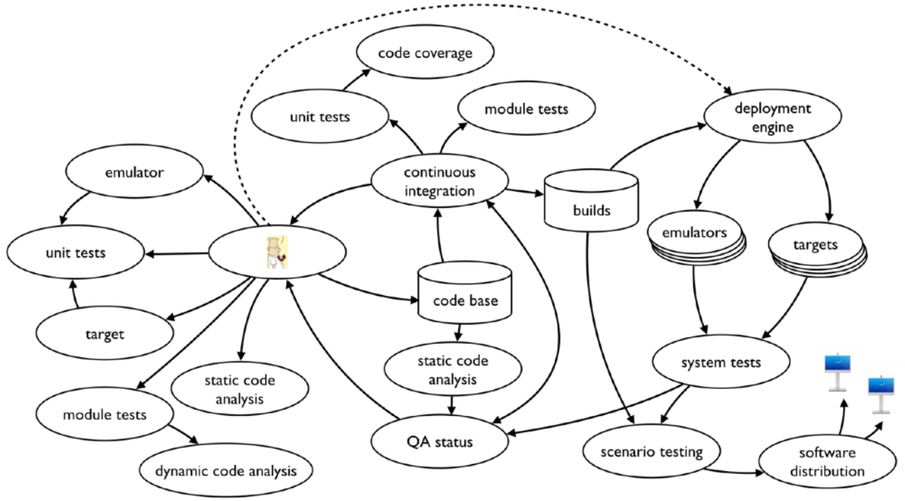
Feedback-driven Product Development

how we do it at Lysaker and how you can design your own system



Cisco's development and innovation centre in Norway (Lysaker) develops videoconferencing products, telepresence technology and collaboration solutions. This is embedded product development involving advanced mechanics, customised electronics, movable parts and millions of lines of software mostly written in C and C++. Over the last two decades we have gradually established a workflow that very much supports lean and agile product development for hundreds of engineers working closely together. A lot of effort goes into establishing effective feedback loops guiding the whole development process. We are not only talking about rapid feedback from build systems and continuous integration, but also from regression tests, advanced scenario testing and real users. The focus on establishing feedback loops goes beyond the product development workflow, it is a principle applicable to the whole organization.

This talk will present a concrete insight into the software development workflow that we are using today, before discussing what you need to consider if you want to set up an equally effective feedback-driven product development workflow in your organization. The talk is relevant for everyone involved product development where software is a key component.

Olve Maudal, Cisco Systems Norway

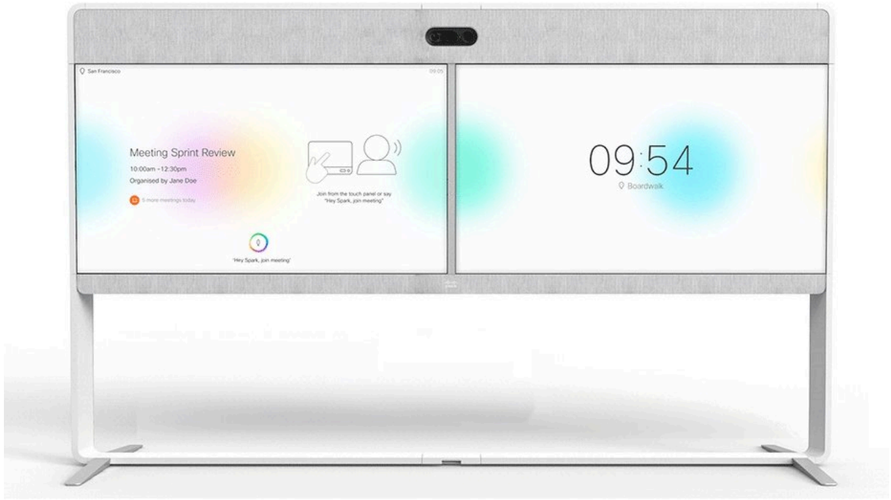
a 60 minute keynote at Equinor Developer Conference, Sola, Norway, September 26, 2018

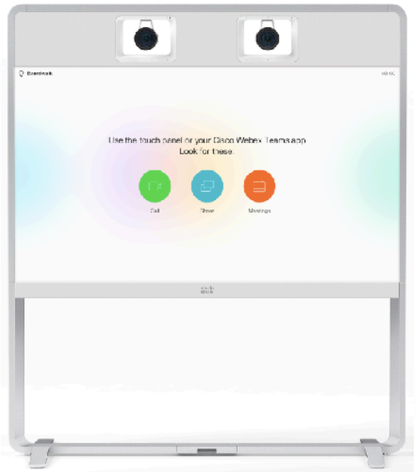
Cisco Systems, Innovation Center
Lysaker, Norway

Telepresence

Telepresence

Some of the stuff we develop at Lysaker









Hi Mona
Tap the touch panel
or start an activity from Spark

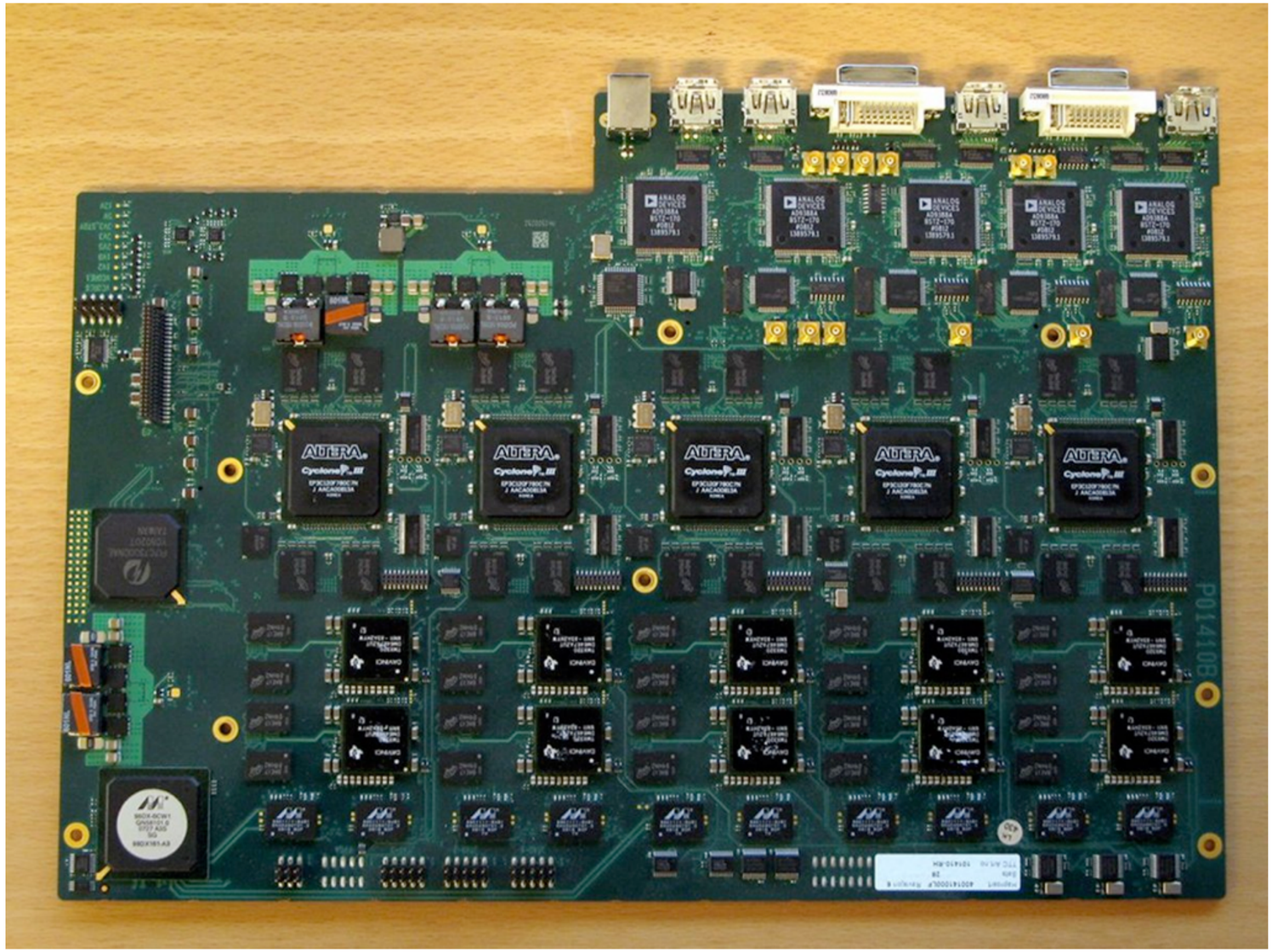




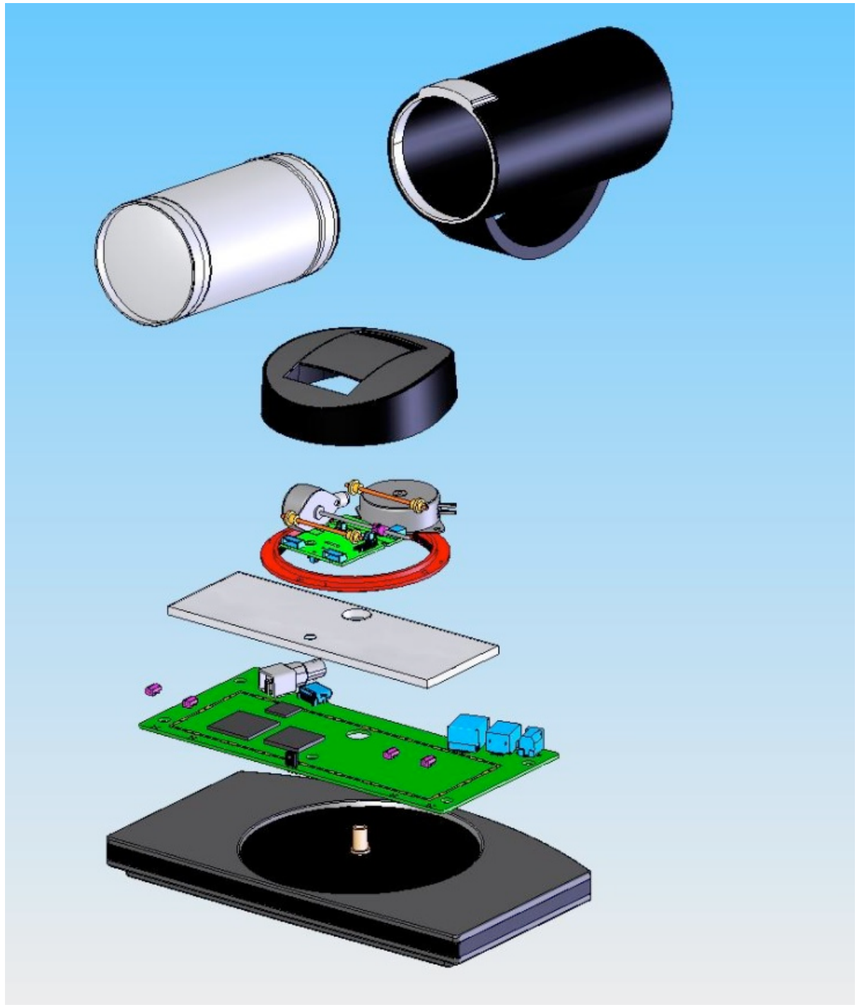
at Lysaker we are ~350 engineers

we do...

Electronics / Hardware



Mechanics



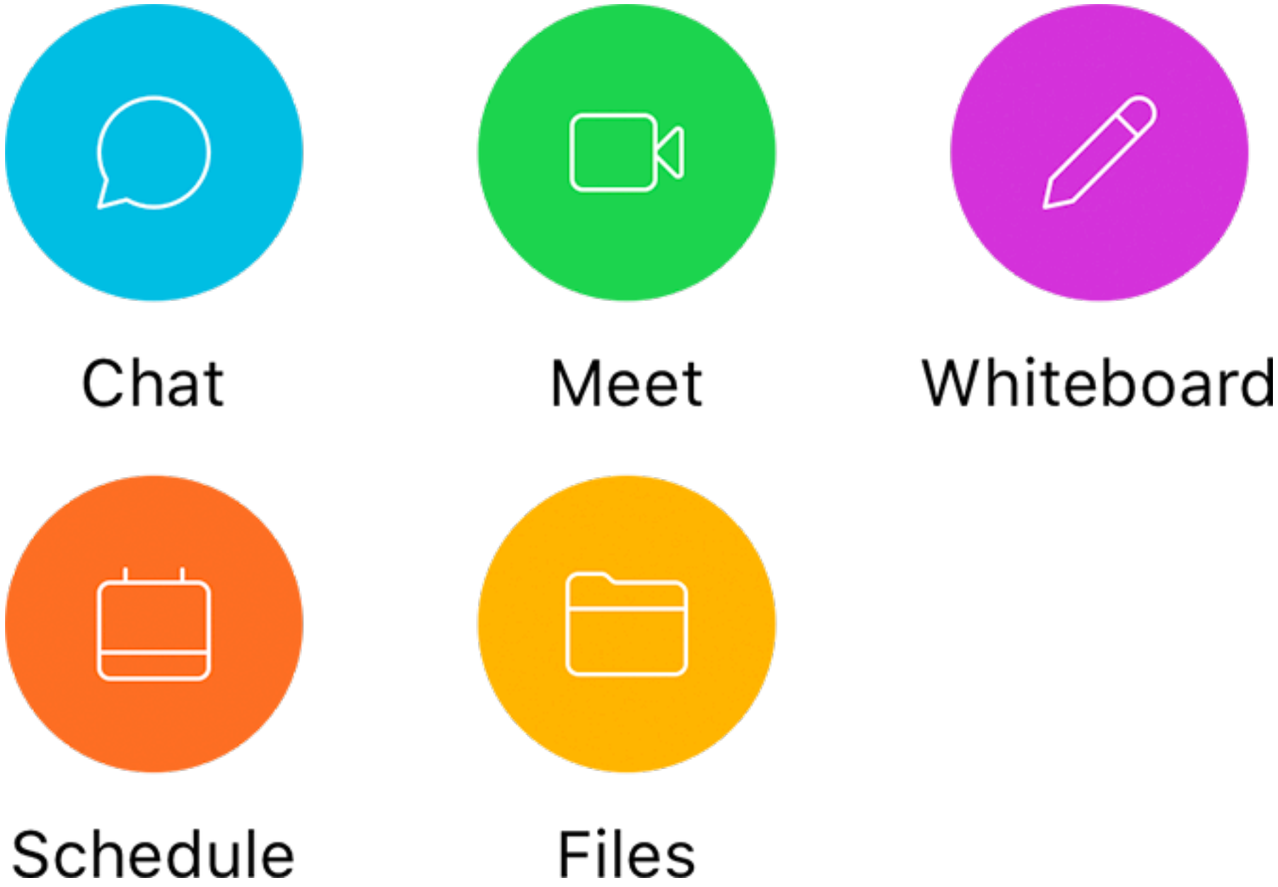
Industrial Design

1992

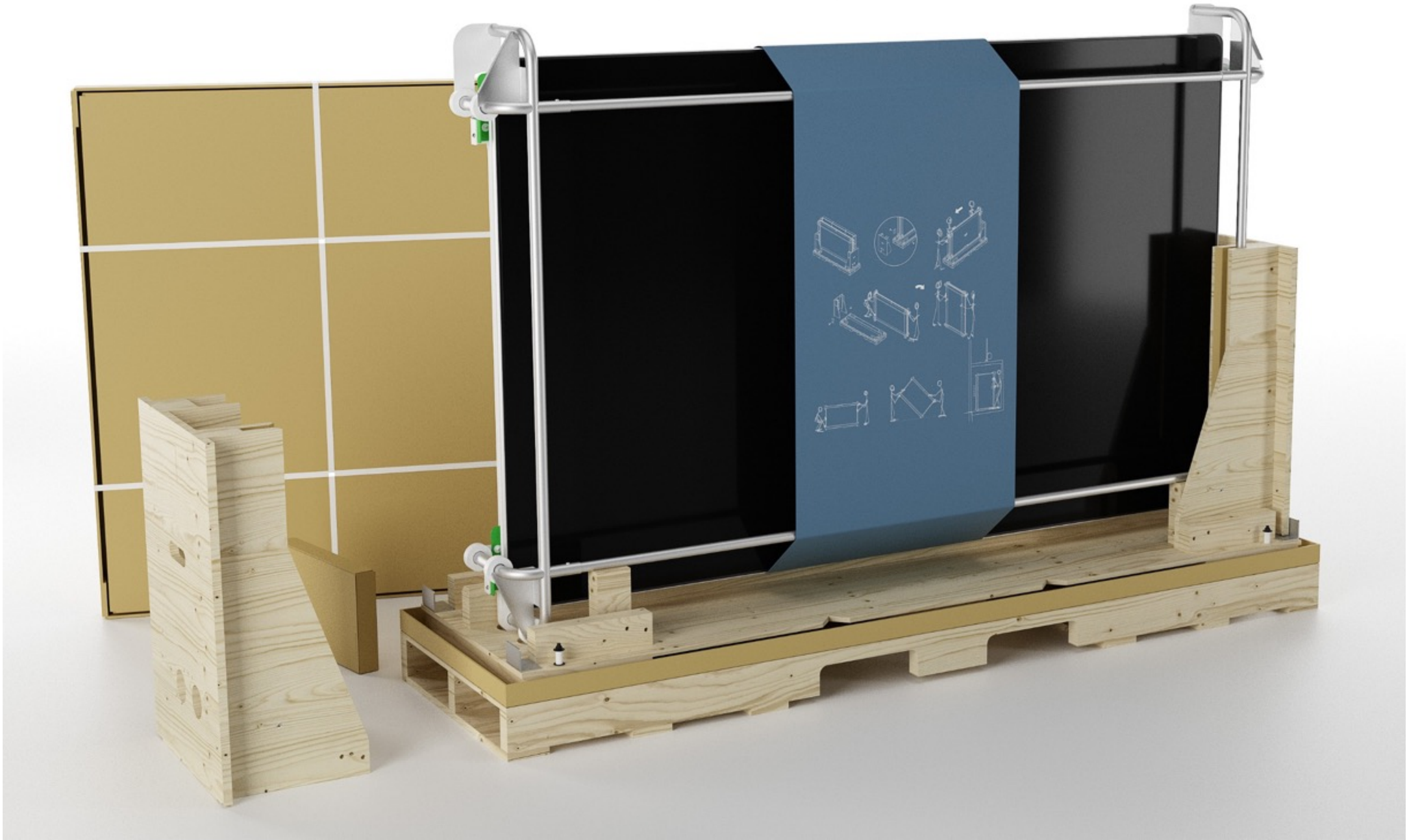
2015

reddot award 2015
winner

User Experience Design

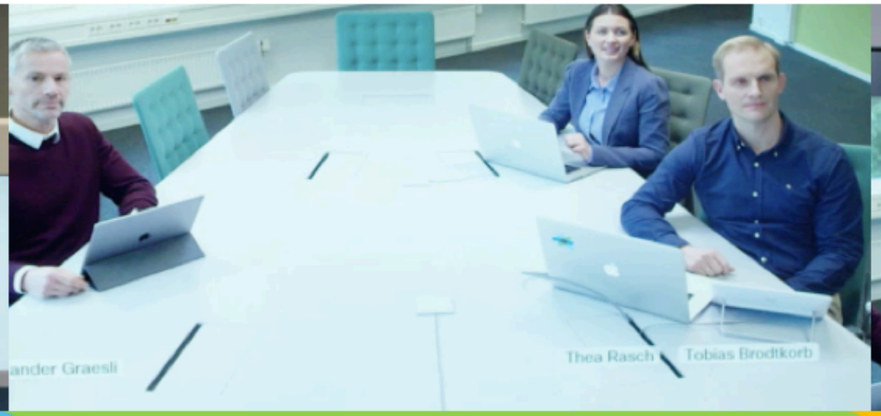


Unboxing and Logistics

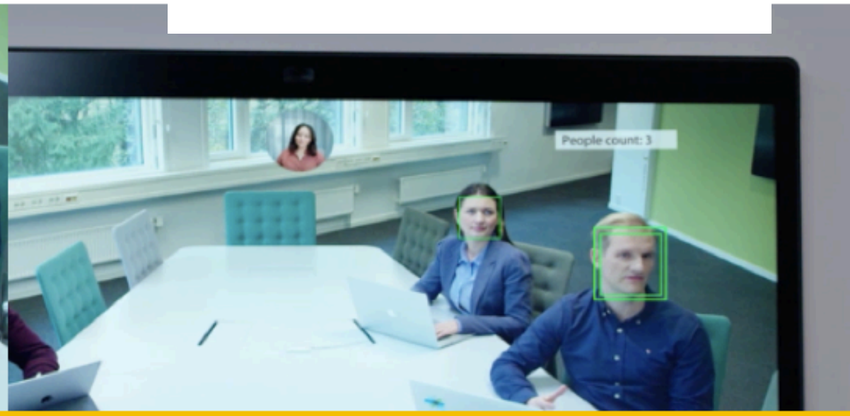


Machine Learning and Artificial Intelligene

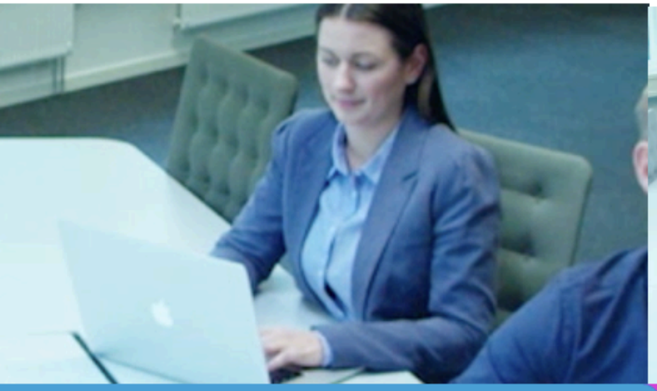
Webex Assistant (June, Teams)



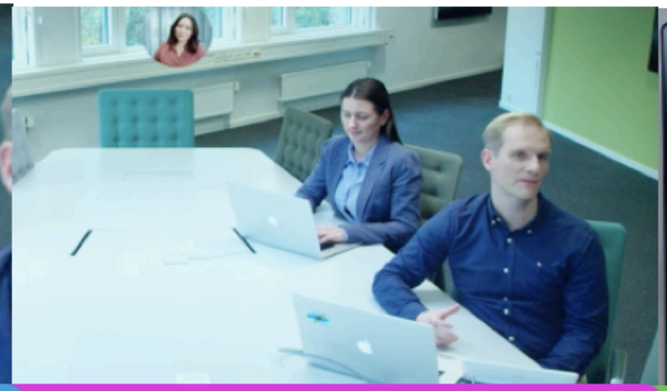
Face recognition* (Teams) only)



In-room analytics

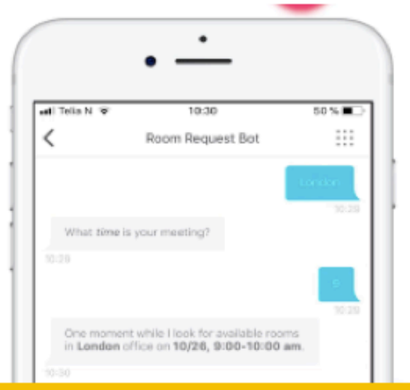


Noise suppression*



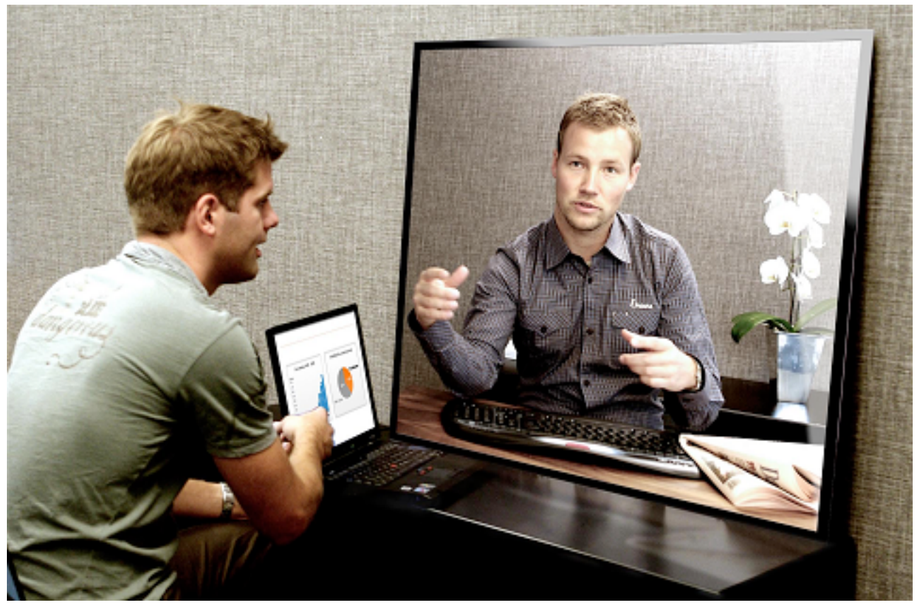
Speaker tracking

Best view



Bots (Teams)

Looking into

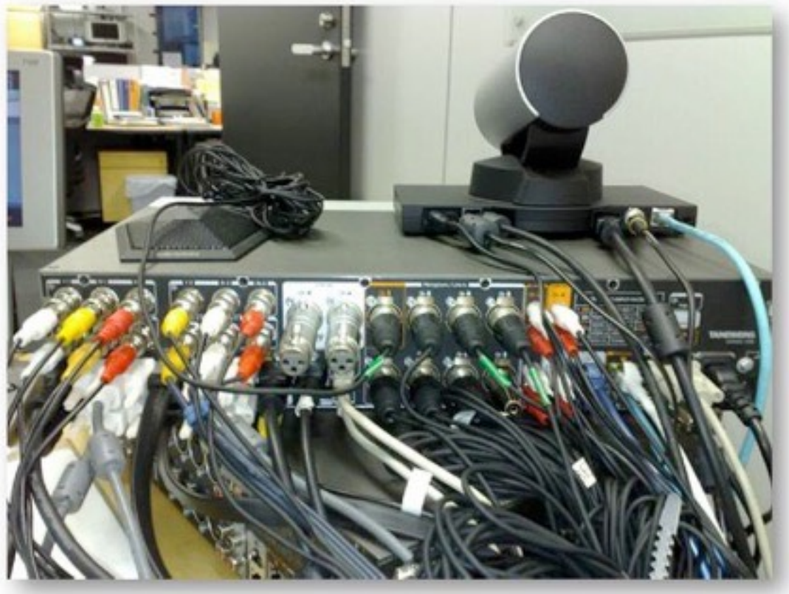


the future

but, by far, most of us work with

software developement

At Lysaker we have been developing telepresence products and collaboration solutions for more than two decades (since ~1991)



"... an organization that develops spectacular products and outperforms all competitors"

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- Effective feedback loops
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

The secret sauce

http://www.seriouseats.com/recipes/images/20101007-pizza-lab-sauce.jpg

The most important ingrediences

- **Effective feedback loops**
- Slack
- Professionalism
- Focus on value
- Systems thinking
- Transparency
- Release early, release often

Facts about advanced product development

Few high tech projects are like running
down on a paved road where you can see the ...

... goal in the end of the road.

Most projects are more like...

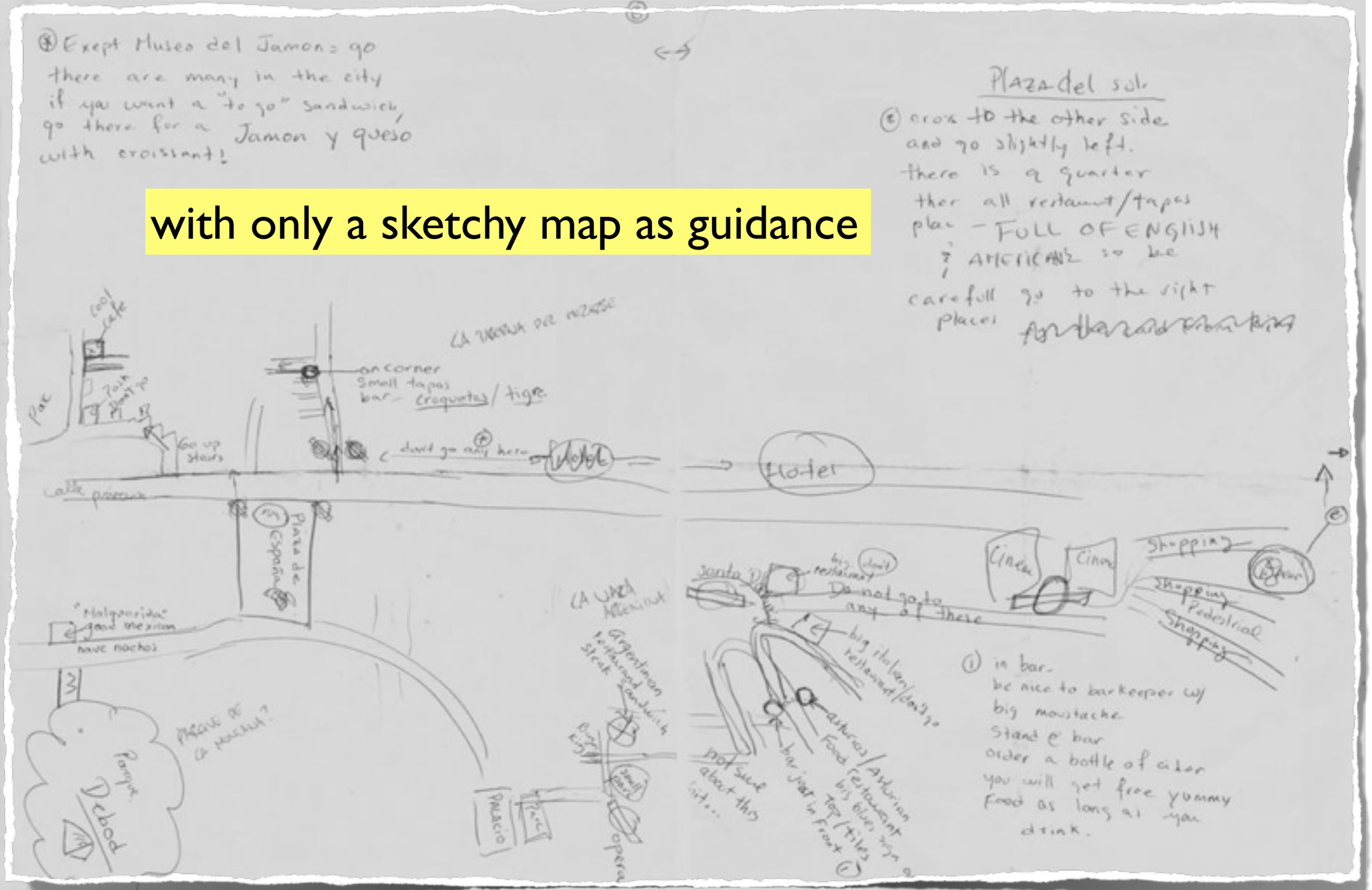

extreme orienteering


in impossible terrain


with a group of people


in the dark

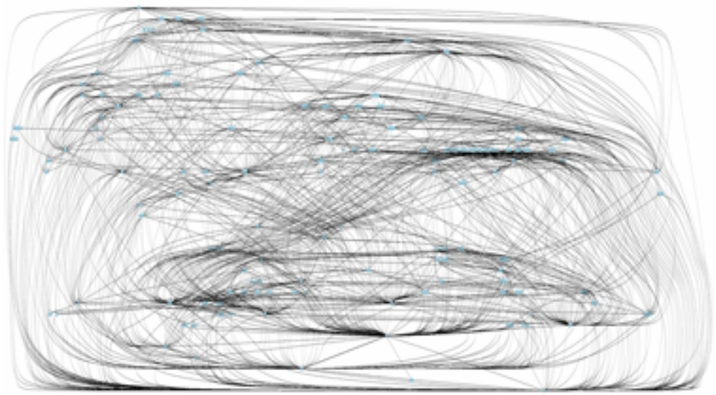
with only a sketchy map as guidance



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

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

The main codebase



- embedded software development
- about 200 active software developers
- typically more than 100 commits per day
- 4-5 million lines of code, mostly C and C++
- visible traces back to the late 1980's
- ~20 products, ~50 builds

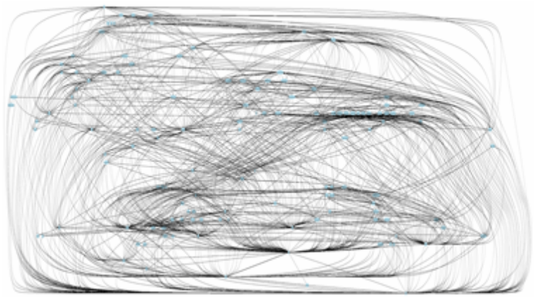
1987

1992

1997

2000

2003



2004

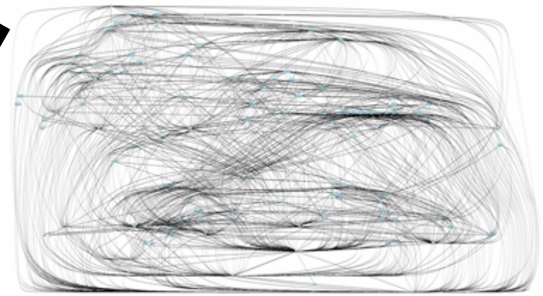
2006

2006

2005

2010

2008



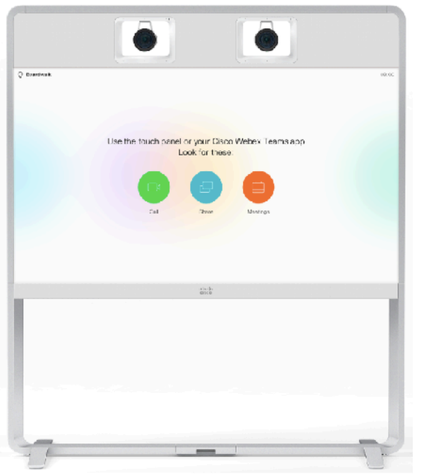
2009

2009

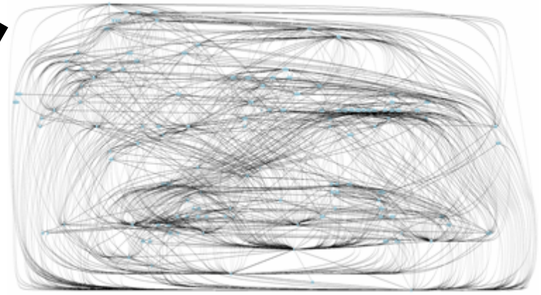
2008

2013

2014



2014



2019

2013

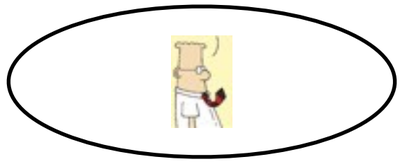
2017

2018

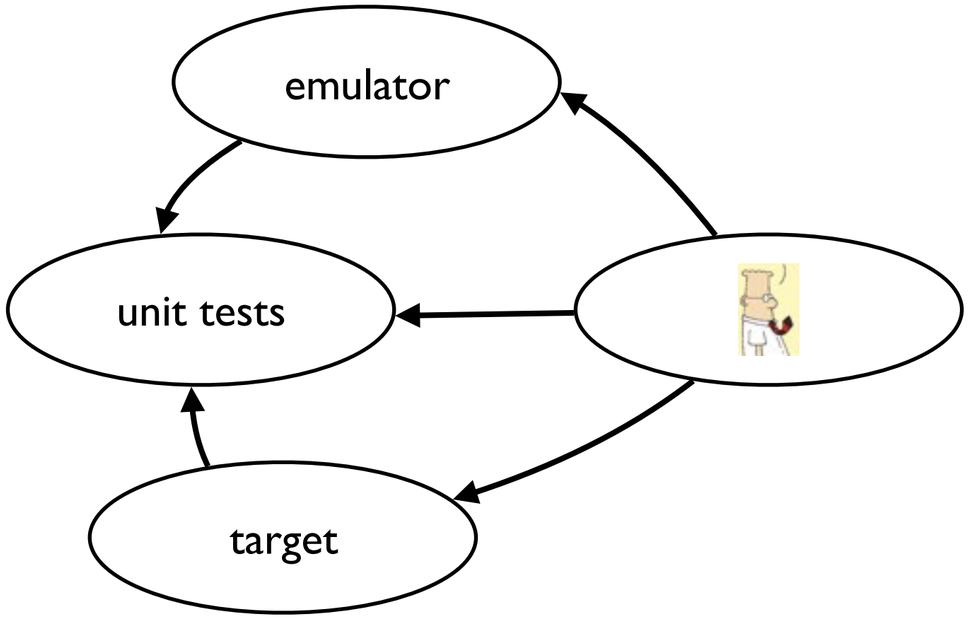
2016

software development workflow

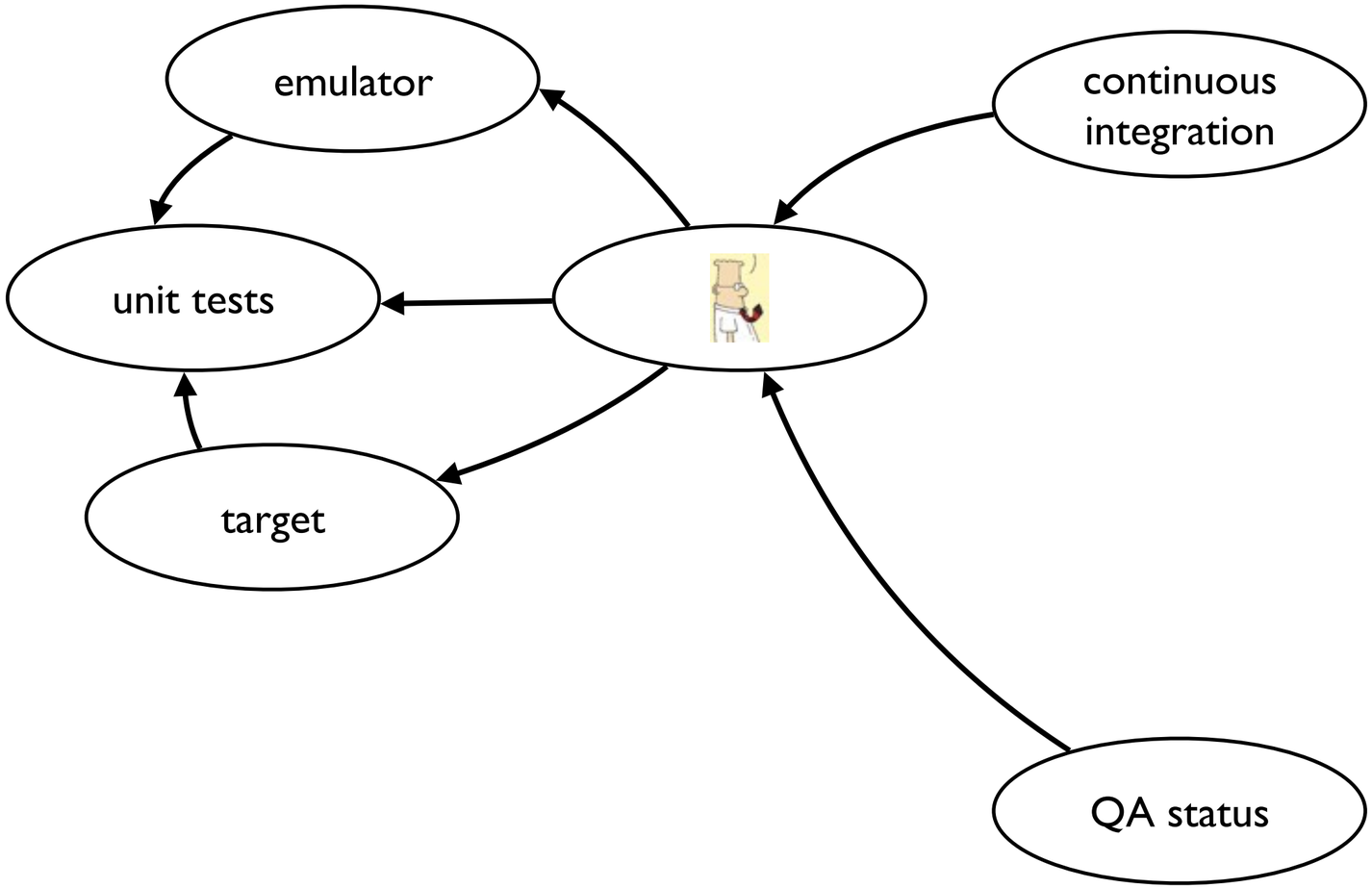
Continuous integration and deployment system



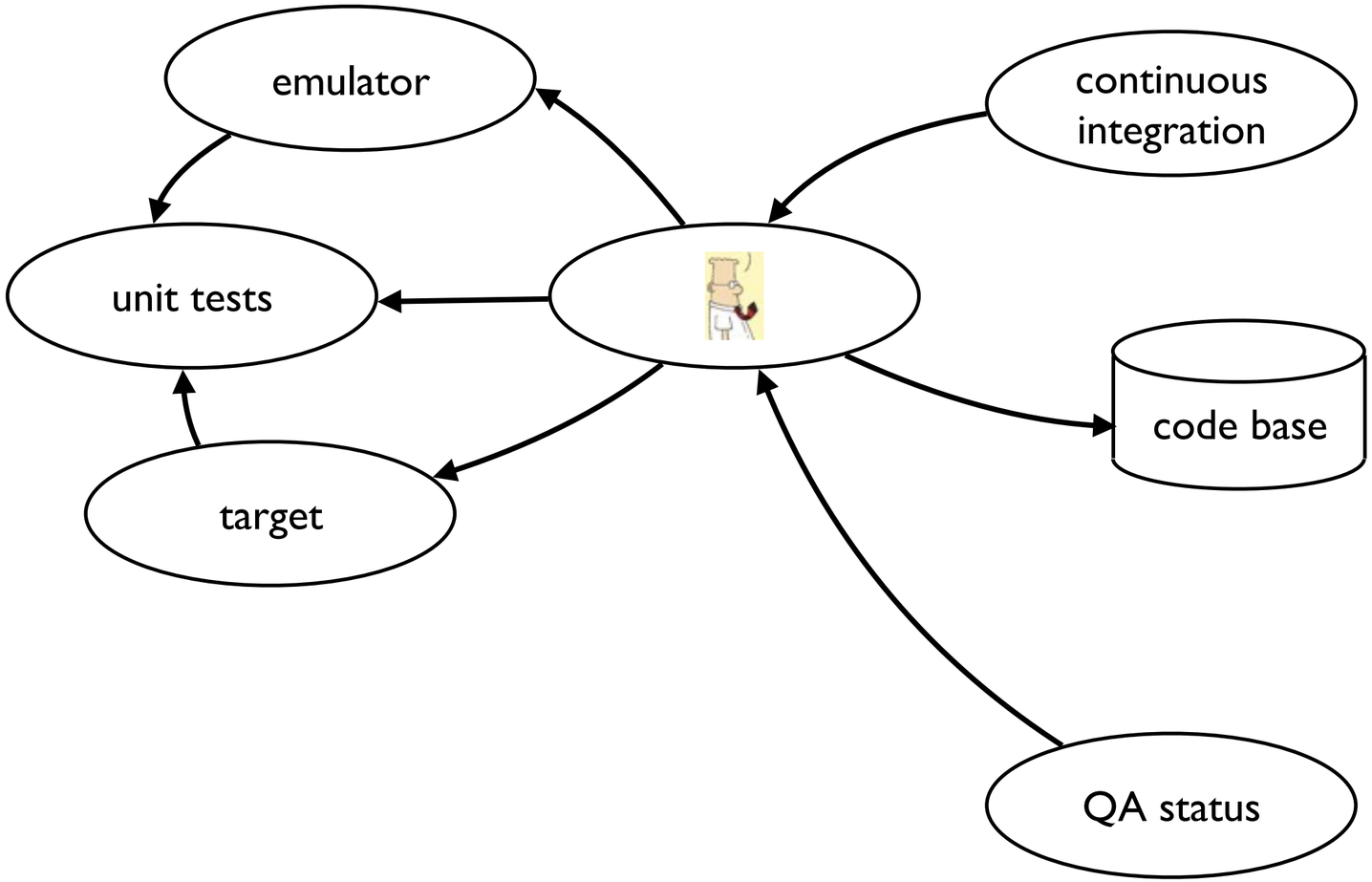
Continuous integration and deployment system



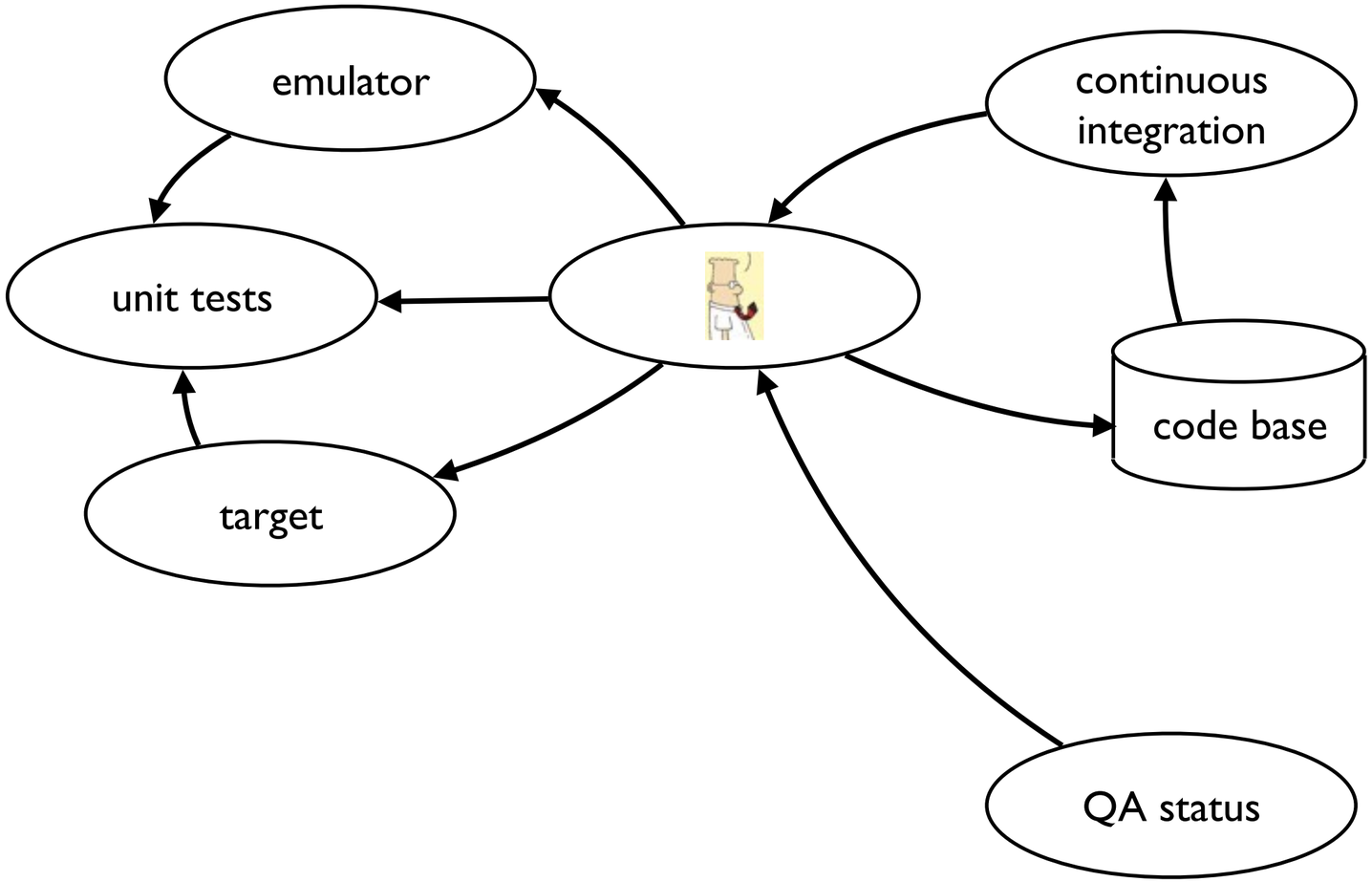
Continuous integration and deployment system



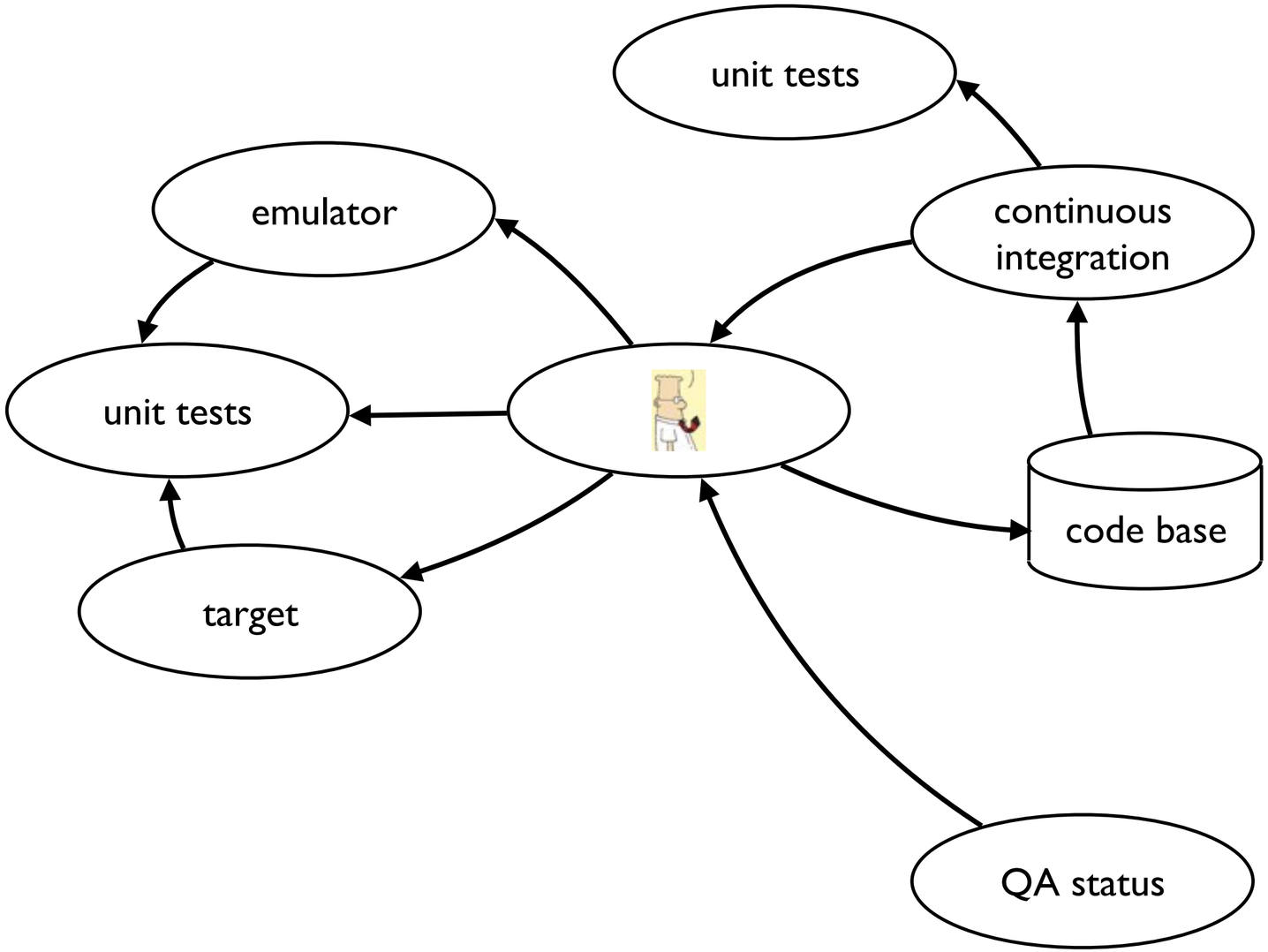
Continuous integration and deployment system



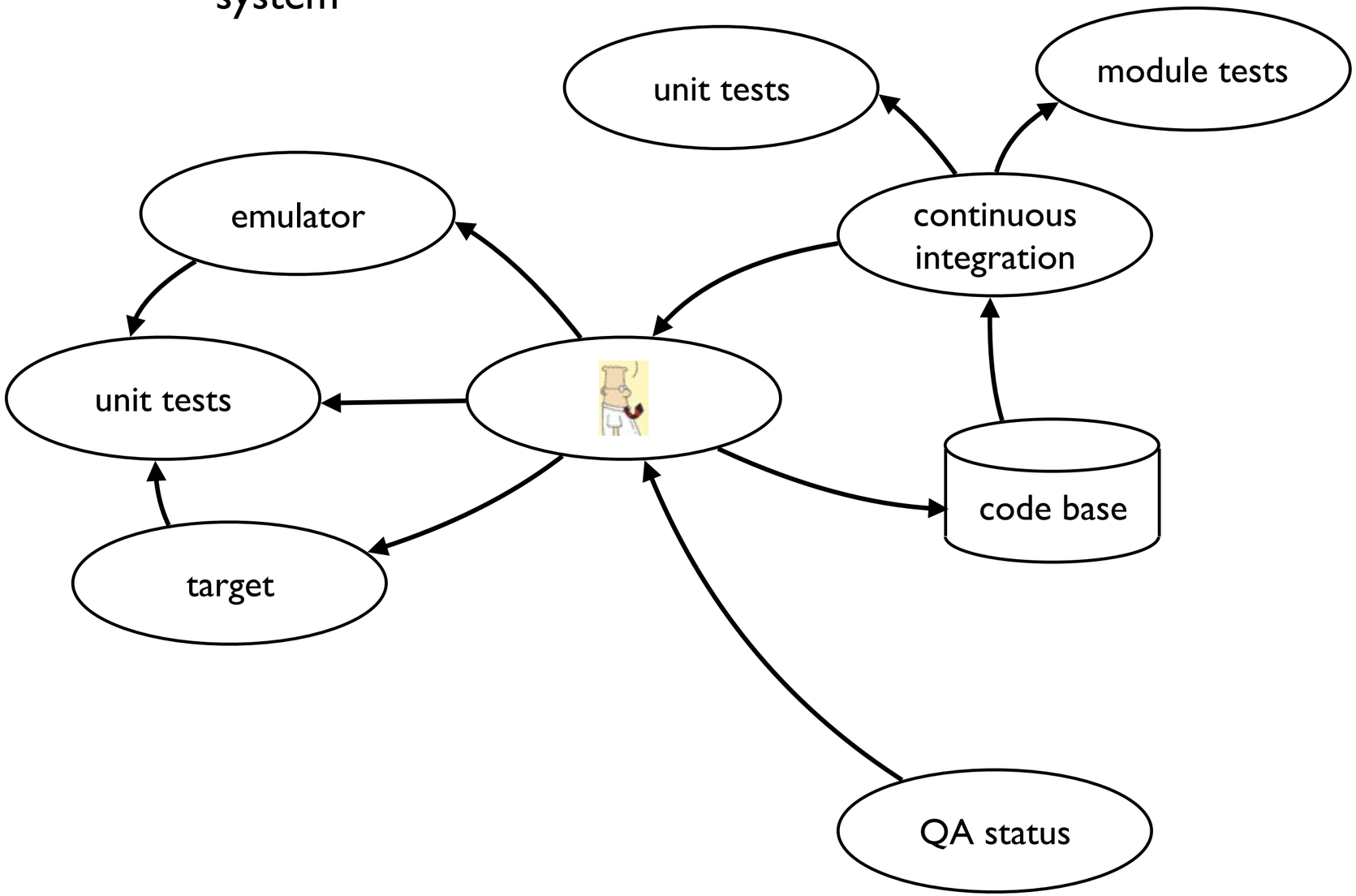
Continuous integration and deployment system



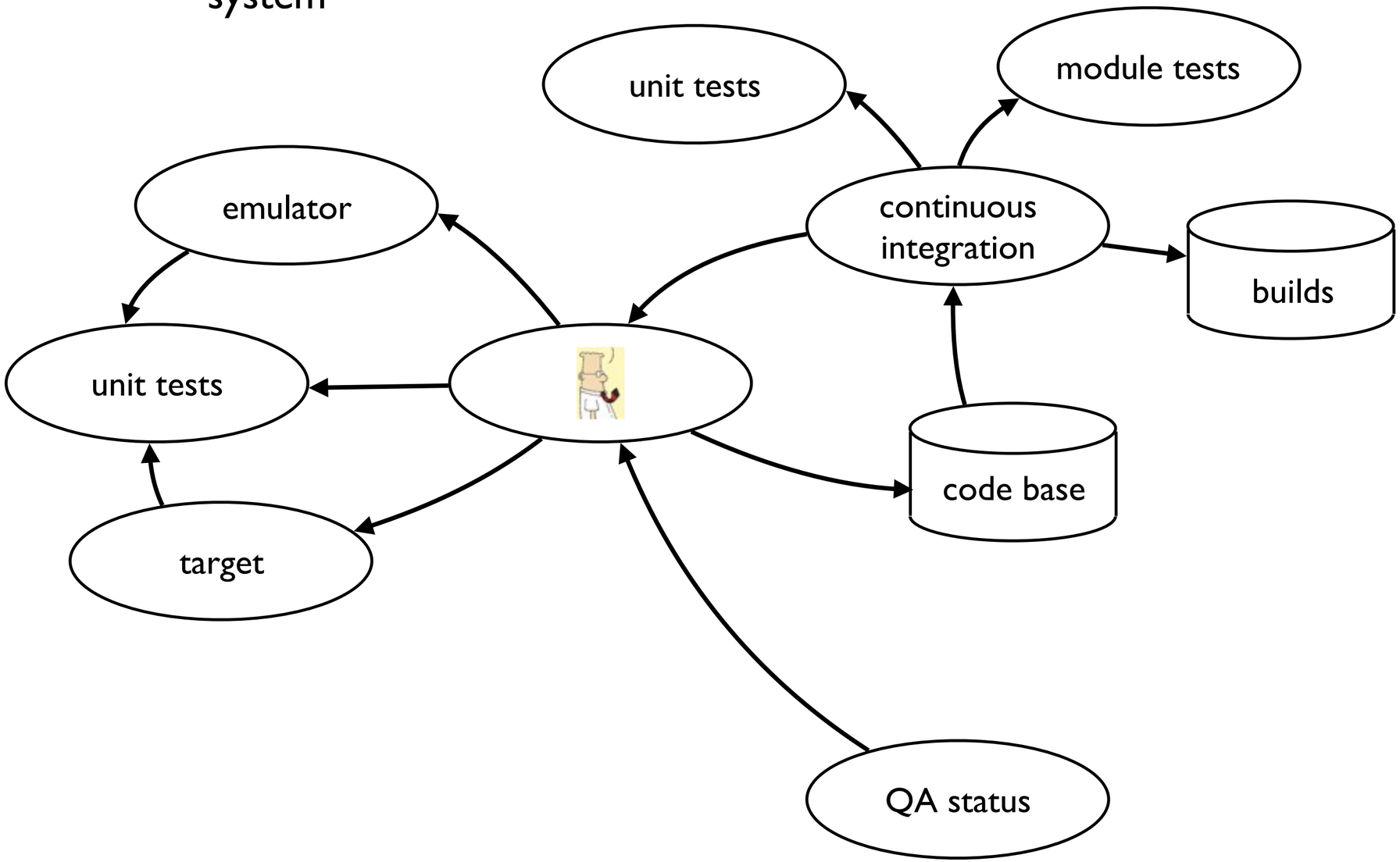
Continuous integration and deployment system



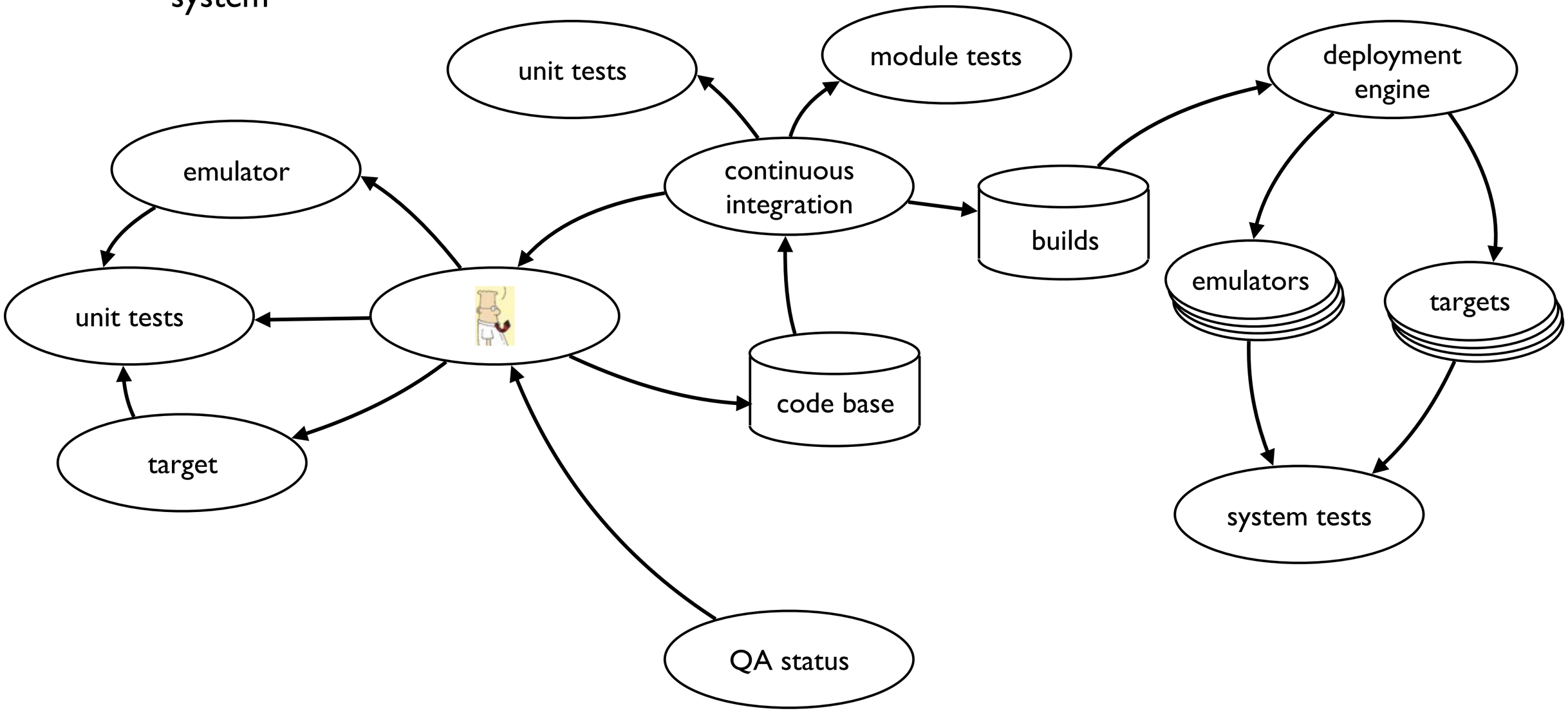
Continuous integration and deployment system



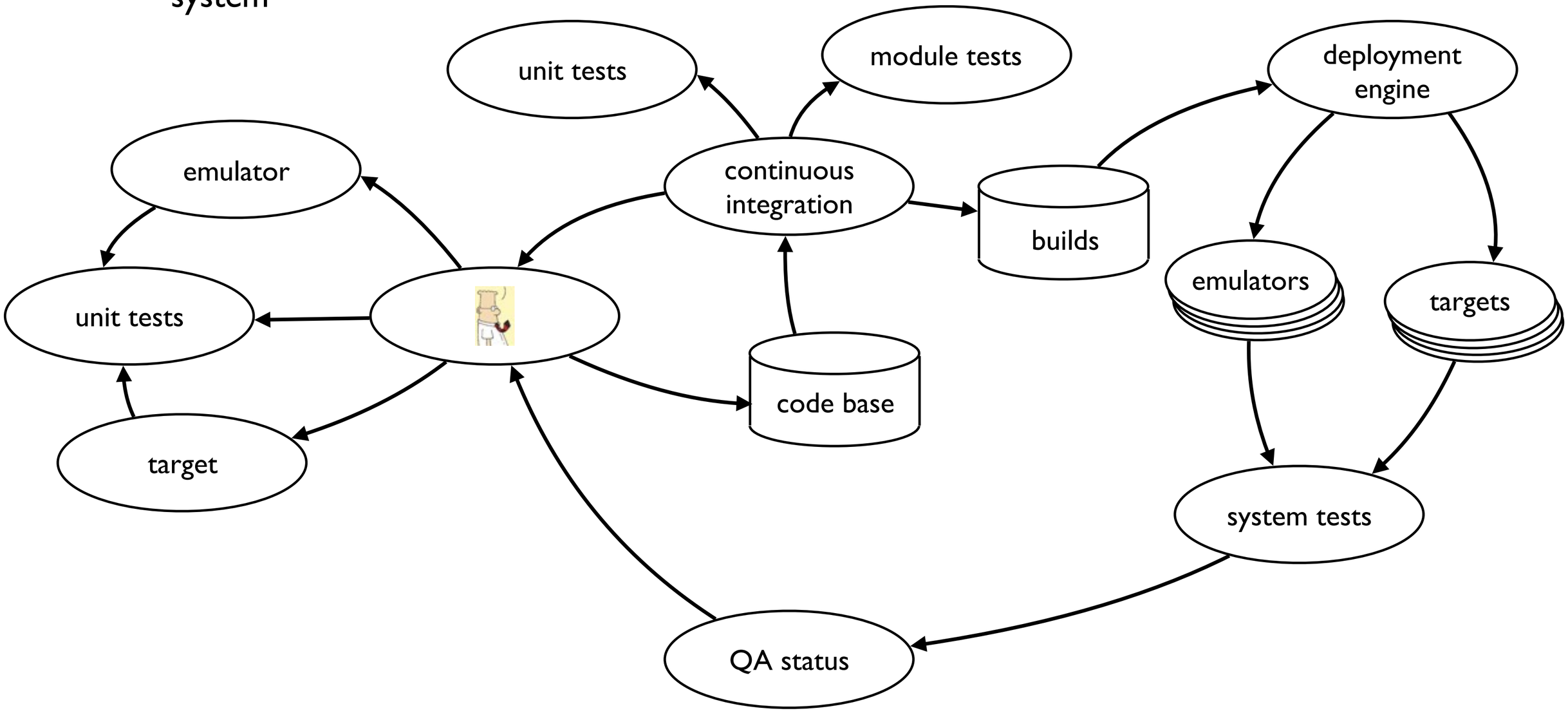
Continuous integration and deployment system



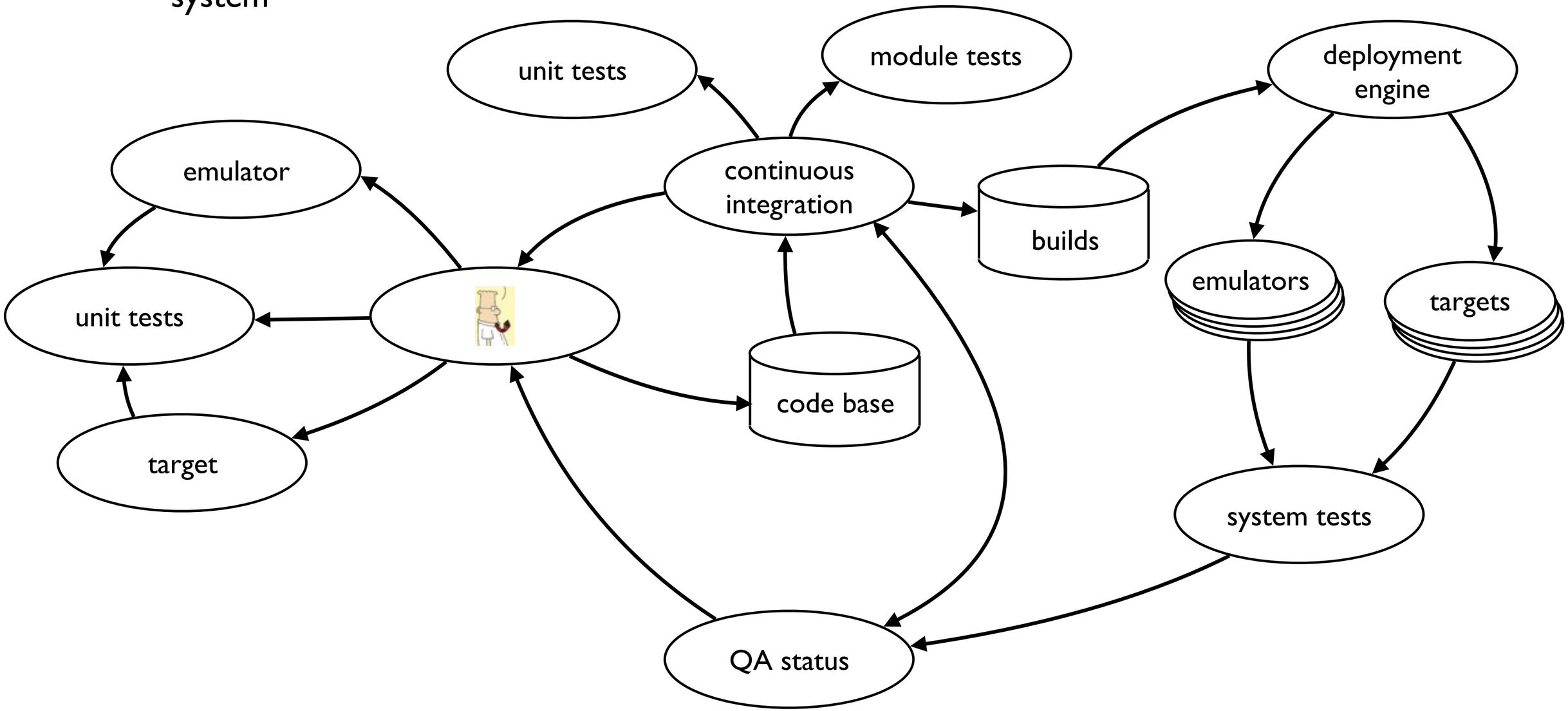
Continuous integration and deployment system



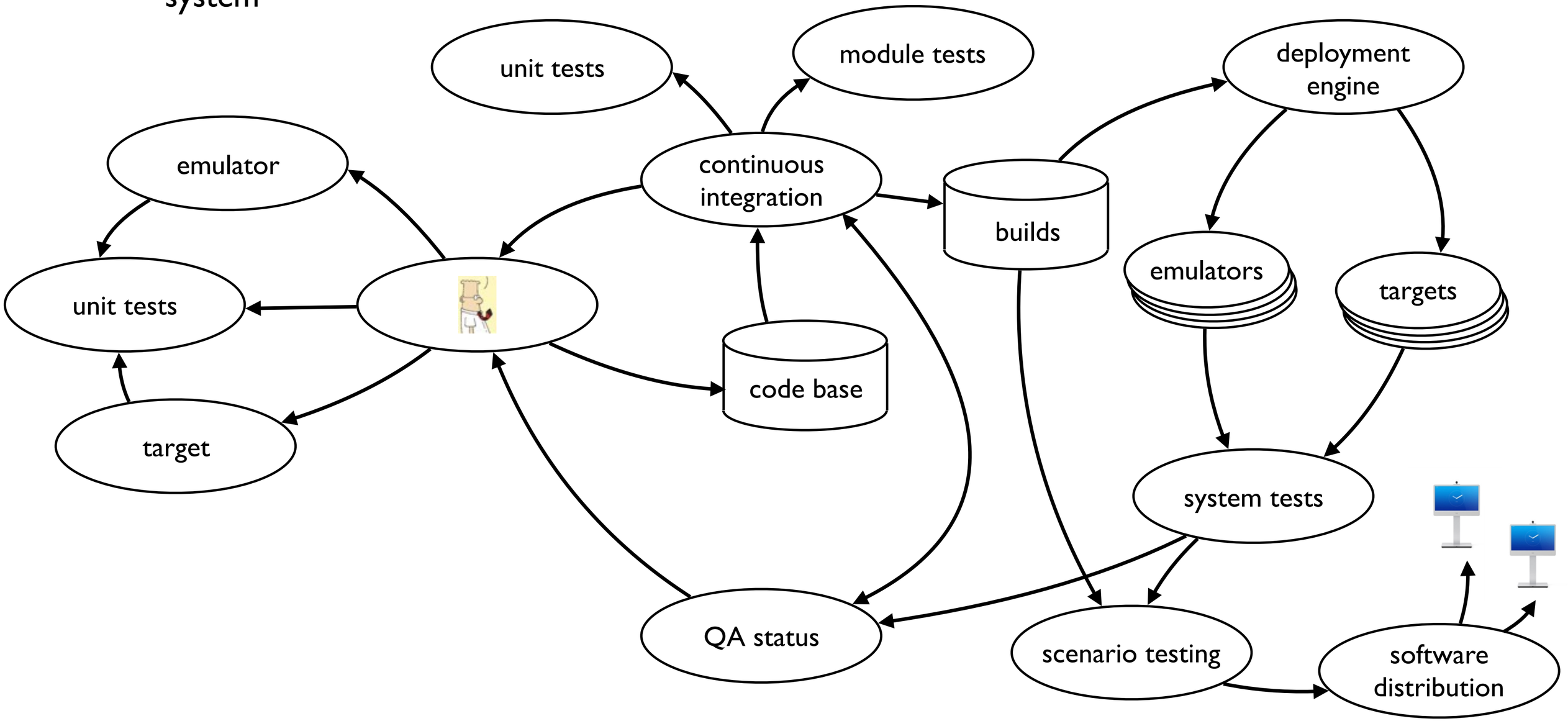
Continuous integration and deployment system



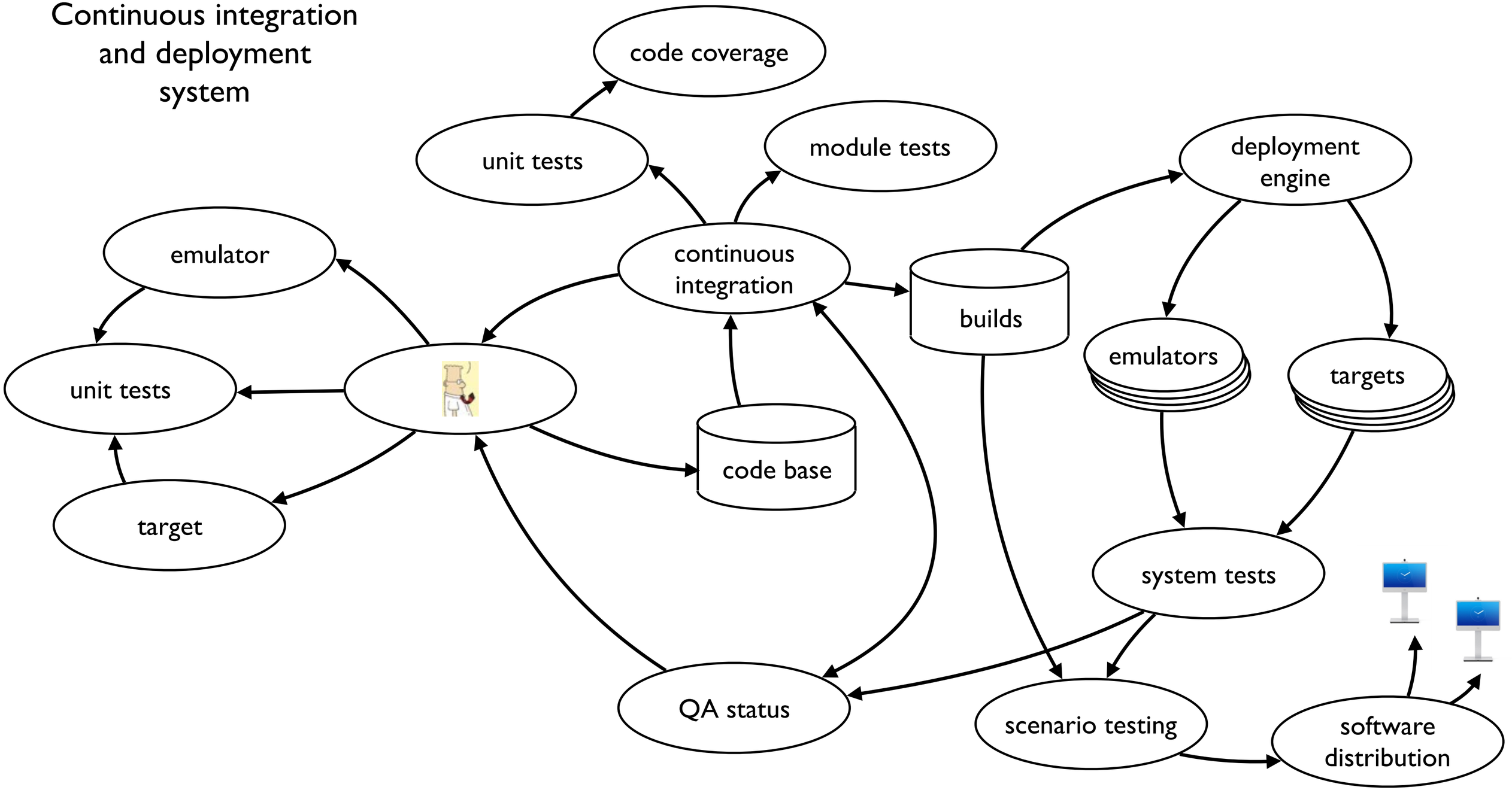
Continuous integration and deployment system



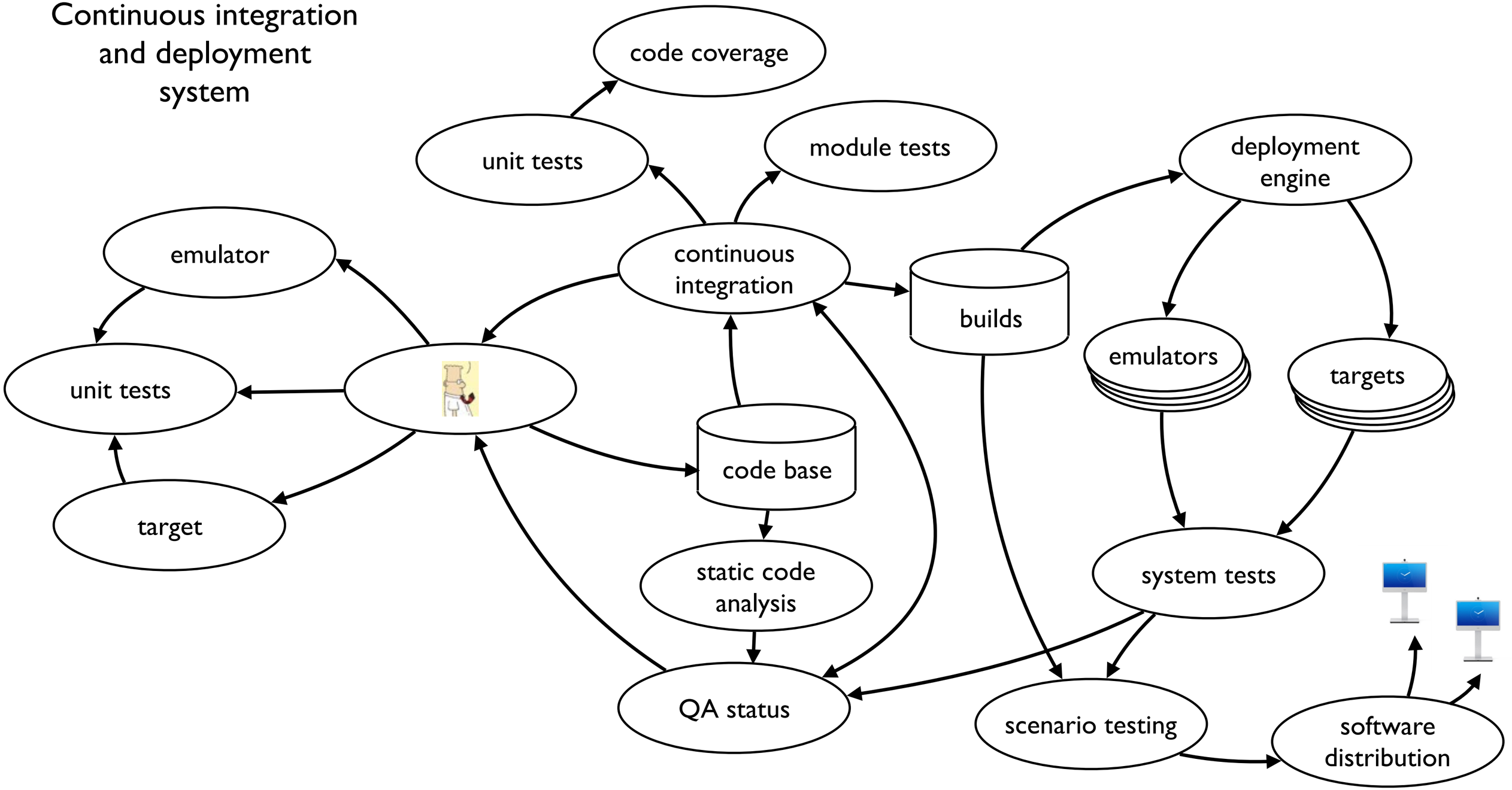
Continuous integration and deployment system



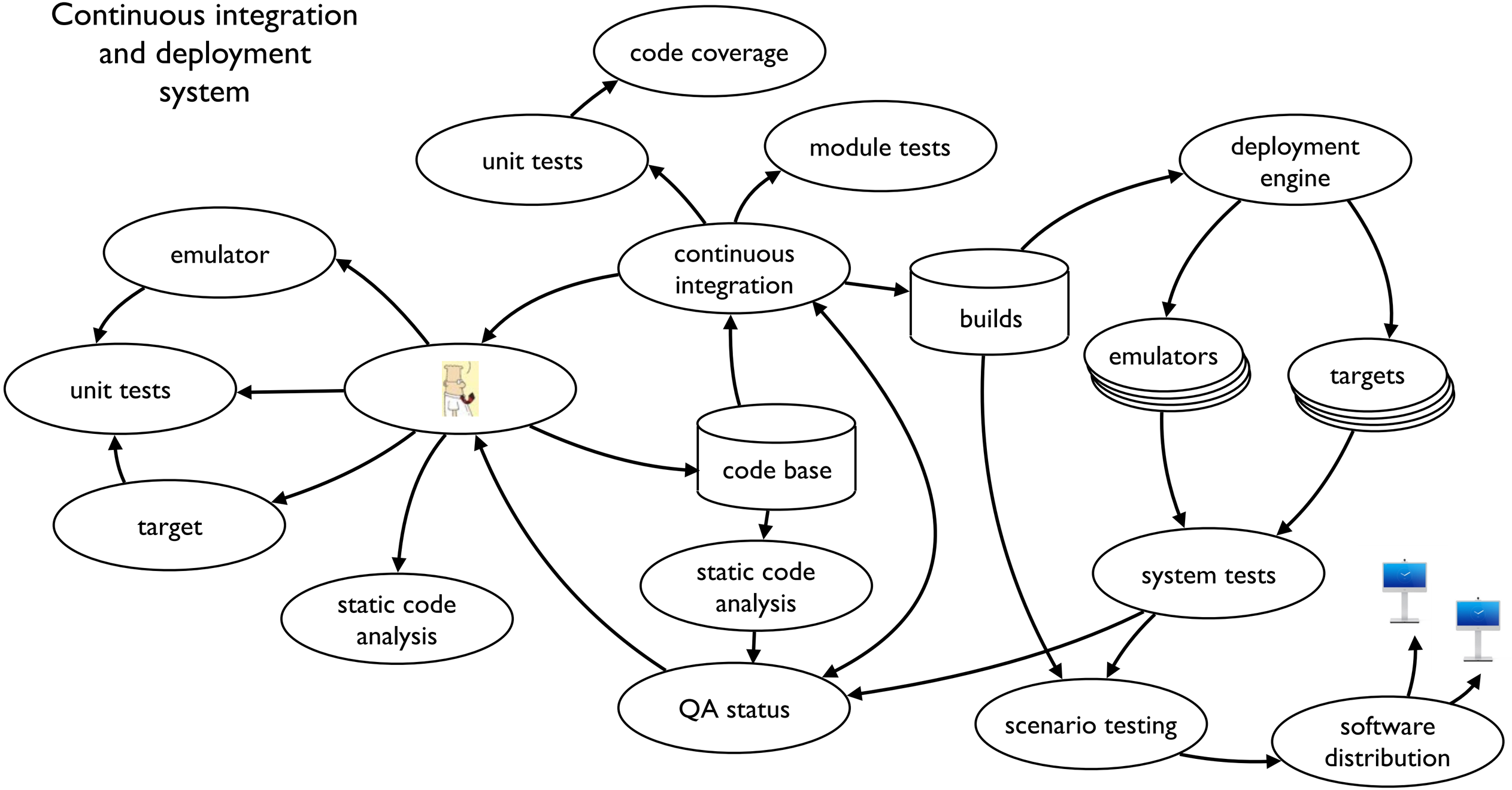
Continuous integration and deployment system



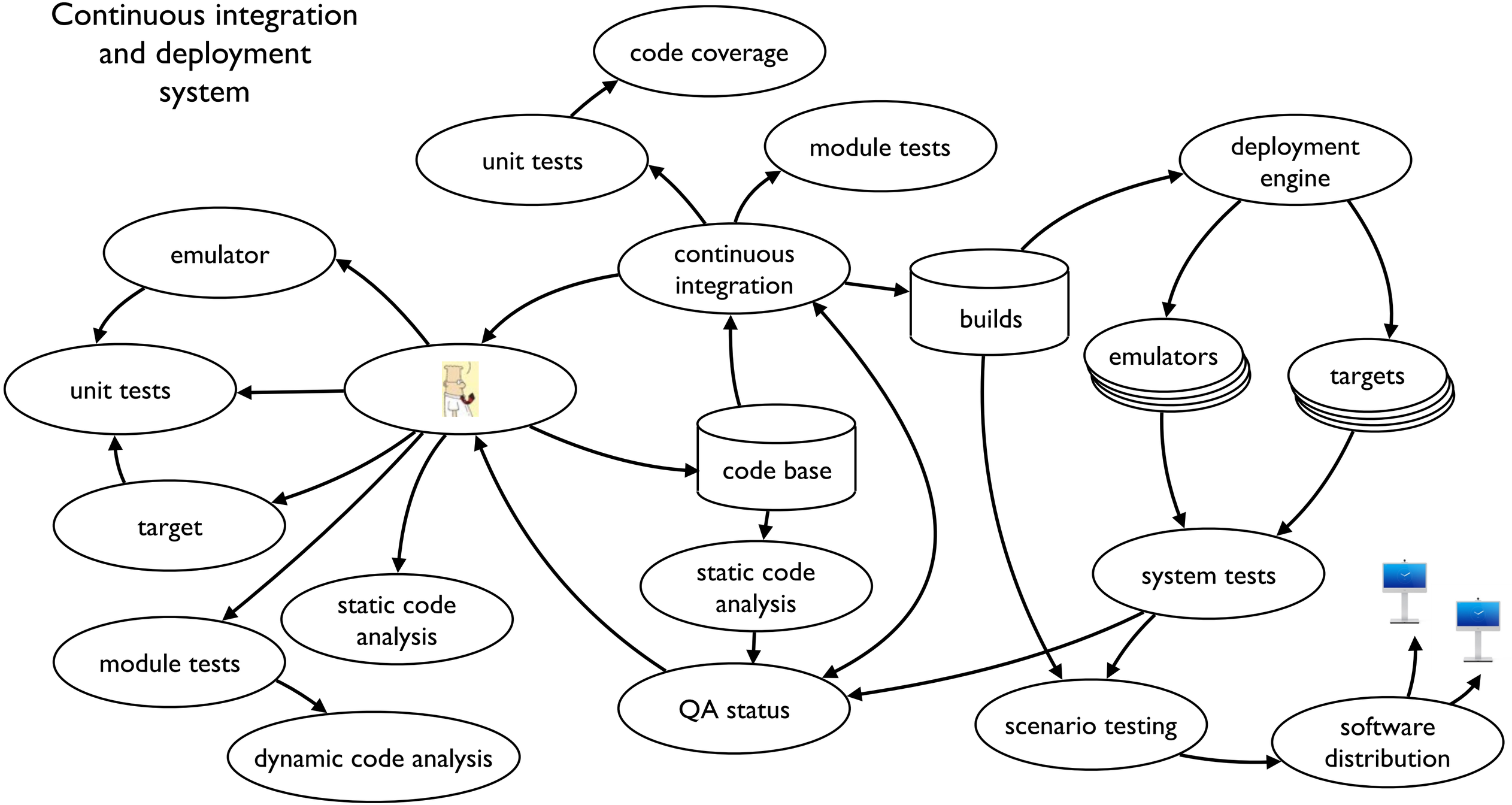
Continuous integration and deployment system



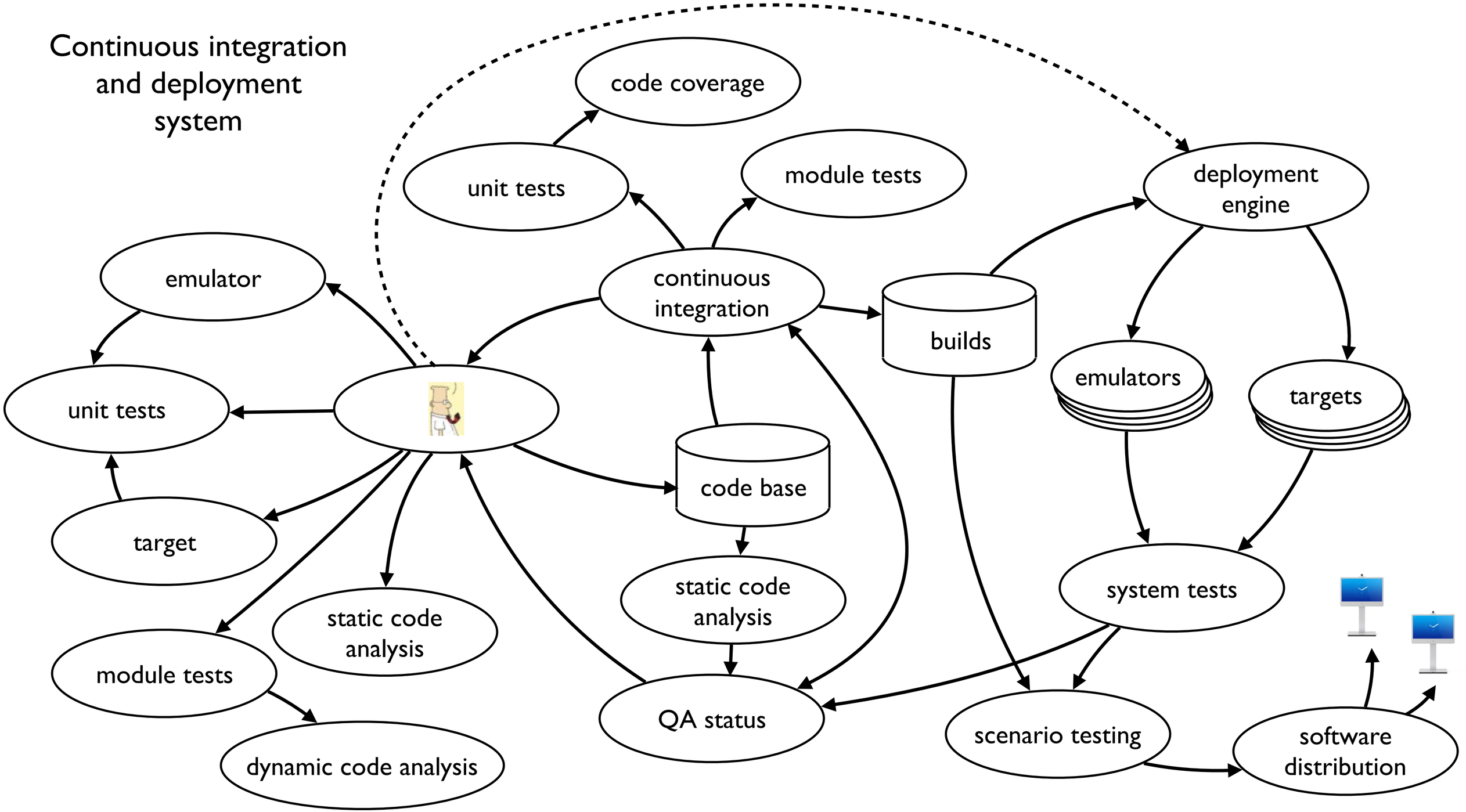
Continuous integration and deployment system



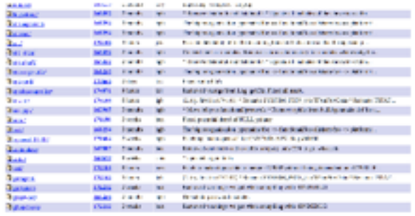
Continuous integration and deployment system



Continuous integration and deployment system



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



viewvc

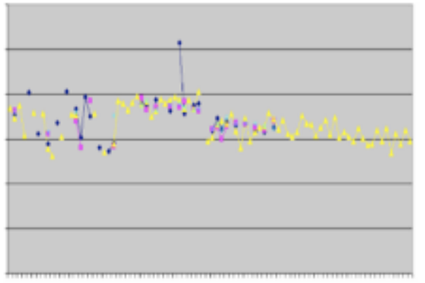
diff from viewvc

audio delay trend

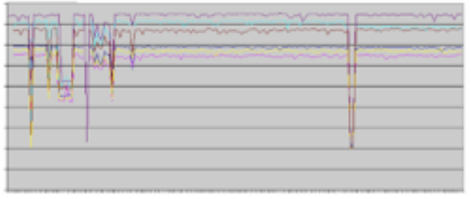
bugzilla

irc channel

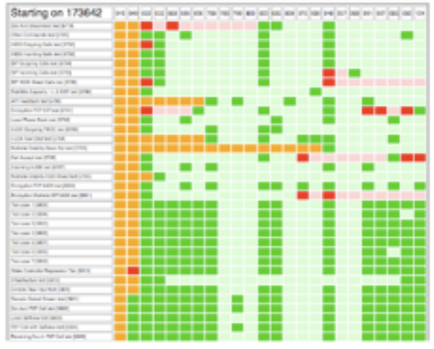
continuous integration



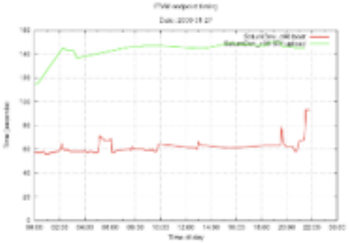
H.264 delay trend



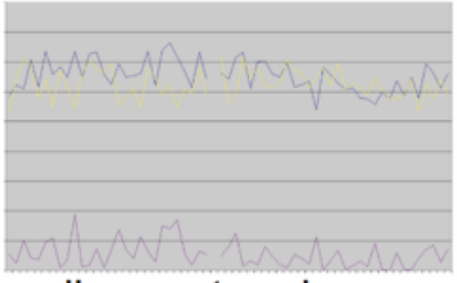
PESQ trend



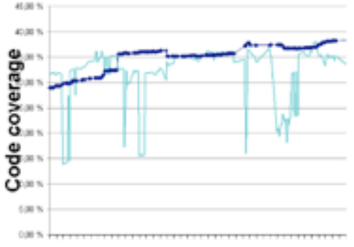
system tests

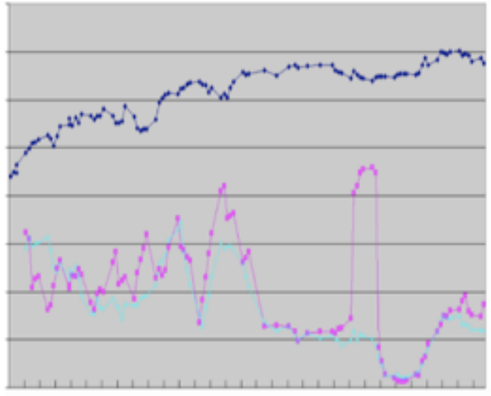


endpoint timing



lipsync trend

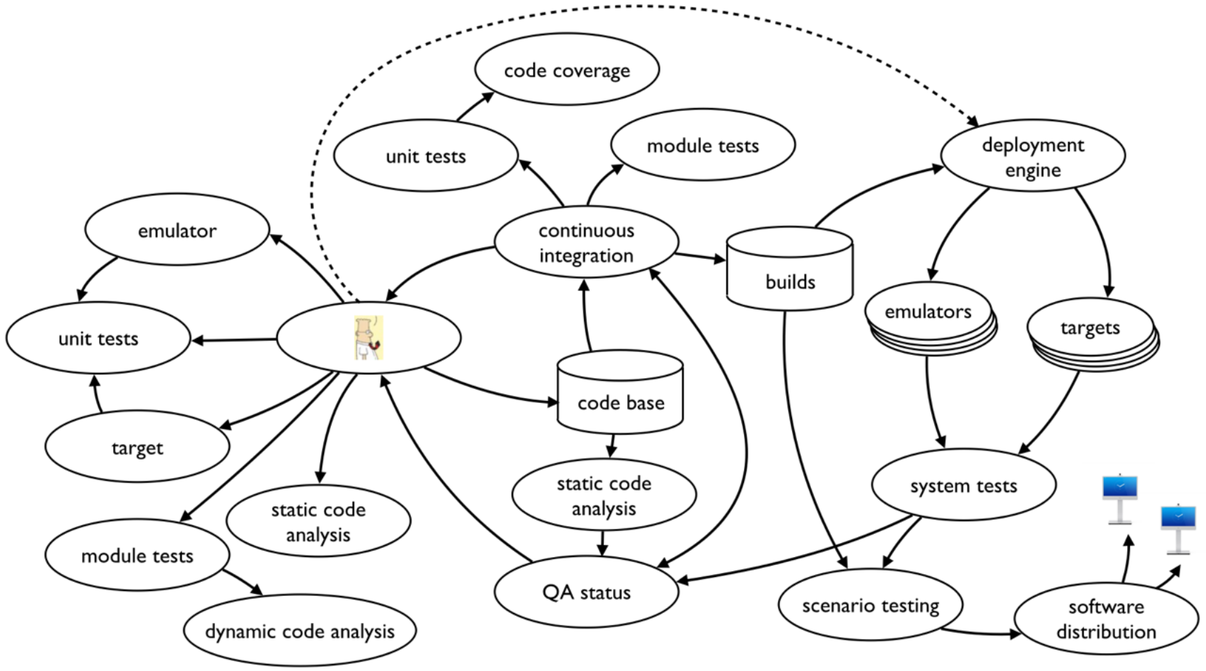


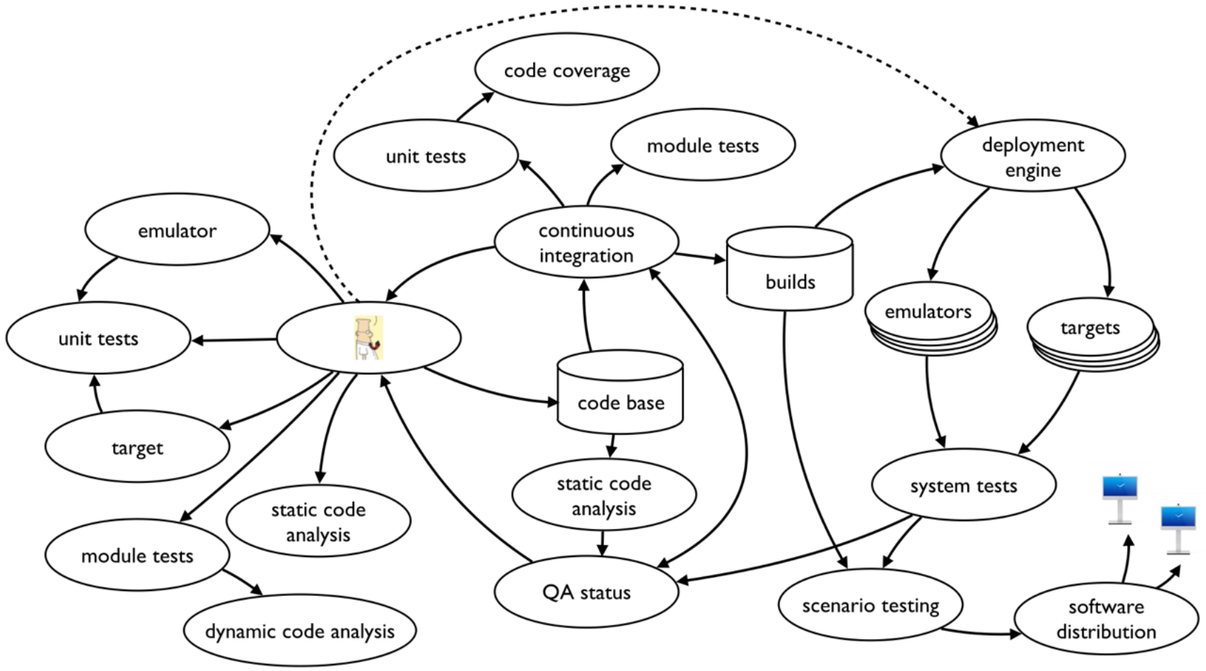


defect trends

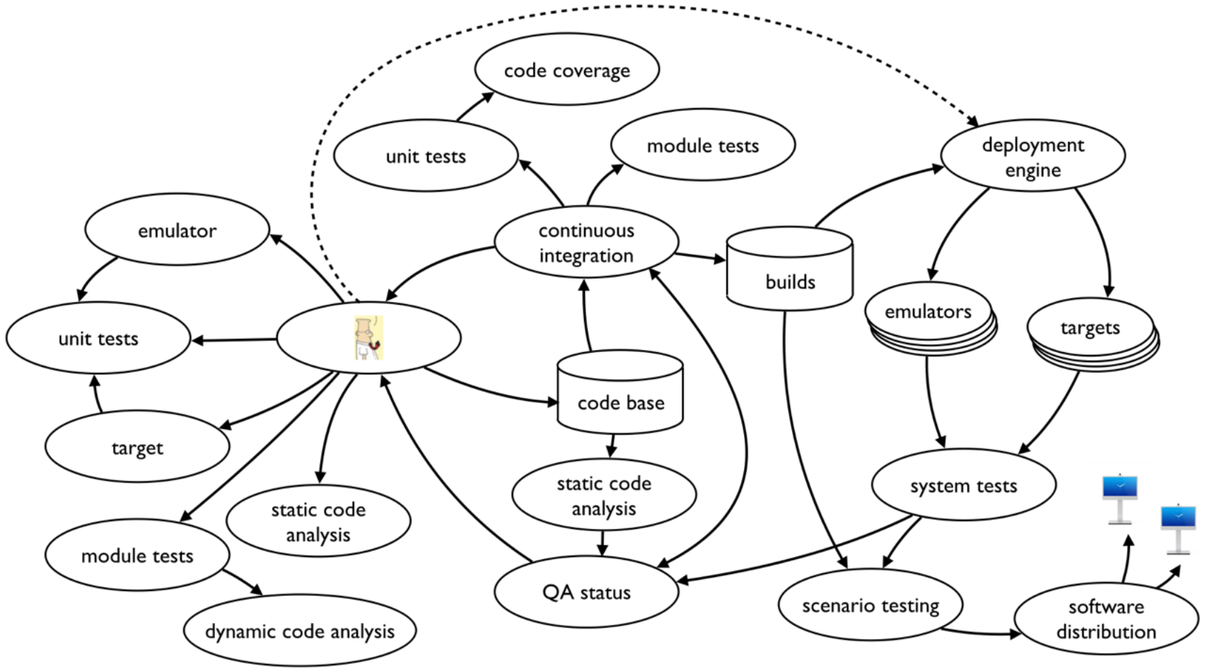
coverity

QA Status


code coverage
unit tests
module tests
deployment engine
emulator
continuous integration
builds
emulators
targets
unit tests
code base
target
static code analysis
system tests
static code analysis
module tests
QA status
scenario testing
software distribution
dynamic code analysis

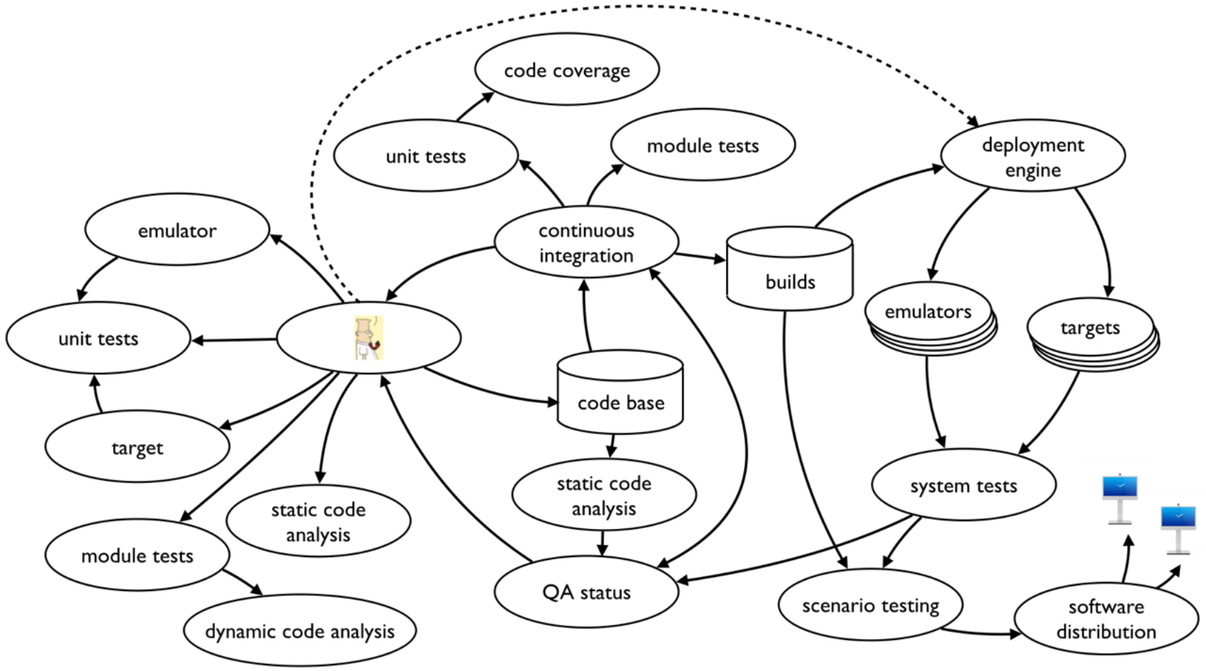


We want something like that! Where do we start?



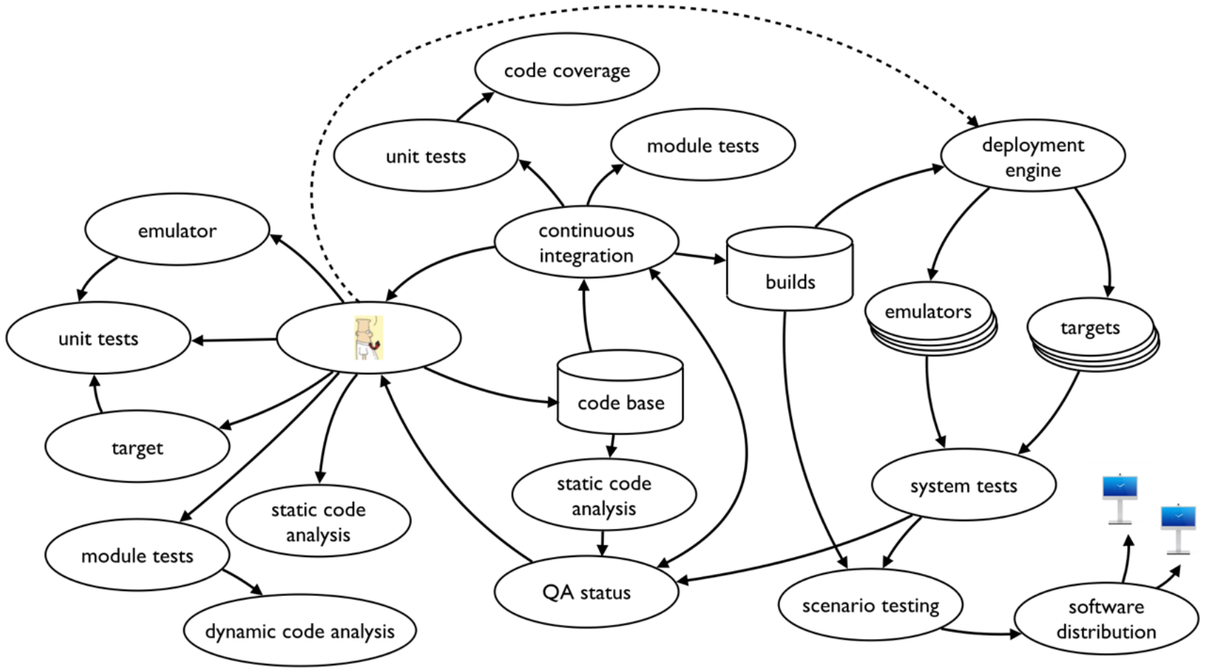
We want something like that! Where do we start?

- Create a robust build system



We want something like that! Where do we start?

- Create a robust build system
- Integrate continuously



We want something like that! Where do we start?

- Create a robust build system
- Integrate continuously
- Grow professionalism

Create a robust build system

Create a robust build system

- Embedded? Create your own build system!

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code
- Aim for a clean build, eg get rid of warnings (-Werror)

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code
- Aim for a clean build, eg get rid of warnings (-Werror)
- Superfast, incremental and partial builds

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code
- Aim for a clean build, eg get rid of warnings (-Werror)
- Superfast, incremental and partial builds
- Heterogeneous development environment (avoid the VS6 effect)

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code
- Aim for a clean build, eg get rid of warnings (-Werror)
- Superfast, incremental and partial builds
- Heterogeneous development environment (avoid the VS6 effect)
- Invest in writing good emulators

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code
- Aim for a clean build, eg get rid of warnings (-Werror)
- Superfast, incremental and partial builds
- Heterogeneous development environment (avoid the VS6 effect)
- Invest in writing good emulators
- Make sure unit tests can run on dev machine, emulator and target

Create a robust build system

- Embedded? Create your own build system!
- Check in build system with your code
- Aim for a clean build, eg get rid of warnings (-Werror)
- Superfast, incremental and partial builds
- Heterogeneous development environment (avoid the VS6 effect)
- Invest in writing good emulators
- Make sure unit tests can run on dev machine, emulator and target
- Integrate your test systems into your build system (`--test-all`)

Integrate continuously

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Feature branches are evil! Try feature toggles instead.

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Feature branches are evil! Try feature toggles instead.
- Test everything, for all commits

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Feature branches are evil! Try feature toggles instead.
- Test everything, for all commits
- Focus on superfast feedback

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Feature branches are evil! Try feature toggles instead.
- Test everything, for all commits
- Focus on superfast feedback
- Invest in equipment for fast and complete system testing

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Feature branches are evil! Try feature toggles instead.
- Test everything, for all commits
- Focus on superfast feedback
- Invest in equipment for fast and complete system testing
- Prune unused metrics and feedback mechanisms

Integrate continuously

- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Feature branches are evil! Try feature toggles instead.
- Test everything, for all commits
- Focus on superfast feedback
- Invest in equipment for fast and complete system testing
- Prune unused metrics and feedback mechanisms
- Slim down your QA department

Grow professionalism

Grow professionalism

- make sure you have enough slack in the system

Grow professionalism

- make sure you have enough slack in the system
- avoid staged or gated commits, some broken builds are acceptable

Grow professionalism

- make sure you have enough slack in the system
- avoid staged or gated commits, some broken builds are acceptable
- focus on the flow of changes

Grow professionalism

- make sure you have enough slack in the system
- avoid staged or gated commits, some broken builds are acceptable
- focus on the flow of changes
- make everything visible and advocate collective ownership

Grow professionalism

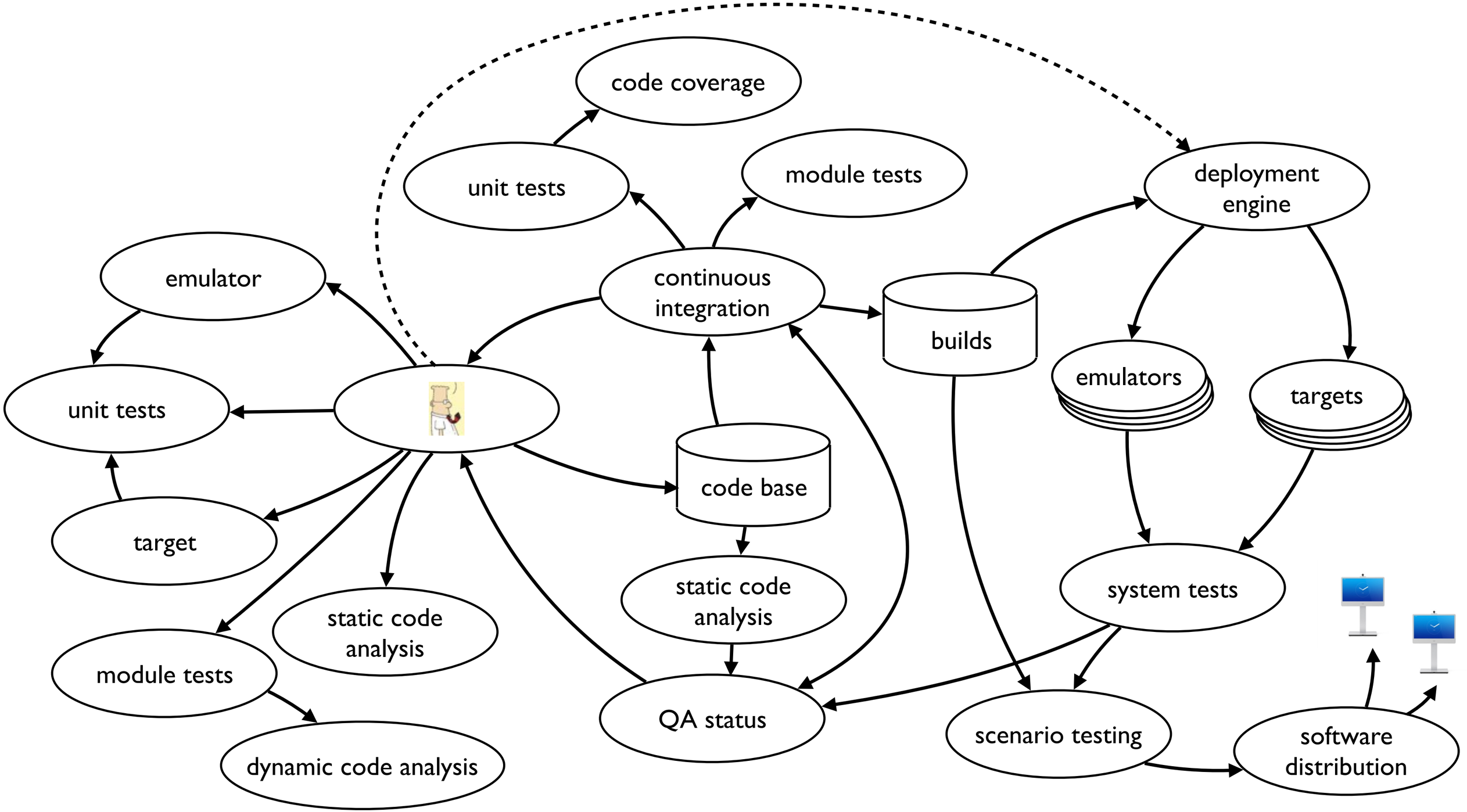
- make sure you have enough slack in the system
- avoid staged or gated commits, some broken builds are acceptable
- focus on the flow of changes
- make everything visible and advocate collective ownership
- encourage code reviews, but avoid mandatory formal code reviews

Grow professionalism

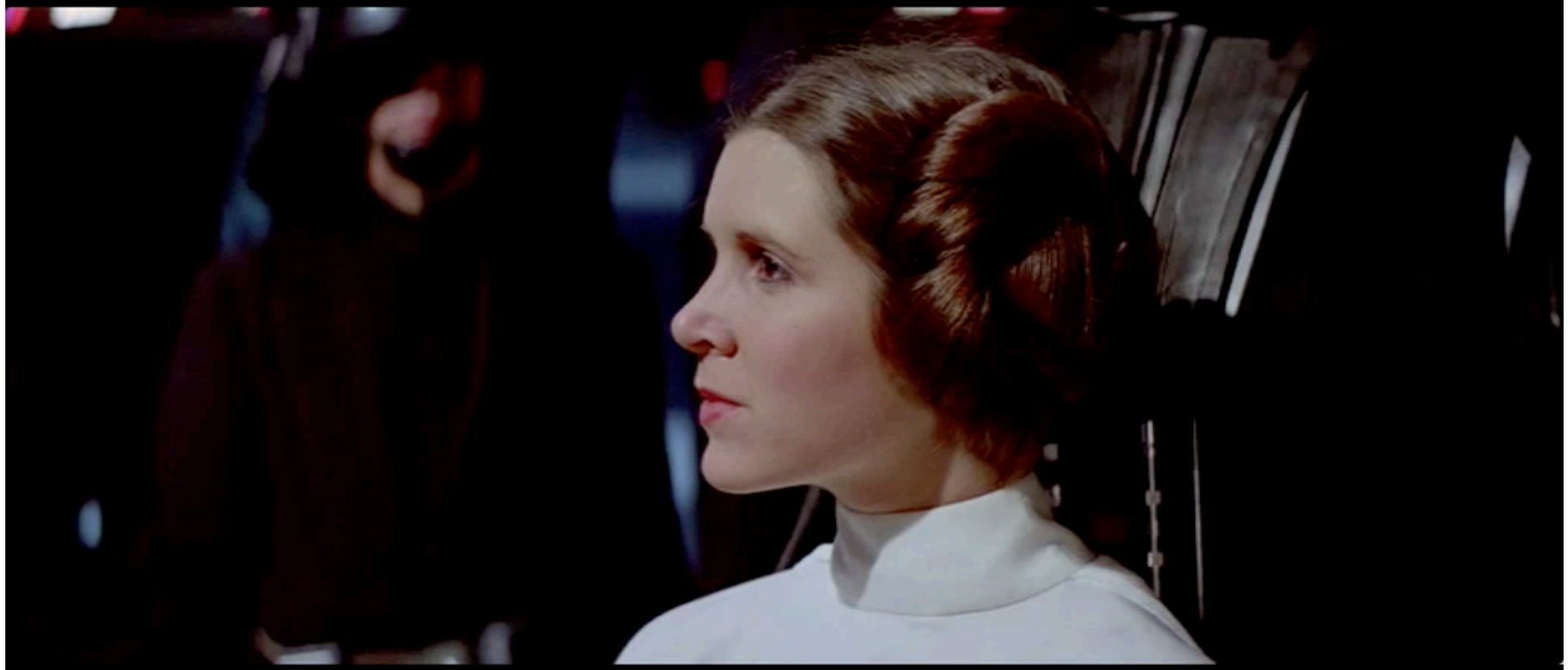
- make sure you have enough slack in the system
- avoid staged or gated commits, some broken builds are acceptable
- focus on the flow of changes
- make everything visible and advocate collective ownership
- encourage code reviews, but avoid mandatory formal code reviews
- beware of the observer effect

Grow professionalism

- make sure you have enough slack in the system
- avoid staged or gated commits, some broken builds are acceptable
- focus on the flow of changes
- make everything visible and advocate collective ownership
- encourage code reviews, but avoid mandatory formal code reviews
- beware of the observer effect
- optimize for your top 80% developers

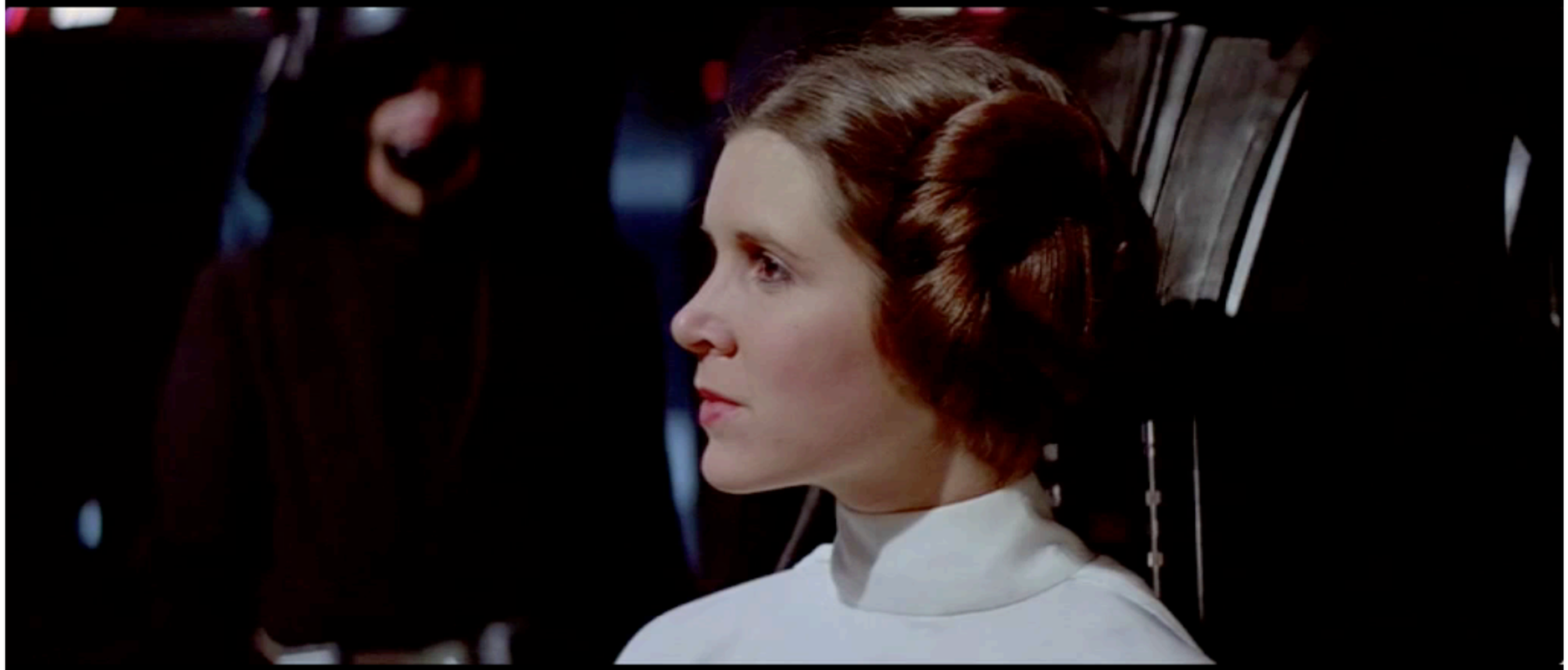

code coverage
unit tests
module tests
deployment engine
emulator
continuous integration
builds
unit tests
emulators
targets
code base
target
static code analysis
system tests
static code analysis
module tests
QA status
scenario testing
software distribution
dynamic code analysis

?



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

(Princess Leia)



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

(Princess Leia)

!

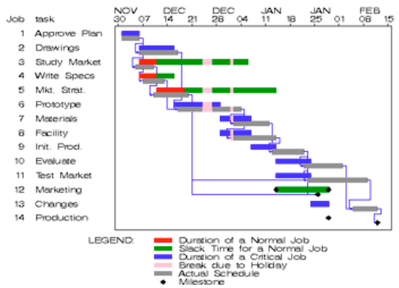
About Agile Principles

There used to be a time, where we believed that anyone could do software development

after all, it was just about programming a computer...

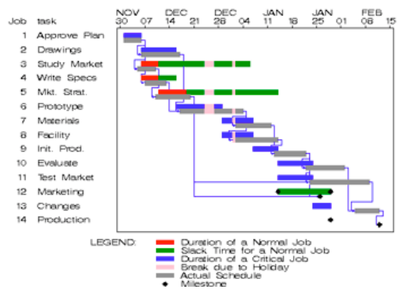
1) get some smart people to analyze the problem

1) get some smart people to analyze the problem



2) create a plan

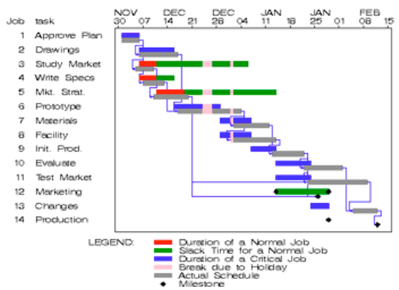
1) get some smart people to analyze the problem



2) create a plan

3) find resources

1) get some smart people to analyze the problem



2) create a plan

3) find resources

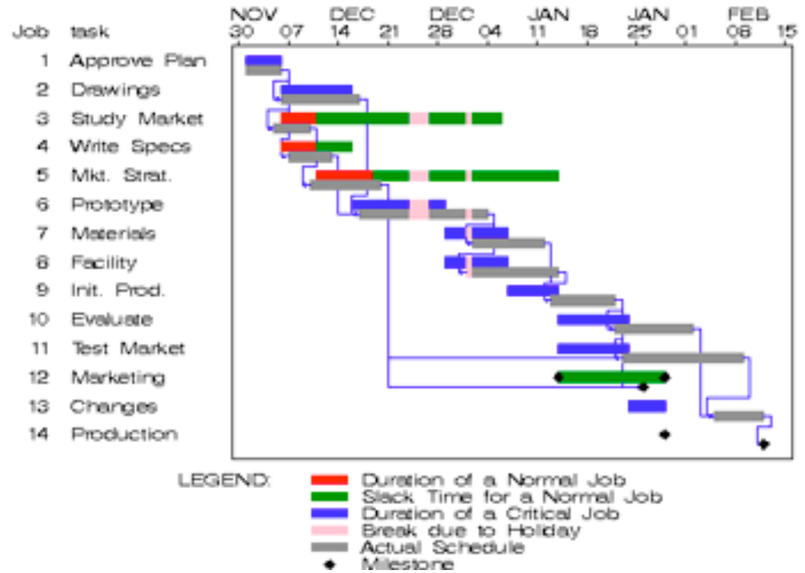
4) execute according to the plan

and when the projects failed

the respons was always:

do more up-front analysis

create a more detailed plan



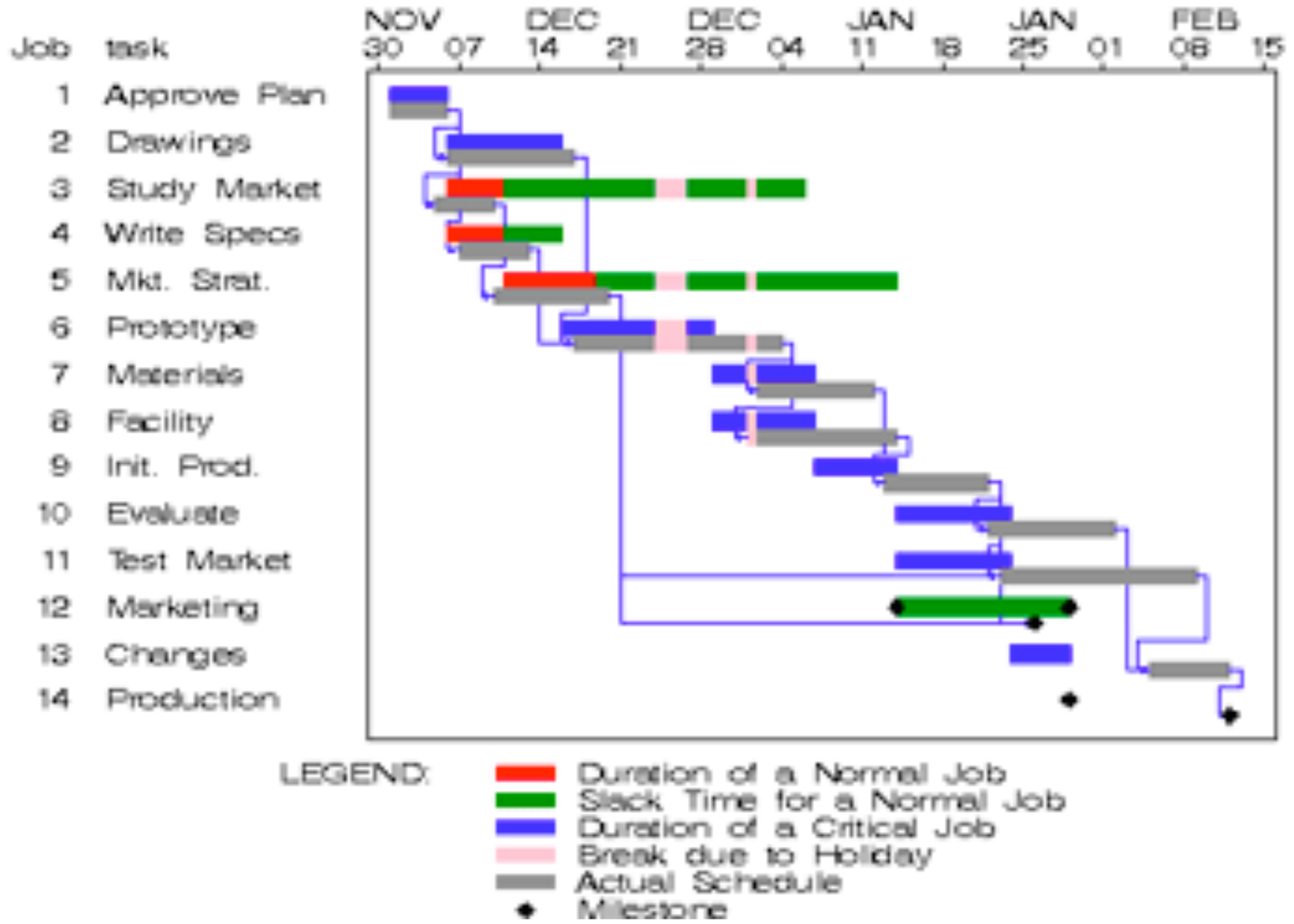
find more resources

and make sure that everyone followed the plan

but the projects still failed

and the respons was, as usual:

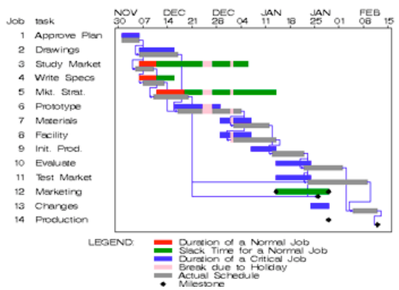

FREE
OR THE UNEMPLOYED
FREE SOUP


NOV
DEC
DEC
JAN
JAN
FEB
30
07
14
21
28
04
11
18
25
01
08
15
Job
task
1 Approve Plan
2 Drawings
3 Study Market
4 Write Specs
5 Mkt. Strat.
6 Prototype
7 Materials
8 Facility
9 Init. Prod.
10 Evaluate
11 Test Market
12 Marketing
13 Changes
14 Production
LEGEND:
Duration of a Normal Job
Slack Time for a Normal Job
Duration of a Critical Job
Break due to Holiday
Actual Schedule
Milestone

but of course...

Dark ages of software development (early 80's to late 90's)

1) get some smart people to analyze the problem



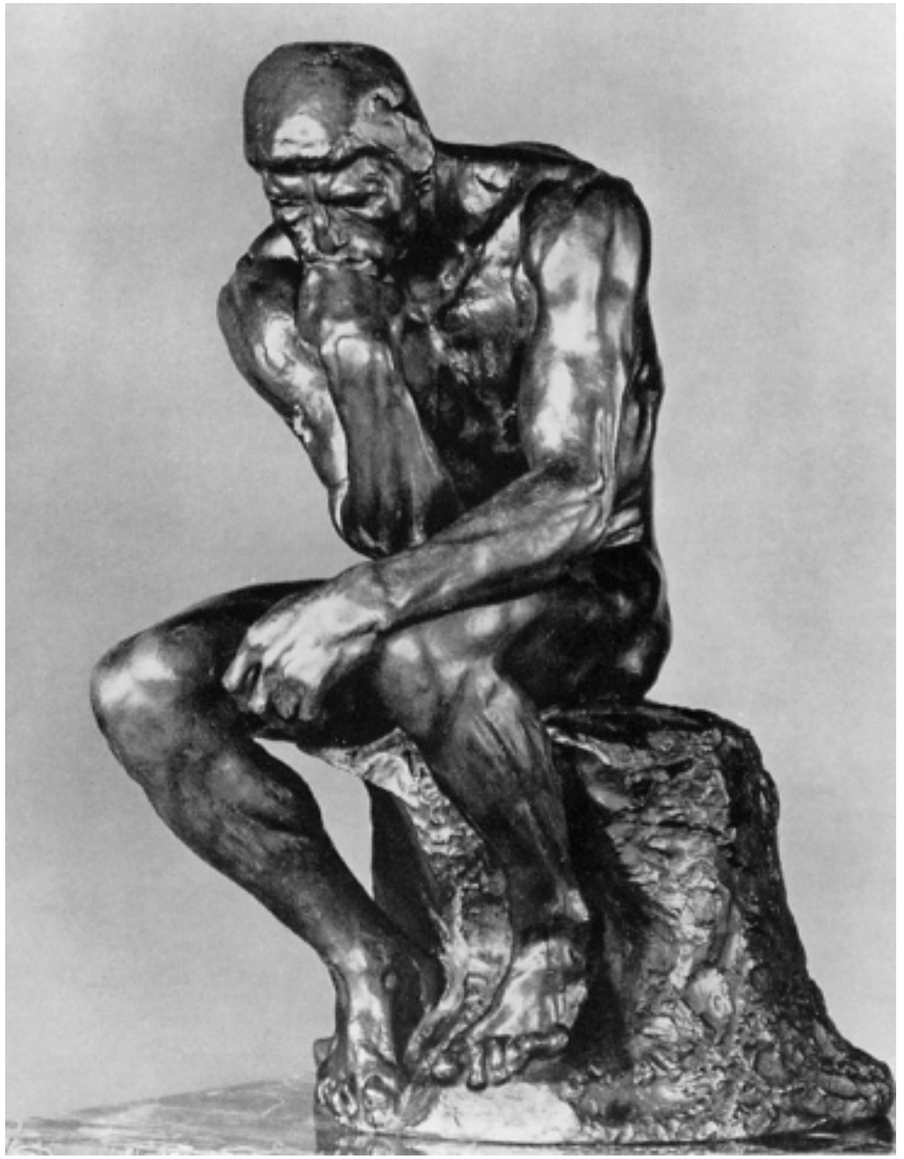
2) create a plan

3) find resources

4) execute according to the plan

We had only discovered a fancy way of playing the “scabby queen” game, also known as the “Old Maid” or “Svarte Per”, always try to “save your ass” by delegating responsibility to someone else.

There must be a better way...



The Agile Manifesto (2001)

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

Individuals and interactions over **processes and tools**
Working software over **comprehensive documentation**
Customer collaboration over **contract negotiation**
Responding to change over **following a plan**

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

The agile manifesto started a huge awakening process in the software industry...

(picture from the 1990 film Awakenings)

processes and tools
comprehensive documentation
contract negotiation
following a plan

processes and tools
comprehensive documentation
contract negotiation
following a plan

Individuals and interactions
Working solutions
Customer collaboration
Responding to change

Individuals and interactions
Working solutions
Customer collaboration
Responding to change

Seven Enemies of Agile

Plans
Commitments
Pressure
Objectives
Documentation
Inspection
Procedures

Plan

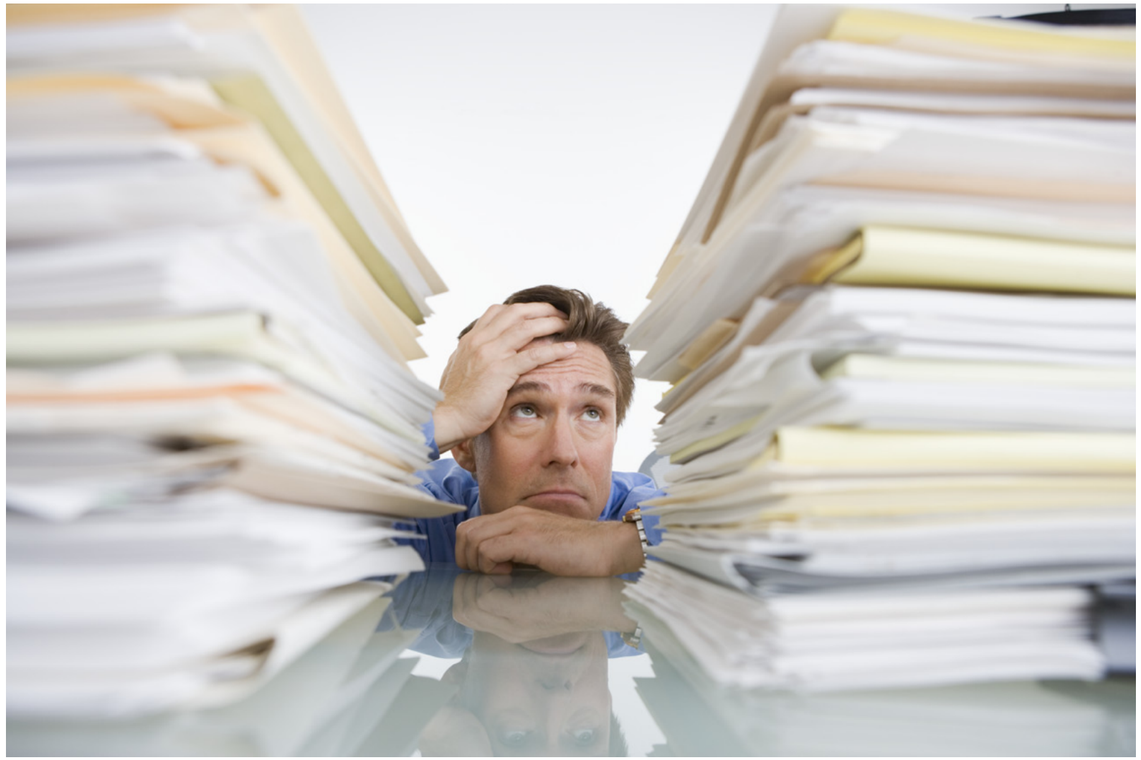
Everyone has a plan 'till they get punched in the mouth.
Mike Tyson

Commitments

Pressure

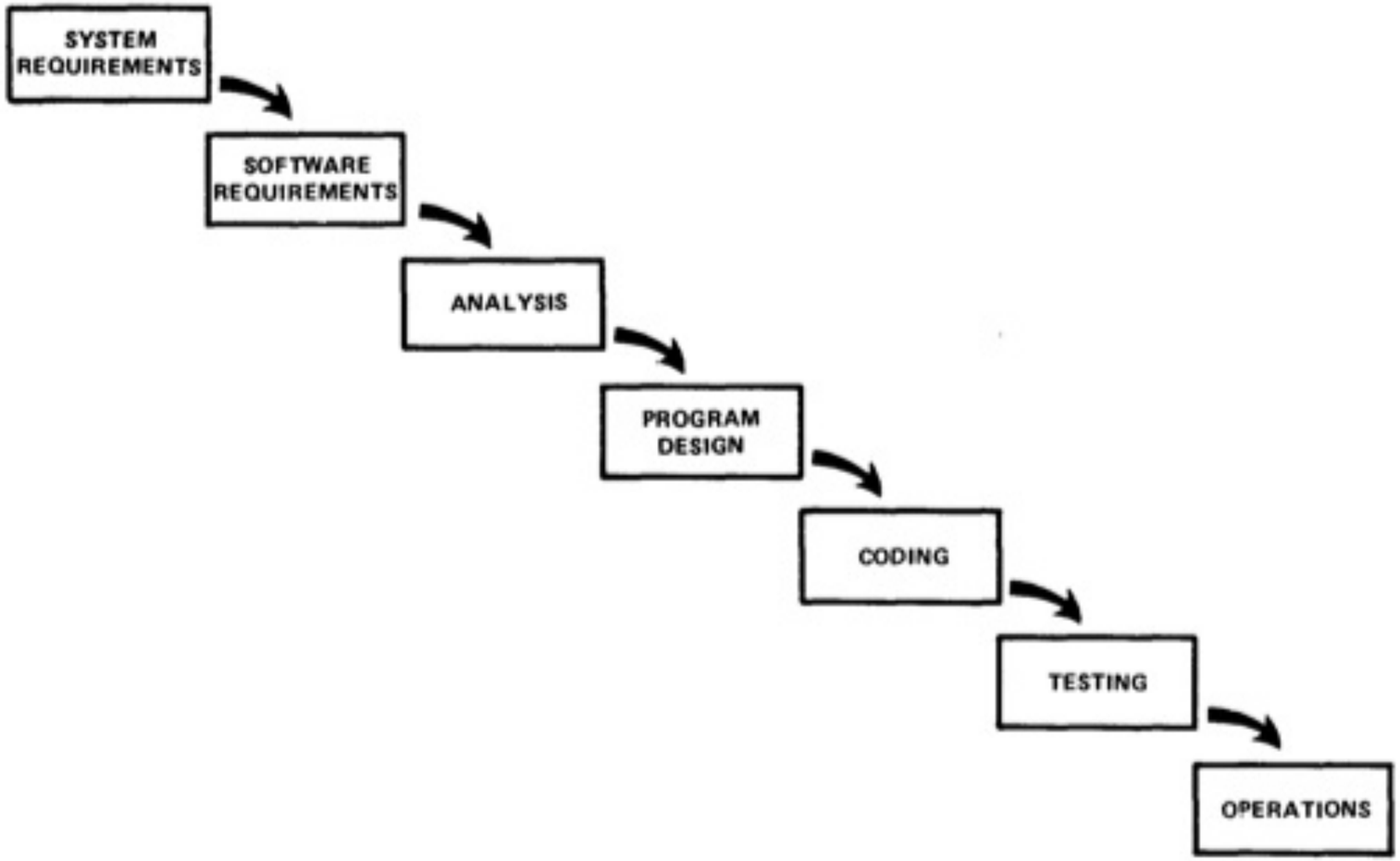
Objectives

Documentation



Inspection

Procedures


SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
ANALYSIS
PROGRAM DESIGN
CODING
TESTING
OPERATIONS

Seven Enemies of Agile

Plans
Commitments
Pressure
Objectives
Documentation
Inspection
Procedures


DWW

Seven Friends of Agile

Seven Friends of Agile

Planning

Seven Friends of Agile

Planning
Collaboration

Seven Friends of Agile

Planning
Collaboration
Slack

Seven Friends of Agile

Planning
Collaboration
Slack
Direction

Seven Friends of Agile

Planning
Collaboration
Slack
Direction
Communication

Seven Friends of Agile

Planning
Collaboration
Slack
Direction
Communication
Reflection

Seven Friends of Agile

Planning
Collaboration
Slack
Direction
Communication
Reflection
Principles

Planning

Collaboration

Slack

Direction

Communication

Reflection

Principles

Plans

Commitments

Pressure

Objectives

Documentation

Inspection

Procedures

Planning

Collaboration

Slack

Direction

Communication

Reflection

Principles

Plans

Commitments

Pressure

Objectives

Documentation

Inspection

Procedures

!