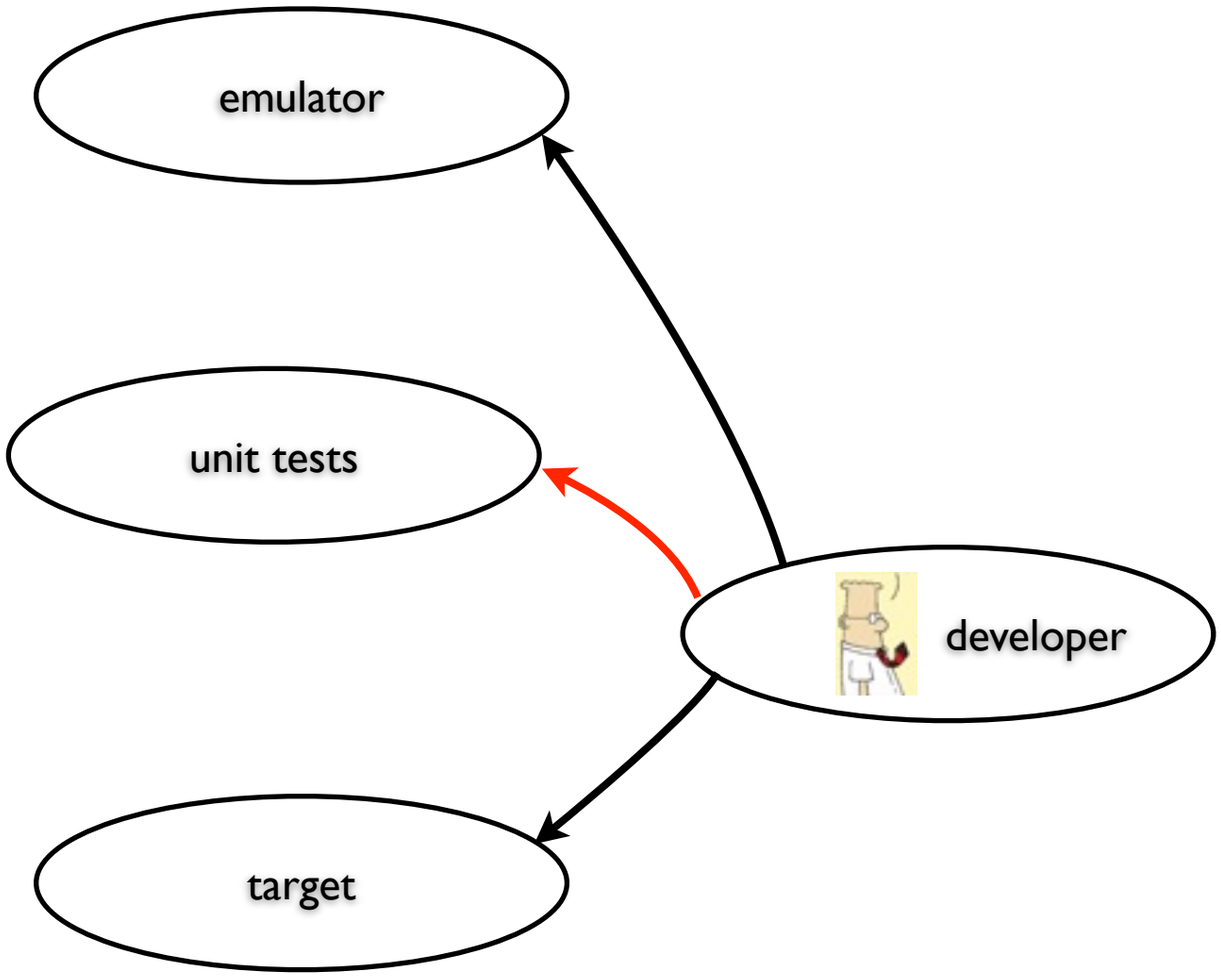
Typical workflow in the Saturn project -
seen from a developers' point of view

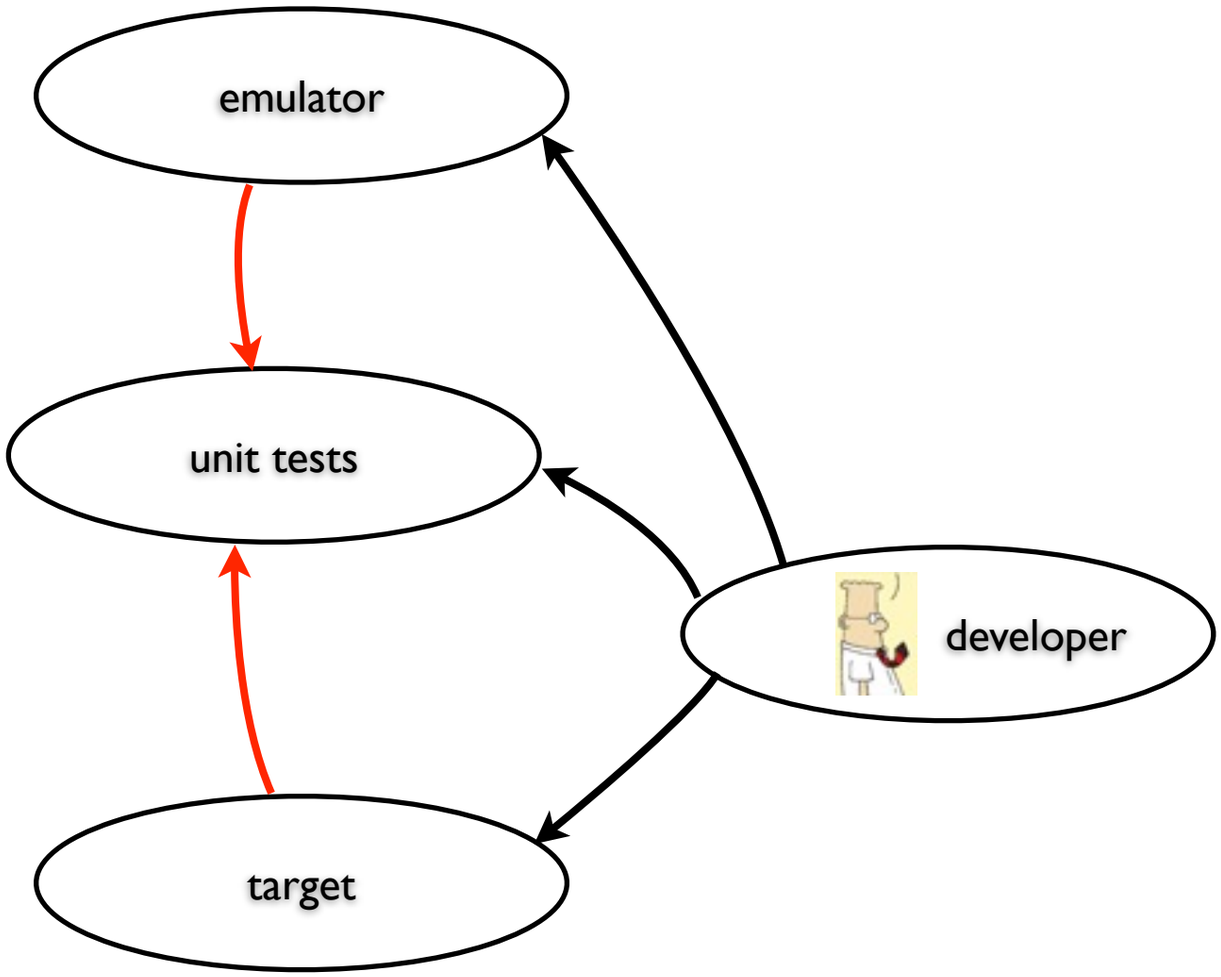


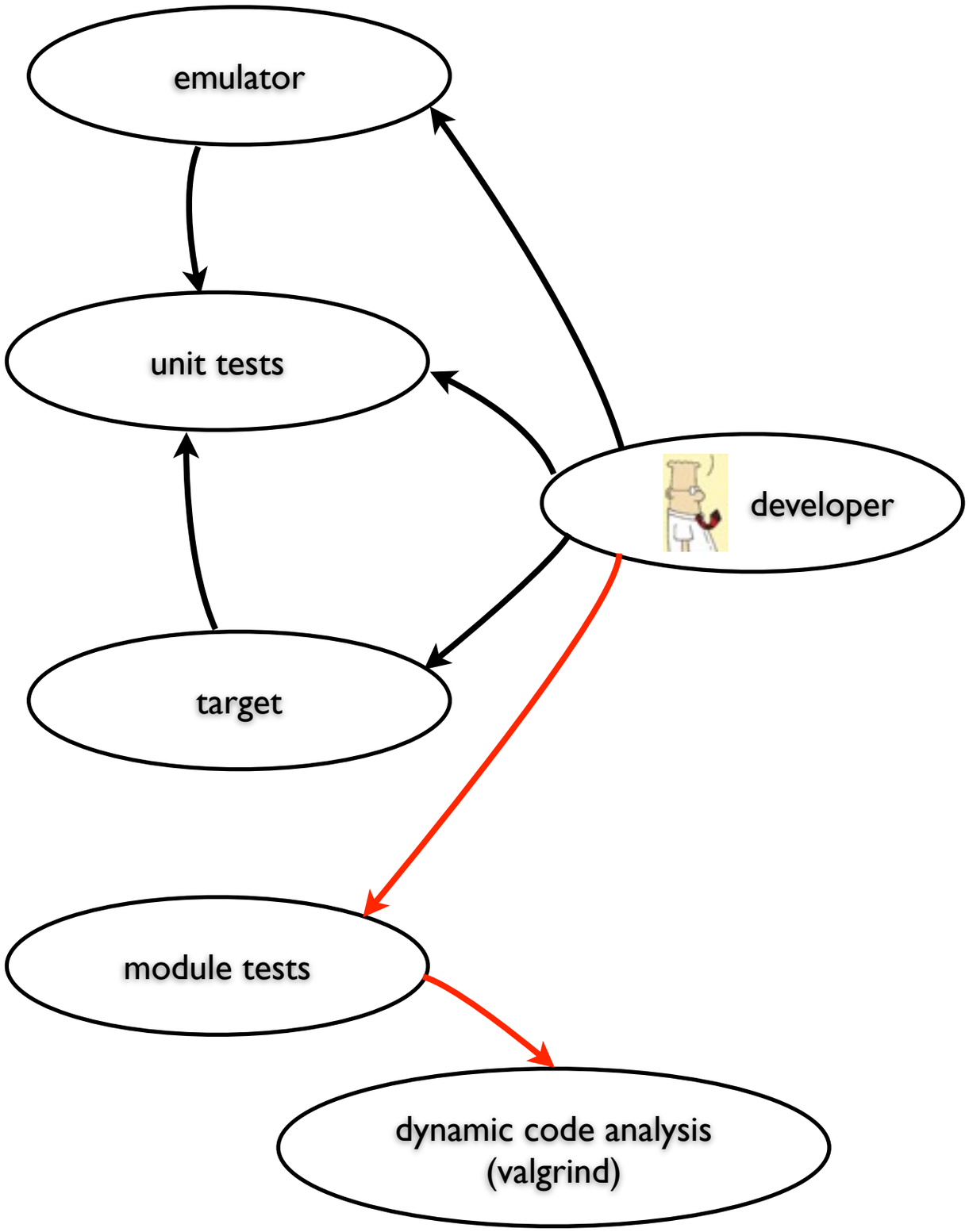
Software development in the Saturn project as seen from a developers point of view.

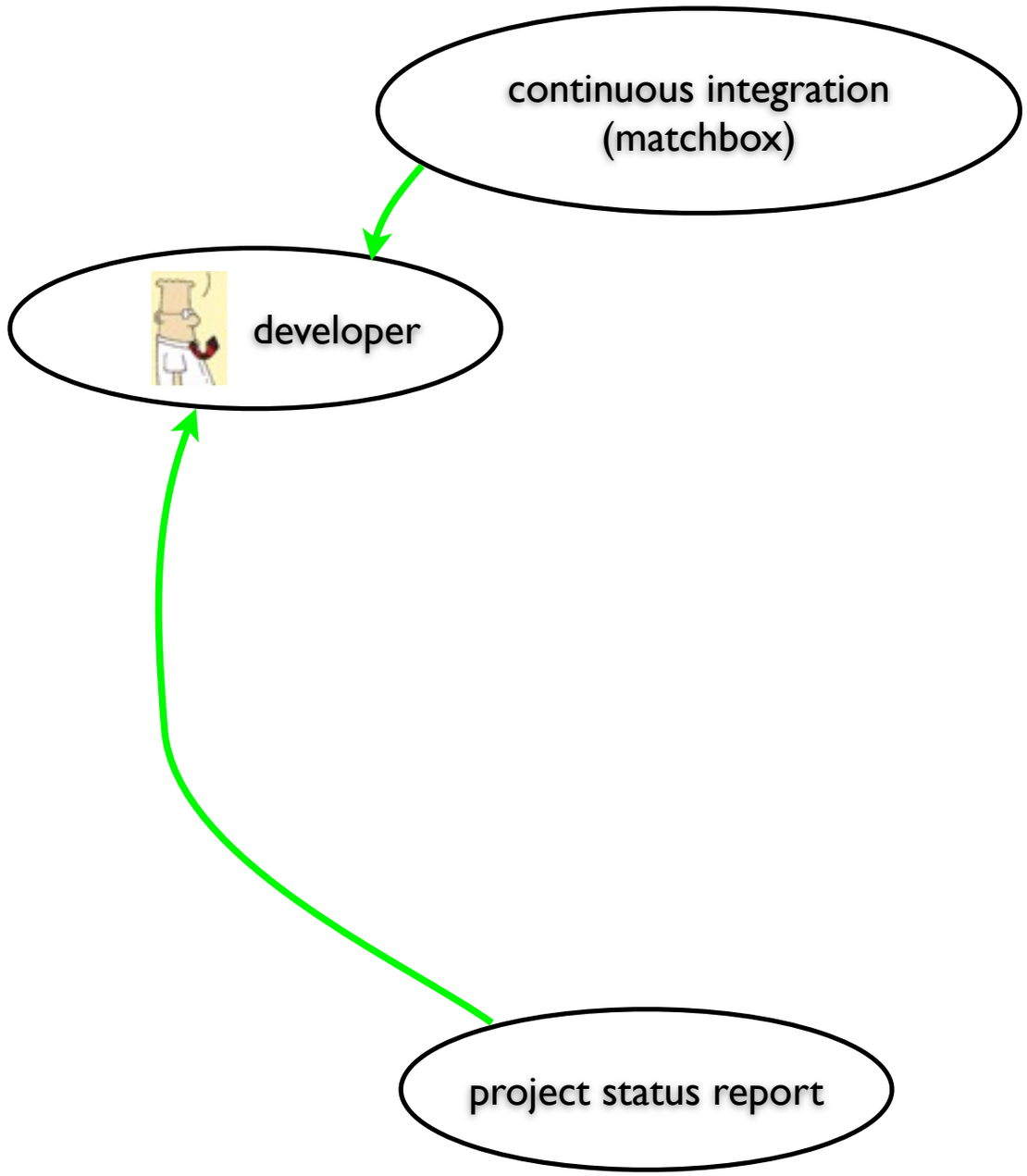

emulator
developer

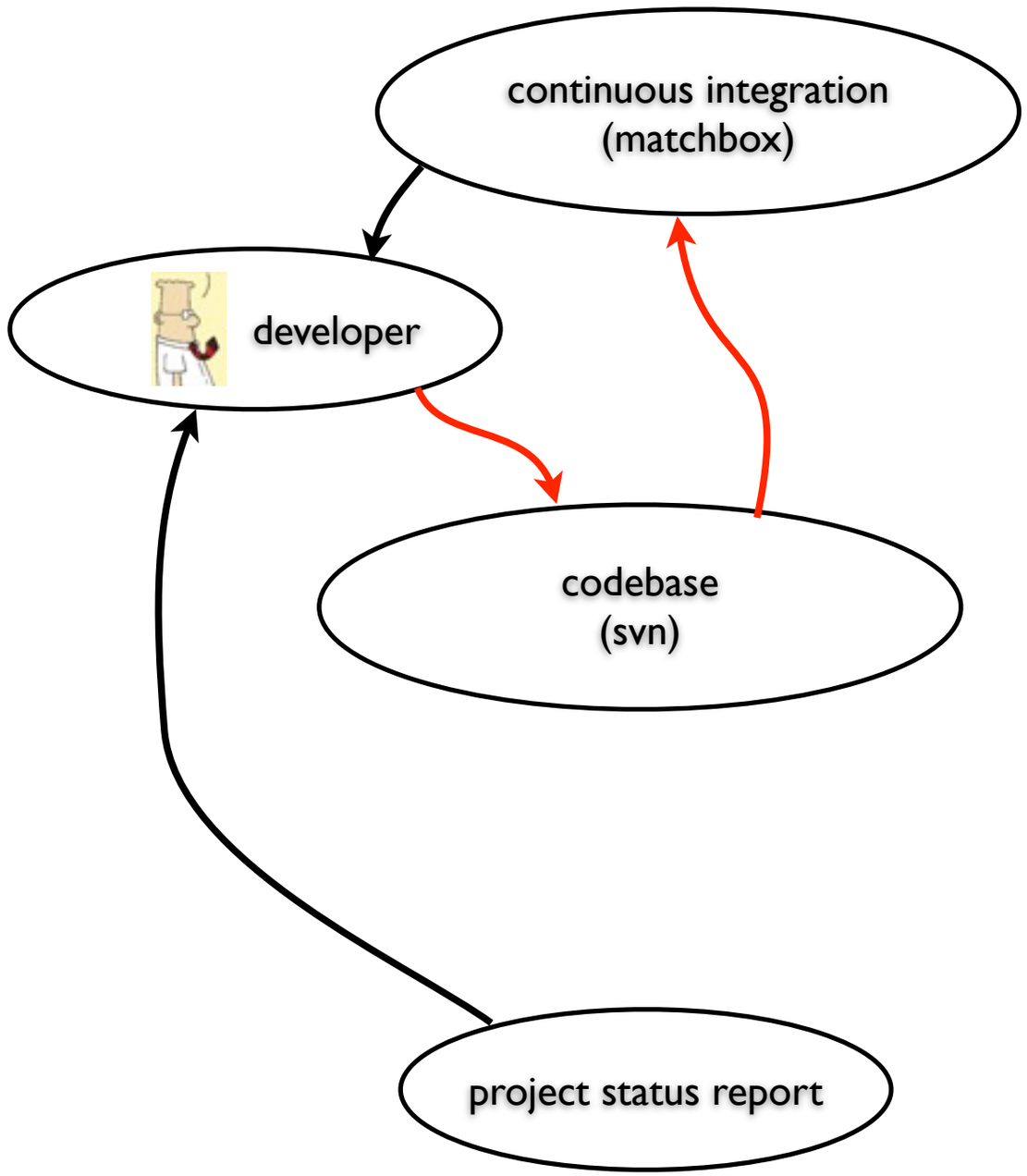

emulator
developer
target

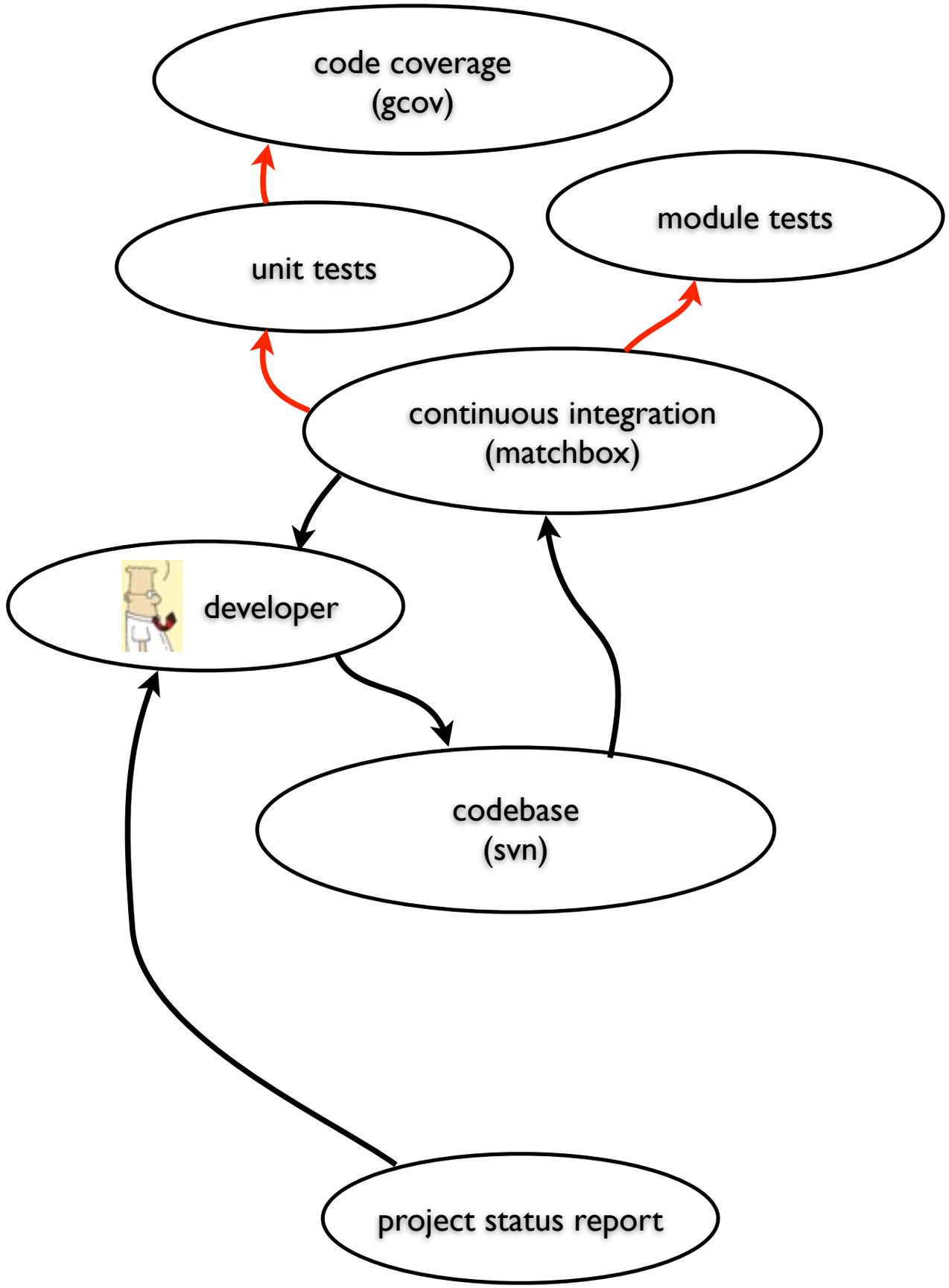

emulator
unit tests
developer
target

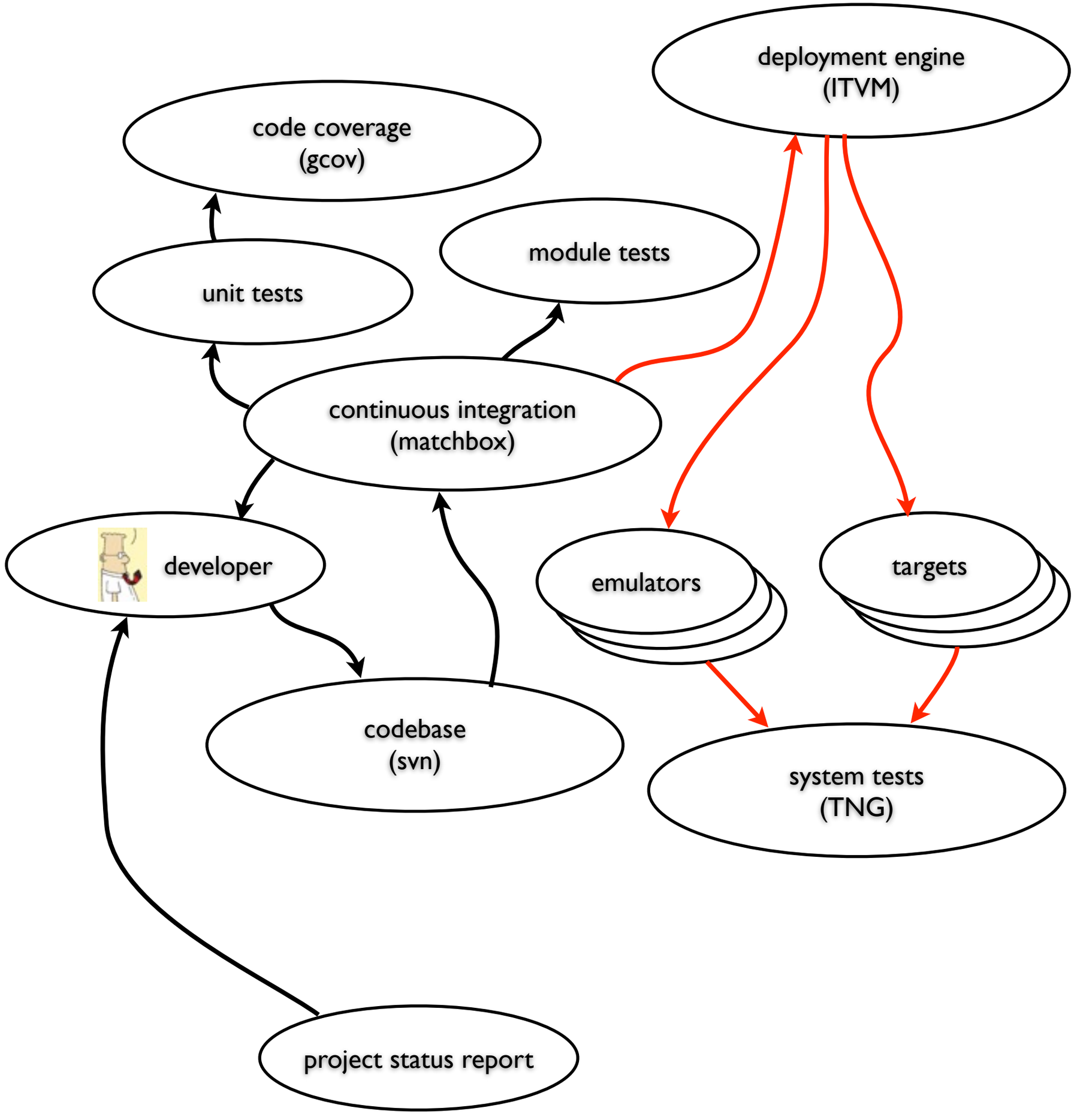

emulator
unit tests
developer
target

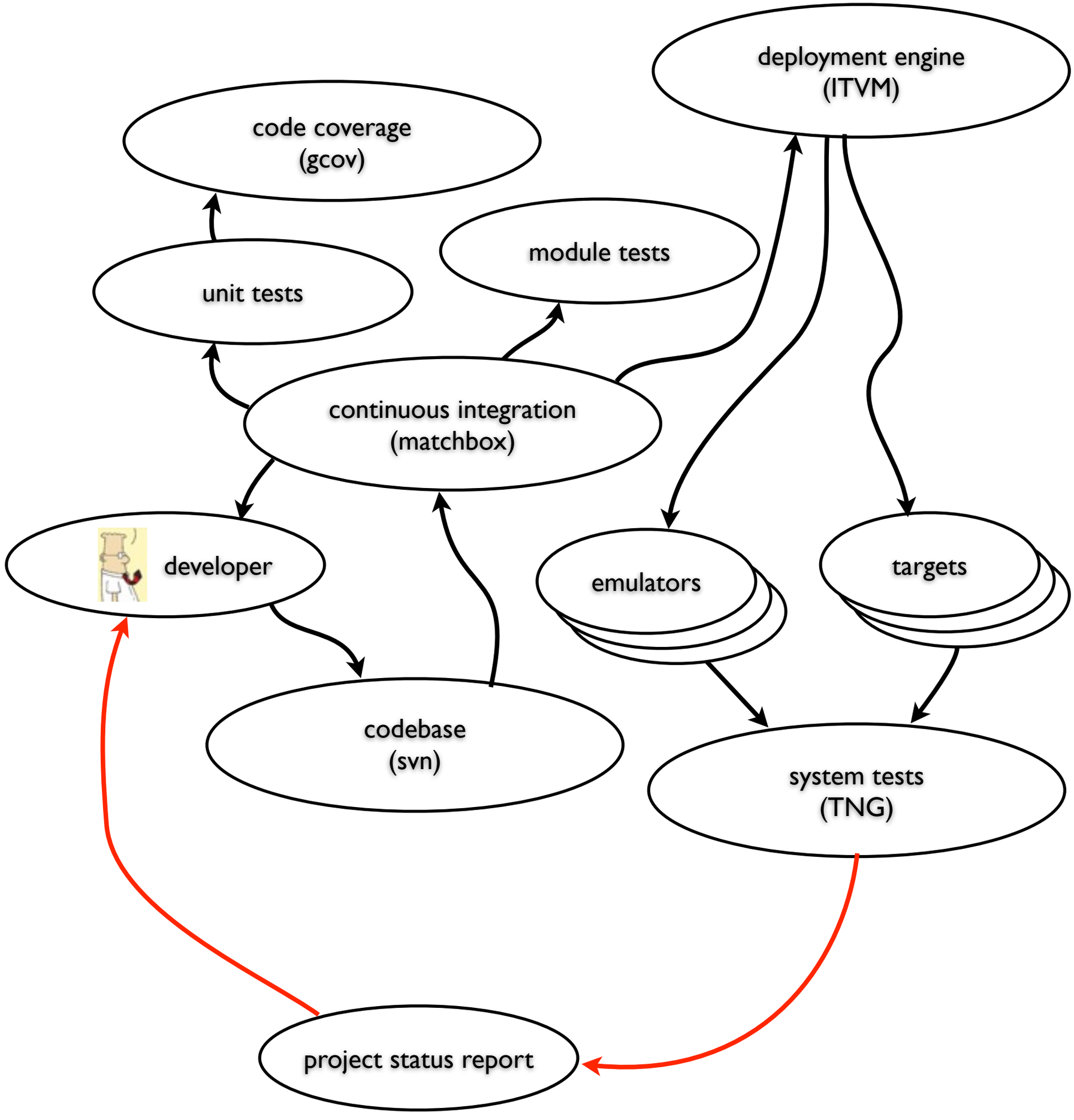

emulator
unit tests
developer
target
module tests
dynamic code analysis
(valgrind)

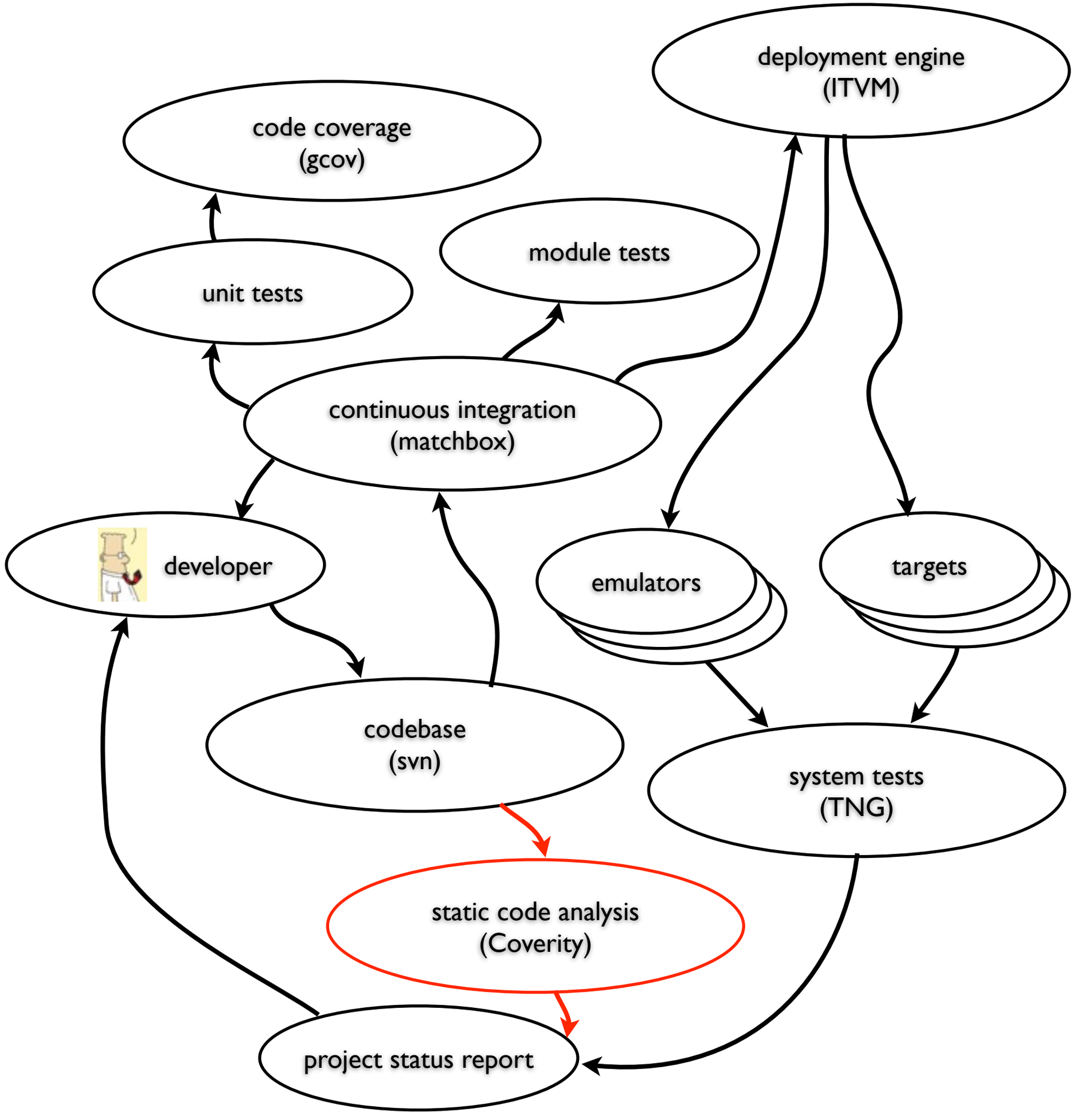

continuous integration
(matchbox)
developer
project status report

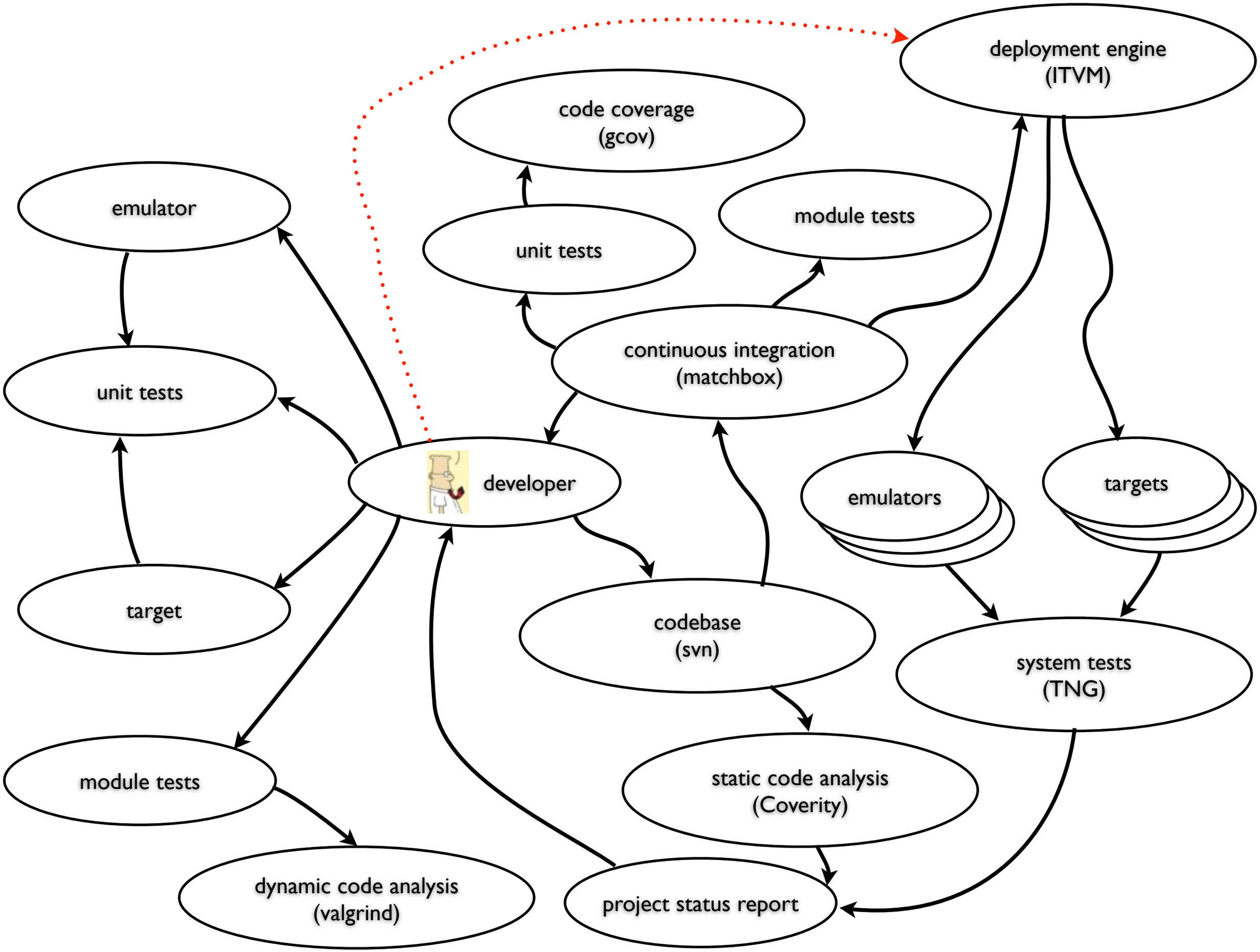

continuous integration
(matchbox)
developer
codebase
(svn)
project status report

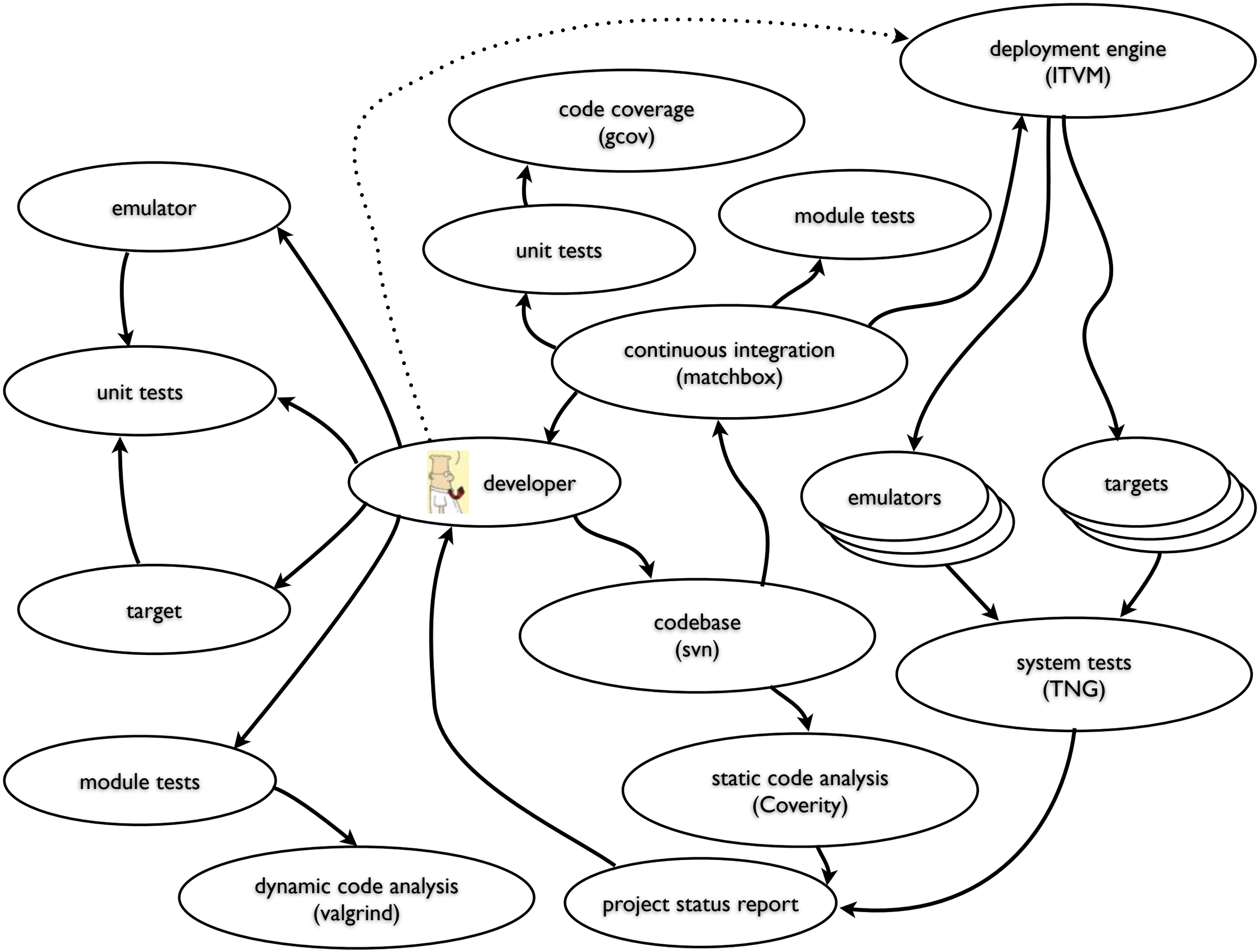

code coverage
(gcov)
unit tests
module tests
continuous integration
(matchbox)
developer
codebase
(svn)
project status report


deployment engine
(ITVM)
code coverage
(gcov)
module tests
unit tests
continuous integration
(matchbox)
developer
emulators
targets
codebase
(svn)
system tests
(TNG)
project status report


deployment engine
(ITVM)
code coverage
(gcov)
module tests
unit tests
continuous integration
(matchbox)
developer
emulators
targets
codebase
(svn)
system tests
(TNG)
project status report


deployment engine
(ITVM)
code coverage
(gcov)
module tests
unit tests
continuous integration
(matchbox)
developer
emulators
targets
codebase
(svn)
system tests
(TNG)
static code analysis
(Coverity)
project status report

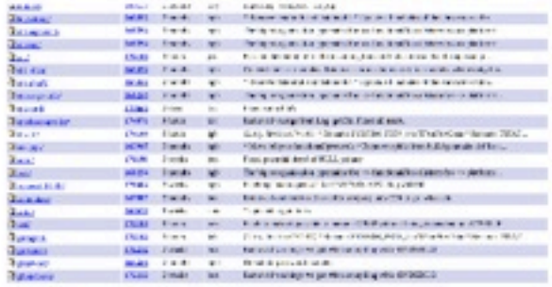

deployment engine
(ITVM)
code coverage
(gcov)
emulator
module tests
unit tests
unit tests
continuous integration
(matchbox)
developer
emulators
targets
target
codebase
(svn)
system tests
(TNG)
module tests
static code analysis
(Coverity)
dynamic code analysis
(valgrind)
project status report


deployment engine
(ITVM)
code coverage
(gcov)
emulator
module tests
unit tests
unit tests
continuous integration
(matchbox)
developer
emulators
targets
target
codebase
(svn)
system tests
(TNG)
module tests
static code analysis
(Coverity)
dynamic code analysis
(valgrind)
project status report

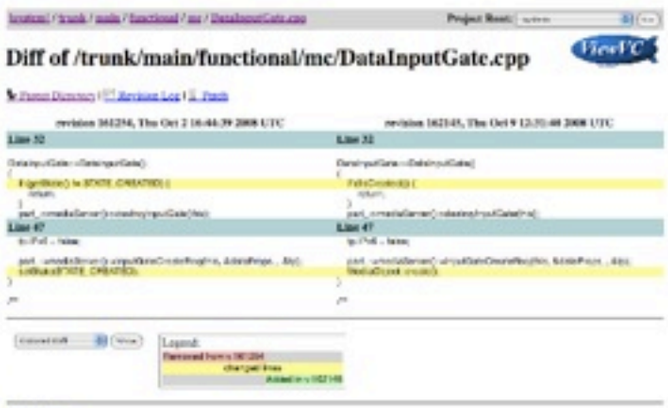
Other aspects of the Saturn project:

- IRC channels
- free choice of development platform
- lot of energy spent on software emulator of actual hardware
- project manager is also configuration manager / build master
- static code analysis (Coverity)
- dynamic code analysis (valgrind)
- build system (genmake2, inhouse python)
- automatic deployment engine (ITVM, inhouse C#)
- automatic system testing (TNG, inhouse python)
- unit test framework (unittest, inhouse C and C++)
- module test framework (inhouse C++)
- code coverage (gcov)
- continuous integration system (matchbox, inhouse python)

Example of visual feedback (HTML pages used by all/most developers)



viewvc



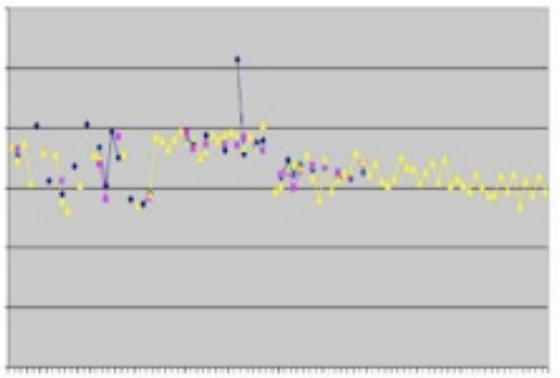
diff from viewvc

audio delay trend

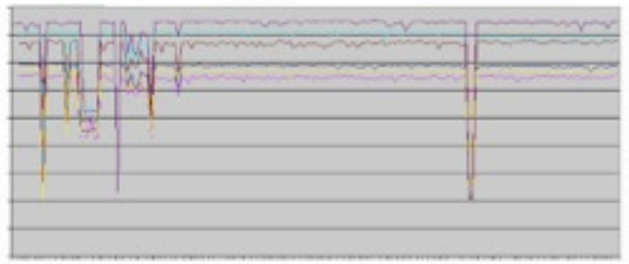
bugzilla

irc channel

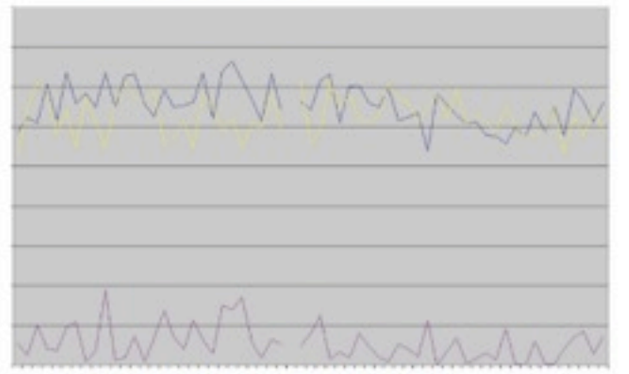
continuous integration



H.264 delay trend

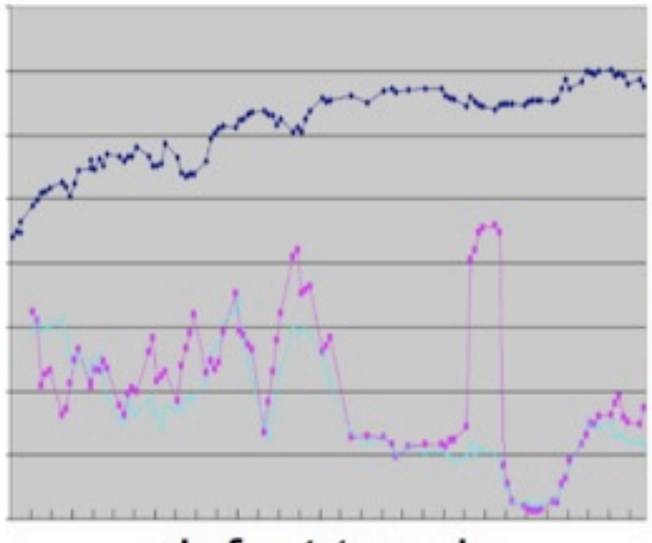


PESQ trend

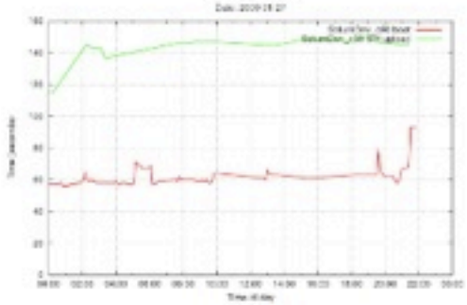


lipsync trend

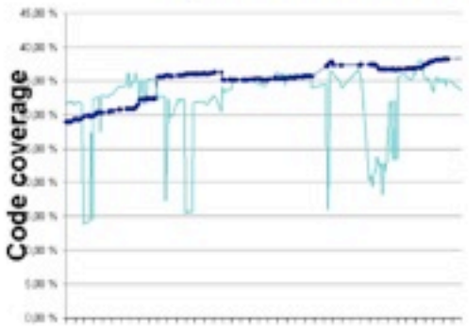
system tests



defect trends

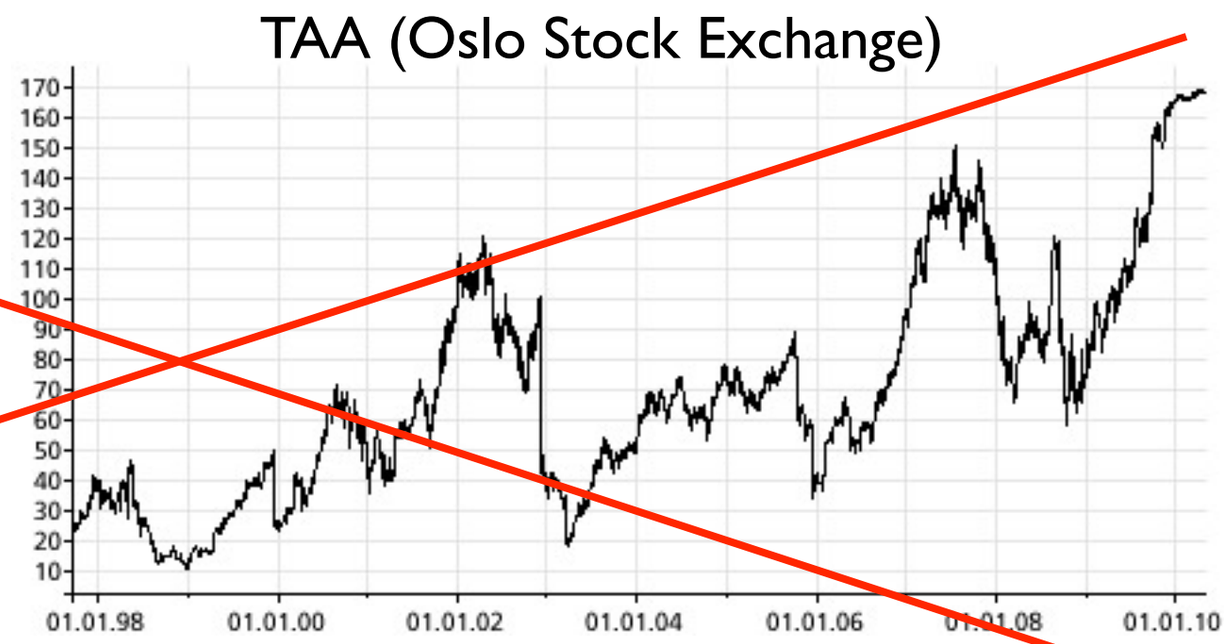


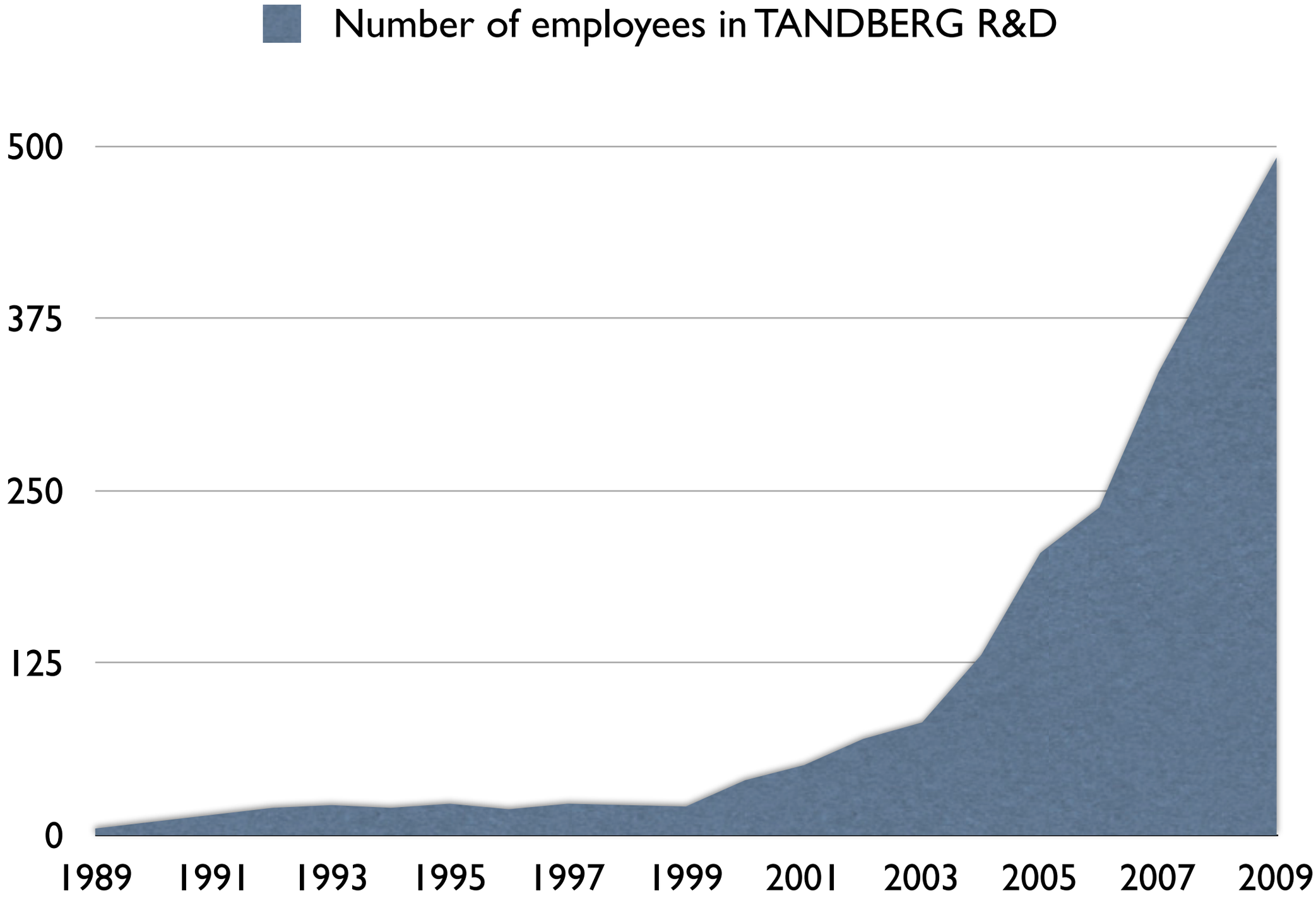
endpoint timing



coverity

QA Status


TAA (Oslo Stock Exchange)
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
20
10
01.01.98
01.01.00
01.01.02
01.01.04
01.01.06
01.01.08
01.01.10



(the numbers are not exact)

As a software engineer joining TANDBERG...

at first you might get this impression...

- No documentation
- No routines
- Fooling around
- Not following plans
- Decision are postponed
- Nobody decides
- Little respect for management
- Little modularization
- Lack of precision
- Sloppiness
- People are not working hard

but then you start to notice...

- No documentation
- No routines
- Fooling around
- Not following plans
- Decision are postponed
- Nobody decides
- Little respect for management
- Little modularization
- Lack of precision
- Sloppiness
- People are not working hard

- People communicate
- Focus on important stuff
- Embedded slack
- Continuous planning
- Effective decisions
- Autonomous organisation
- Respect for the doers
- No integration period
- Spectacular products
- Fast deliveries
- Sustainable pace

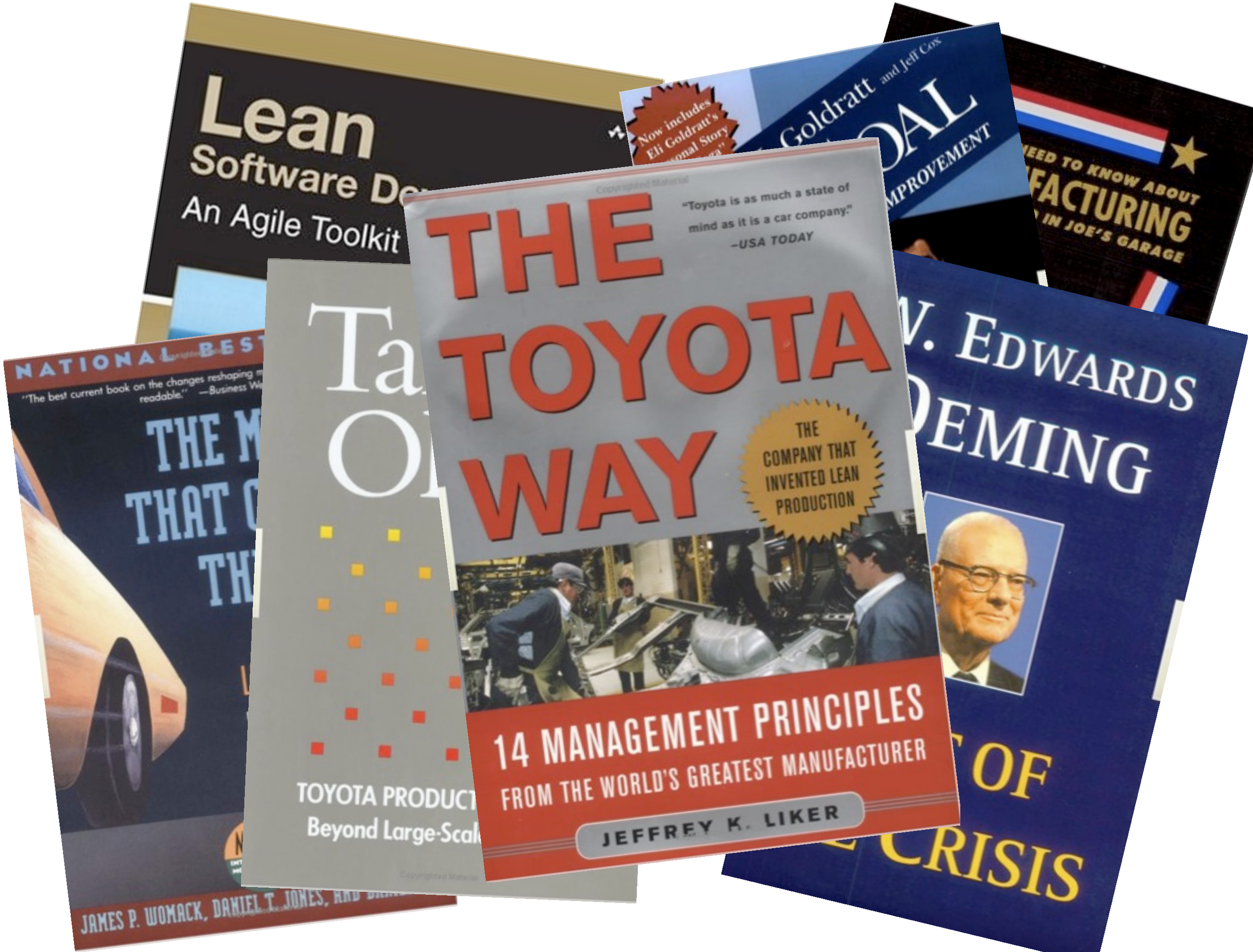
... and while you still see the "negative" stuff, you will start to appreciate the "positive" stuff more.

- No documentation
- No routines
- Fooling around
- Not following plans
- Decision are postponed
- Nobody decides
- Little respect for management
- Little modularization
- Lack of precision
- Sloppiness
- People are not working hard

- People communicate
- Focus on important stuff
- Embedded slack
- Continuous planning
- Effective decisions
- Autonomous organisation
- Respect for the doers
- No integration period
- Spectacular products
- Fast deliveries
- Sustainable pace

Observations from TANDBERG

- People communicate
- Focus on important stuff
- Embedded slack
- Continuous planning
- Effective decisions
- Autonomous organisation
- Respect for the doers
- No integration period
- Spectacular products
- Fast deliveries
- Sustainable pace


Lean
An Agile Toolkit
Now includes Eli Goldratt's
Goldratt and Jeff Cox
IMPROVEMENT
NEED TO KNOW ABOUT
FACTURING
IN JOE'S GARAGE
NATIONAL BEST
"The best current book on the changes reshaping
readable." —Business
THE M
THAT C
TH
JAMES P. WOMACK, DANIEL T. JONES,
Ta
O
TOYOTA PRODUCT
Beyond Large-Scal
THE TOYOTA WAY
"Toyota is as much a state of mind as it is a car company."
–USA TODAY
THE COMPANY THAT INVENTED LEAN PRODUCTION
14 MANAGEMENT PRINCIPLES
FROM THE WORLD'S GREATEST MANUFACTURER
JEFFREY K. LIKER
W. EDWARDS
DEMING
T OF
CRISIS

THE TANDBERG WAY

Product Development in TANDBERG

- No corporate standards or procedures
- Little documentation gives effective communication
- Treat engineers as professionals, not as resources
- Slack is embedded, and “skunk work” projects appreciated
- "Plans are nothing, planning is everything"
- No time recording, and we do not measure project cost
- To fail is OK, therefore we deliver spectacular stuff
- Doers are very much respected in Tandberg
- Autonomous organization
- Communication is a key skill for all our engineers
- We are fast and “sloppy”
- We release early and we release often
- Fun gives profit (not: profit, then fun)
- The company builds on trust

We follow principles, not processes!

Scaling into global development

- rule 0: avoid multisite development, if you can...
- visual communication is essential
- time difference is more challenging than geographical separation
- use continuous integration and automatic testing
- working across trust boundaries is painful
- balance the sites
- cultural differences is more challenging than time difference
- prefer local decisions
- accept duplication of work, beware the share
- move people around, use ambassadors
- beware of the Cover My Ass game
- avoid contracted interfaces, use mediators
- use tools for microcommunication
- focus on system architecture
- knowledge management
- corporate culture

Few software projects are like running on a paved road where you can see the ...

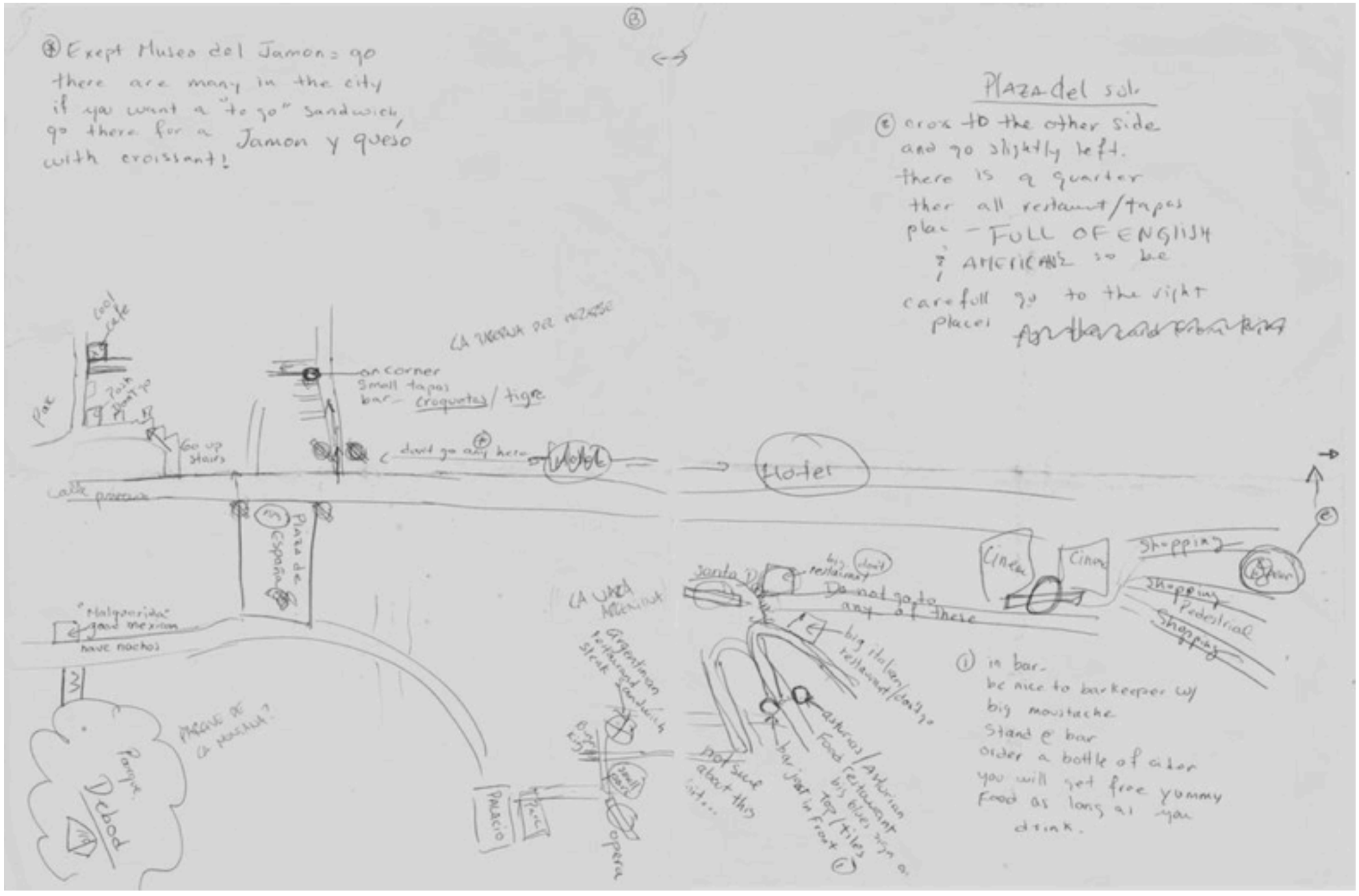
... goal in the end of the road.

Most projects are more like...

extreme orienteering

with a group of people

in the dark



Ⓑ Exept Museo del Jamon: go there are many in the city if you want a "to go" sandwich, go there for a Jamon y queso with croissant!

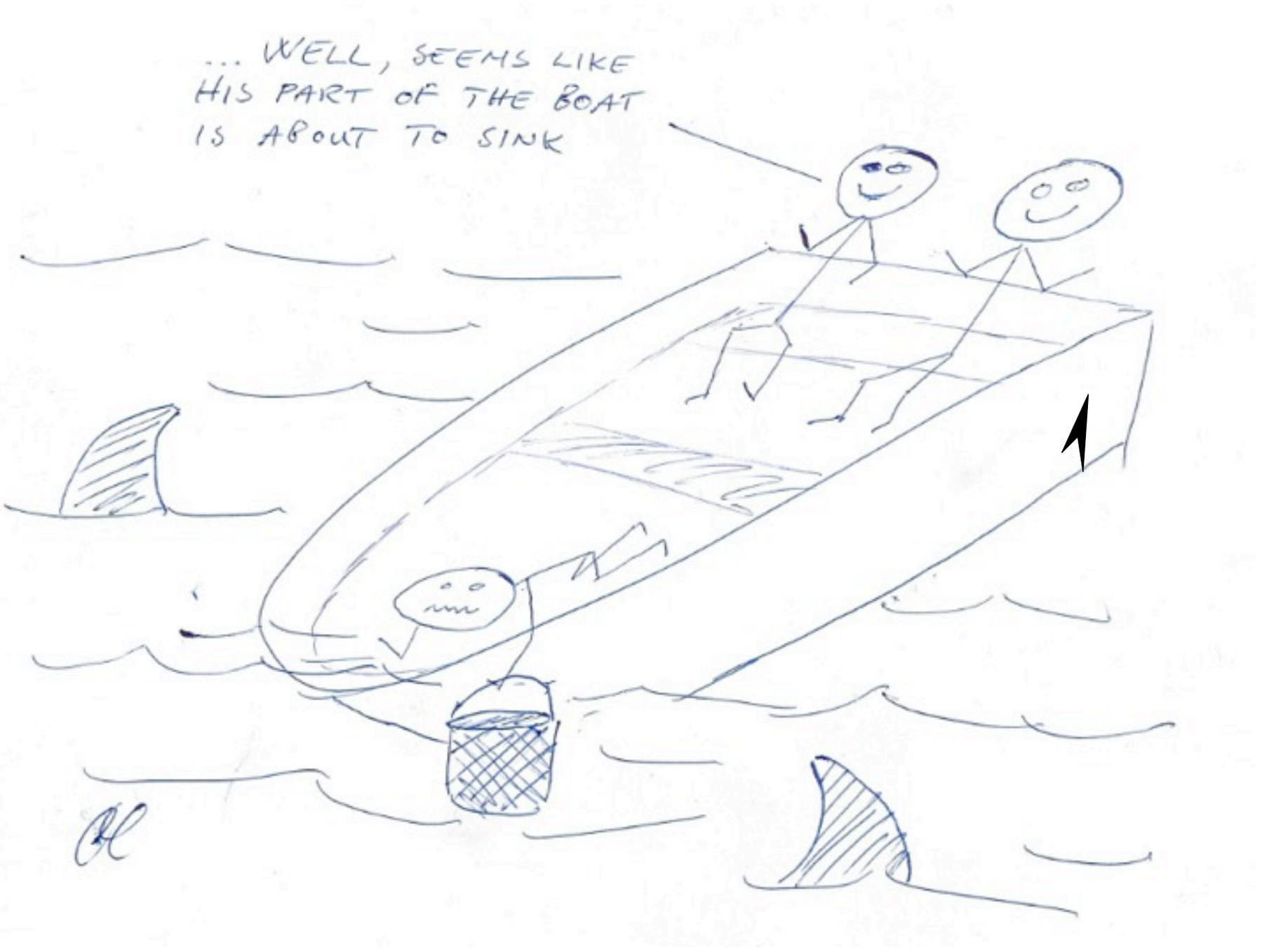
Ⓒ cross to the other side and go slightly left. there is a quarter ther all restaurant/tapas plac - FULL OF ENGLISH ? AMERICANS so be carefull go to the right places! Ar

Ⓔ

① in bar. be nice to barkeeper wy big moustache. Stand @ bar order a bottle of cider you will get free yummy Food as long as you drink.

with only a sketchy map as guidance

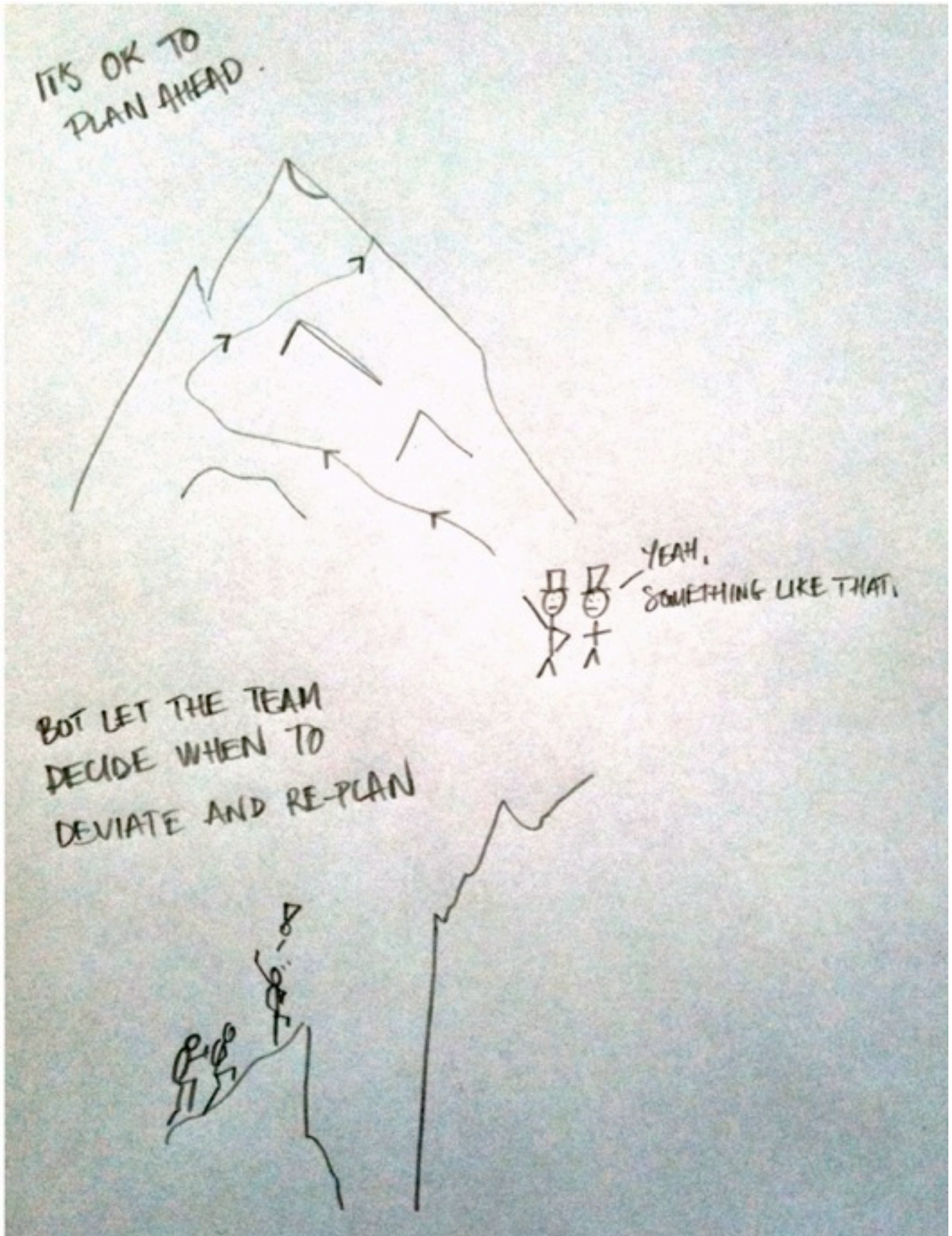
Make sure that everybody is working towards a common goal.



Focus on improving your skills of navigating chaos

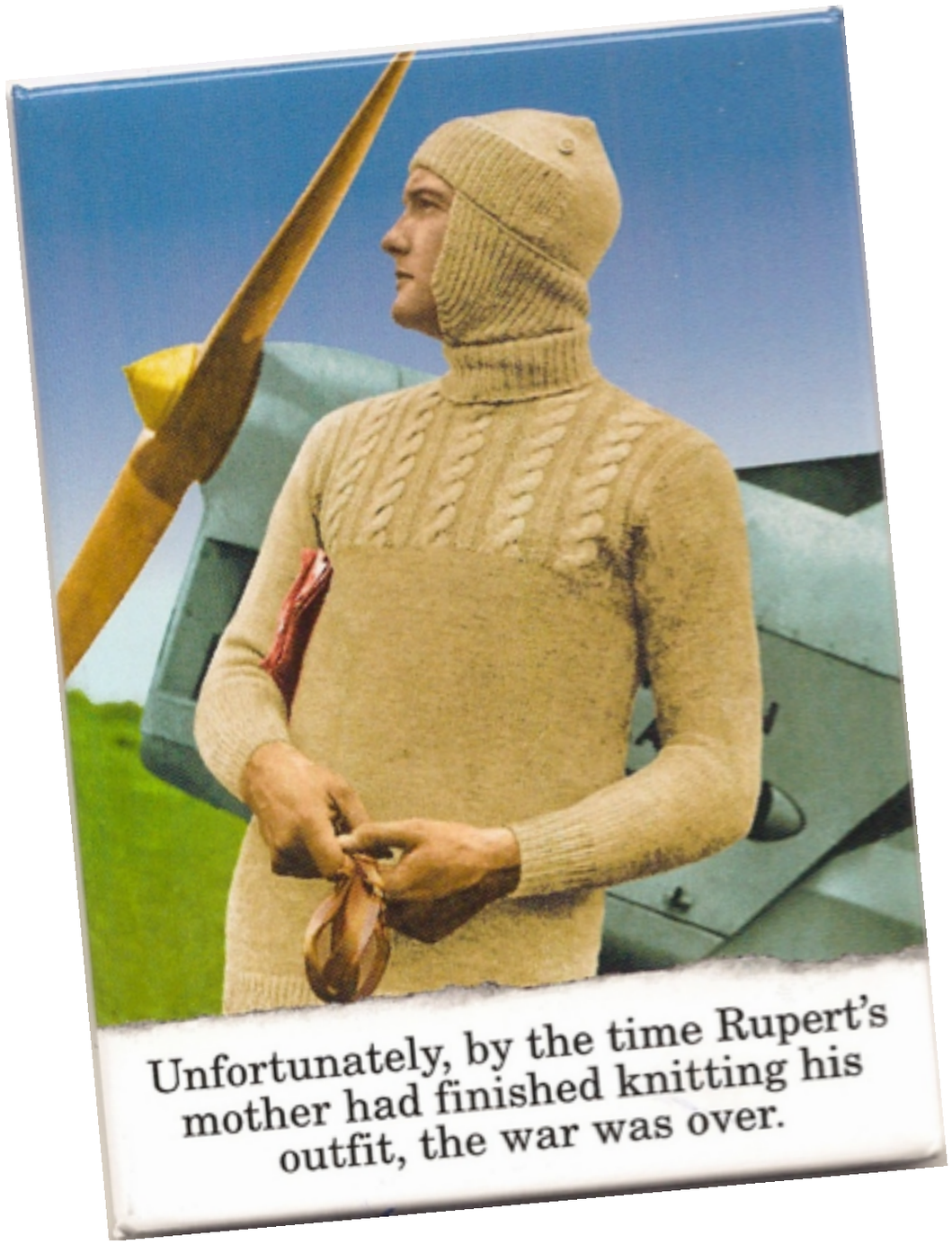
http://www.youtube.com/watch?v=oetF3UTIwbc

Plans are nothing, planning is everything



(geir berset)

Timing is important



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

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

(Princess Leia)

Appendix

Software development should be considered as a **continuous learning process** and a **cooperative game** of **communication** between **professionals**. Effective software development can only be achieved through **frequently repeating cycles** of **preparing**, **changing**, **observing**, reflecting, and **learning**.

[...] we have been designing complex systems whose active components are variable and highly non-linear components called people, without characterizing these components or their effect on the system being designed. Upon reflection, this seems absurd [...]

(Alistair Cockburn, 1999, about "traditional" software methodologies)

"What we don't do is treat our employees like they're all, you know, criminals,"

(Jenn Mann, SAS Institute)

"Controlling an organization by monitoring costs is like driving a car looking out the rear window."

(John Seddon)

Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity.

(General George Patton Jr)

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

(Princess Leia)


TANDBERG