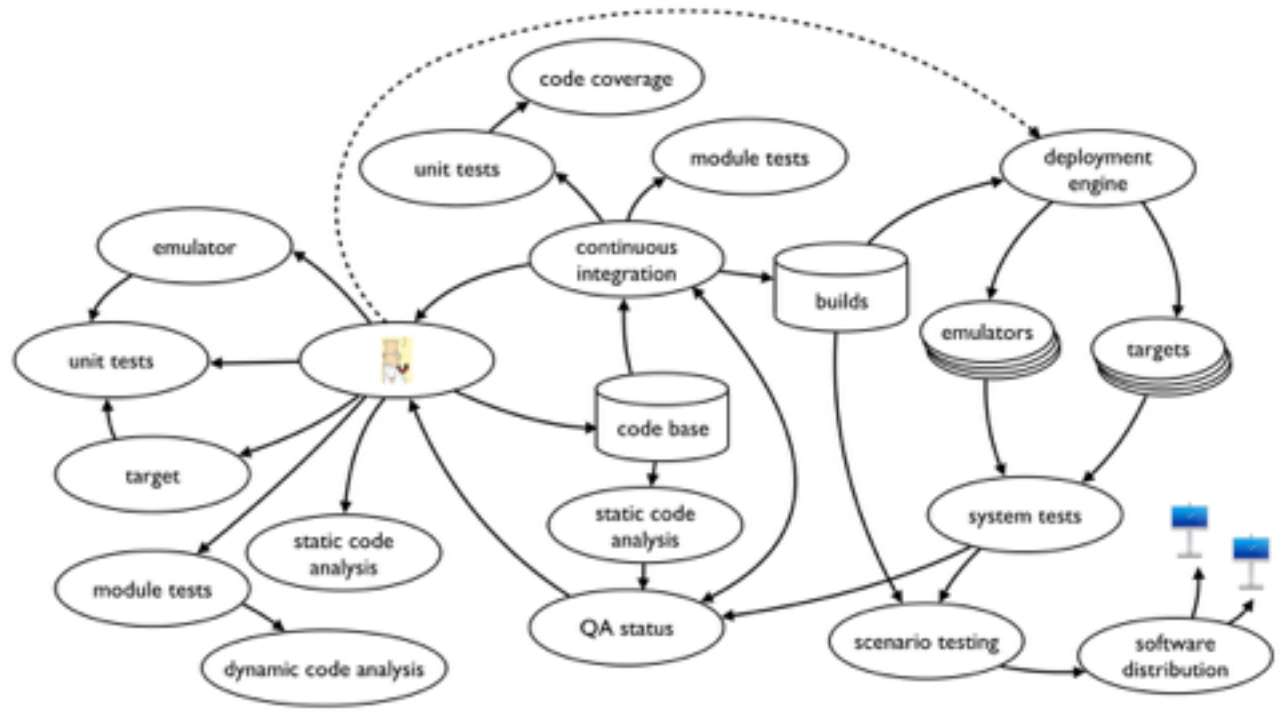
Feedback-driven Product Development

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.

a 60 minute session for Kongsberg Maritime Subsea (Simrad), Horten

November 24, 2016, Olve Maudal

Cisco Systems, Innovation Center
Lysaker, Norway

Telepresence

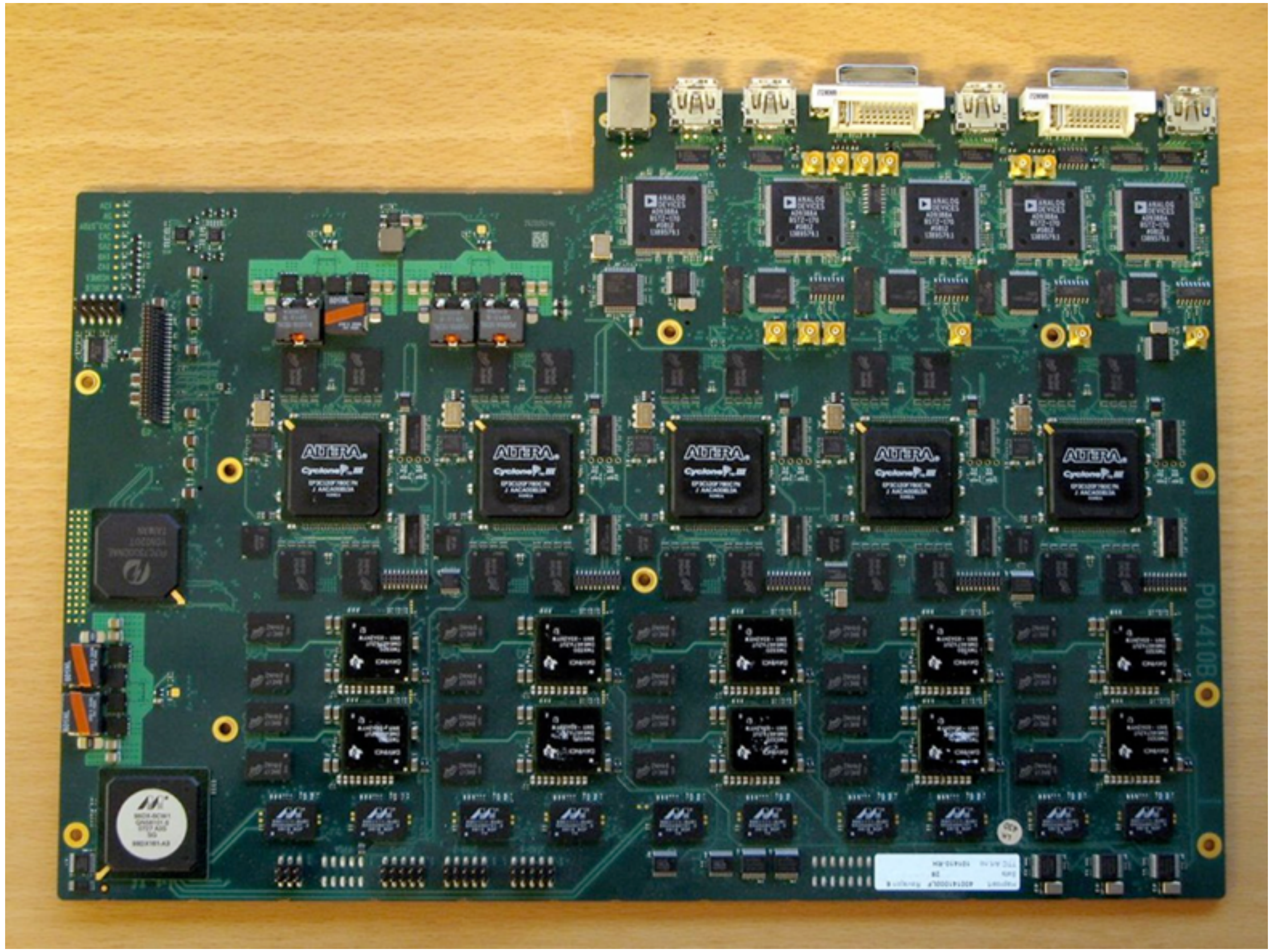
Some of the stuff we develop at Lysaker

at Lysaker we are ~350 engineers

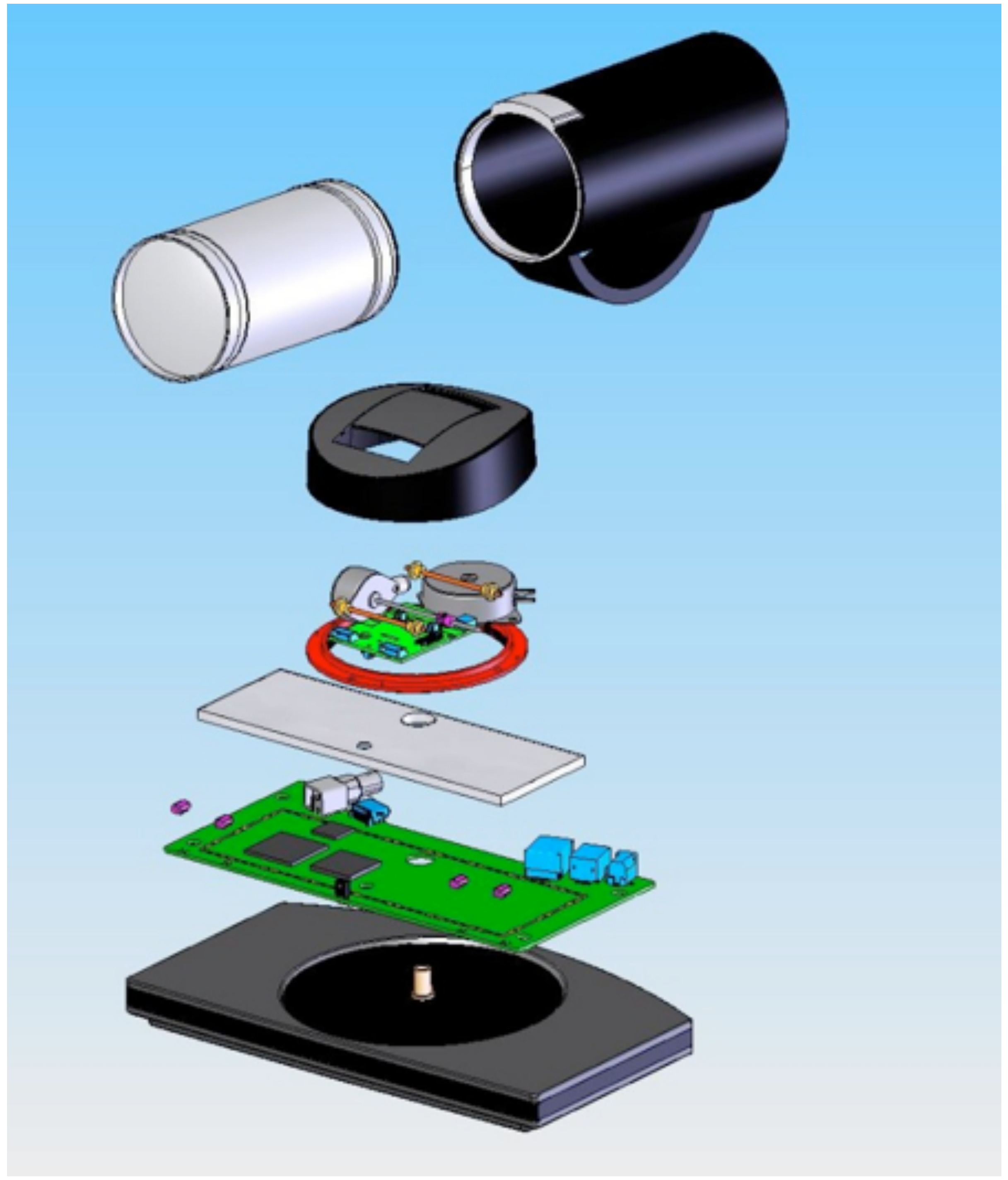
most of us work with software developement

but we also do...

Electronics / Hardware



Mechanics

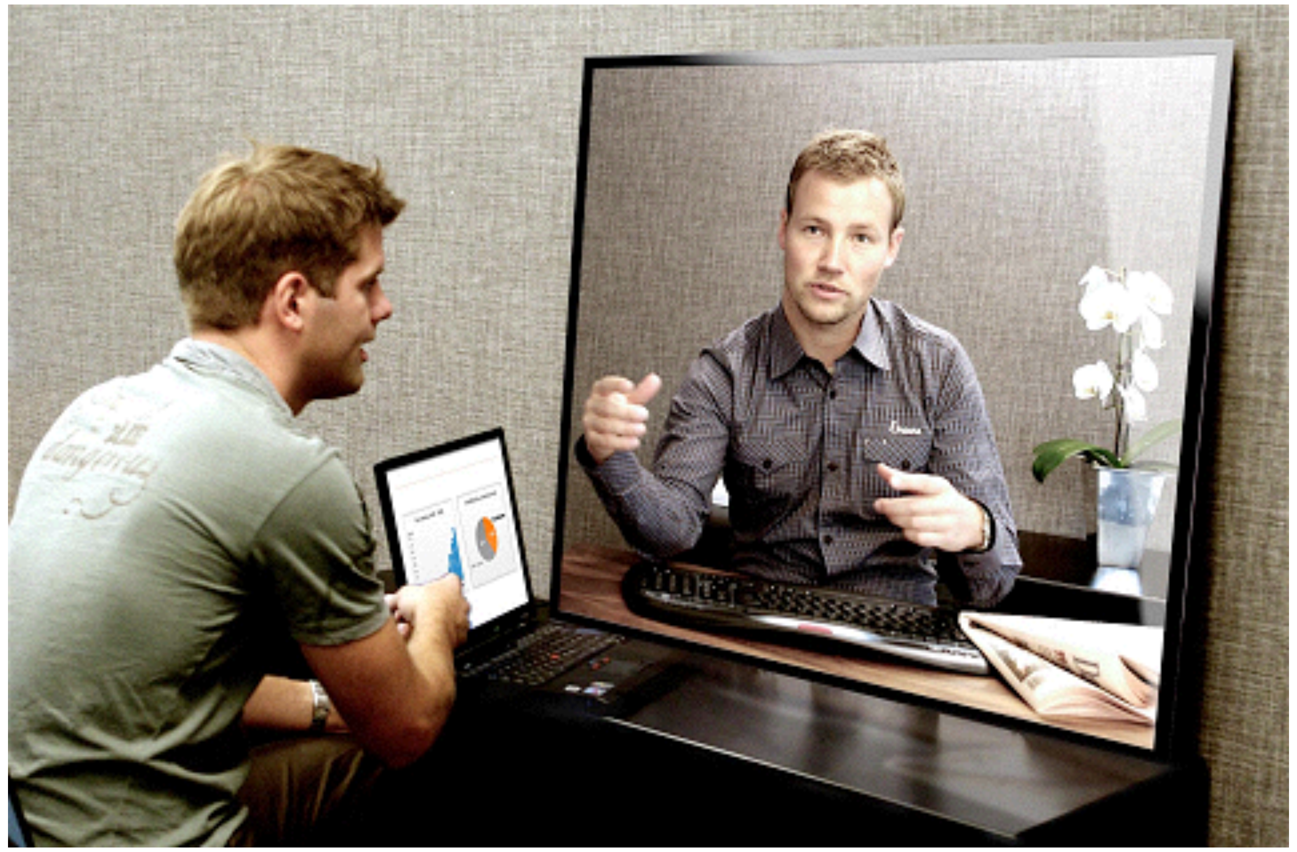


Industrial Design and User Experience Design

1992

2015

Looking into



the future

The main codebase at Lysaker

- embedded software development
- about 200 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

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

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.

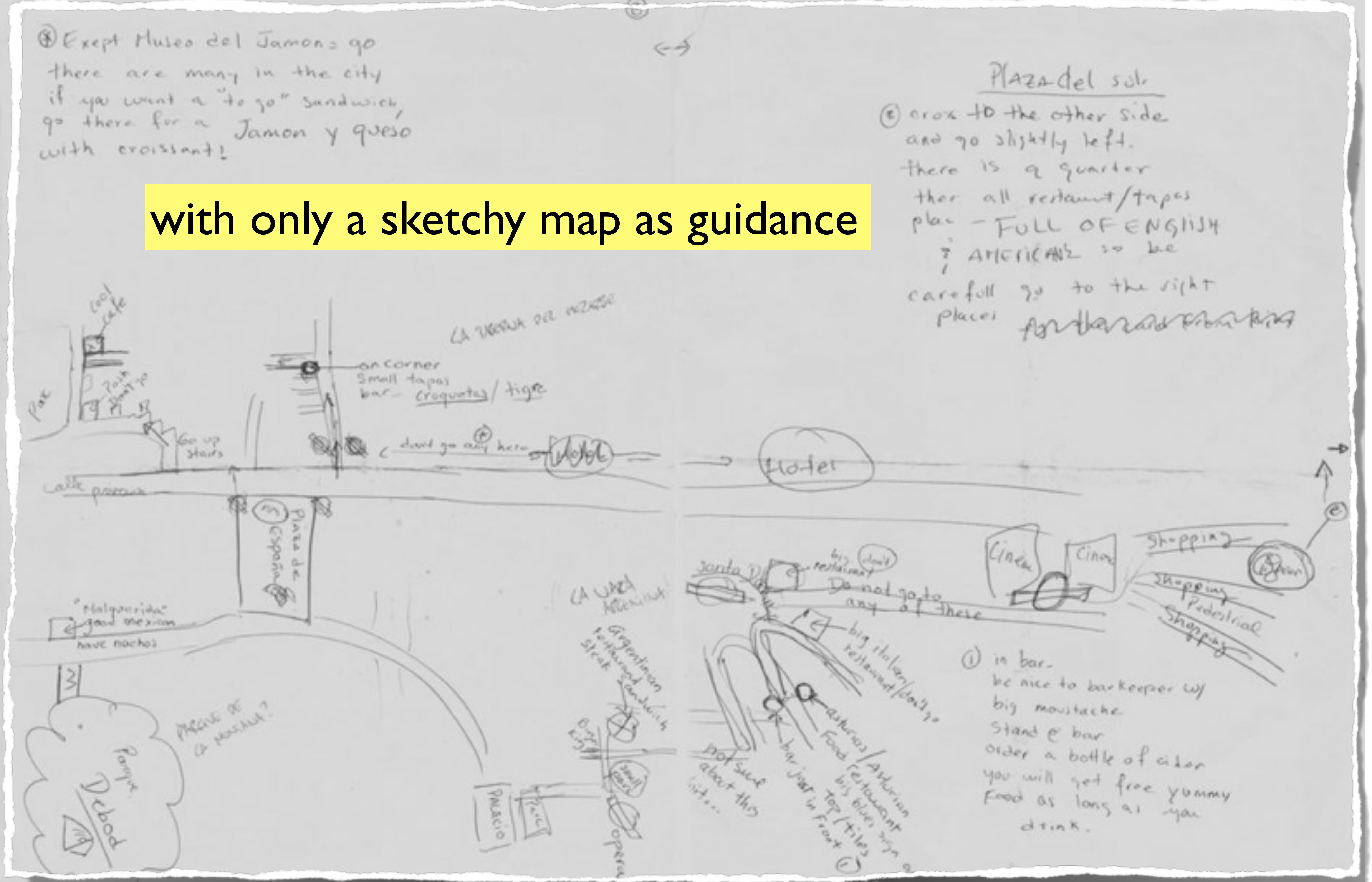
Most projects are more like...

extreme orienteering

in impossible terrain

with a group of people

in the dark



with only a sketchy map as guidance

(8) 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!

PLAZA del sol

(e) 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 ~~[illegible]~~

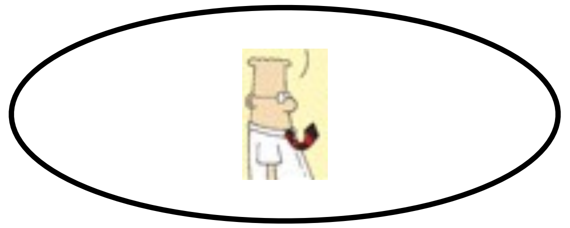
(1) in bar. be nice to barkeeper w/ big moustache. Stand @ bar order a bottle of cider you will get free yummy Food as long as you drink.

cool cafe
Pac
Go up stairs
on corner Small tapas bar — croquetas / tigre
LA TABERNA DEL ALABARDERO
don't go any here
Hotel
Hotel
calle princesa
Plaza de España
"Malgaorida" good mexican have nachos
PARQUE DE LA MONTAÑA?
Parque Debod
LA VACA ARGENTINA
Argentinian restaurant sandwich steak
Burger King
small park
PALACIO
Opera
Santo Domingo
big restaurant don't
Do not go to any of these
big italian restaurant / don't go
asturias / Asturian Food restaurant big blues sign top / tiles
bar just in front
not sure about this one...
Cinema
Cinema
Shopping
Shopping
Pedestrian
Shopping

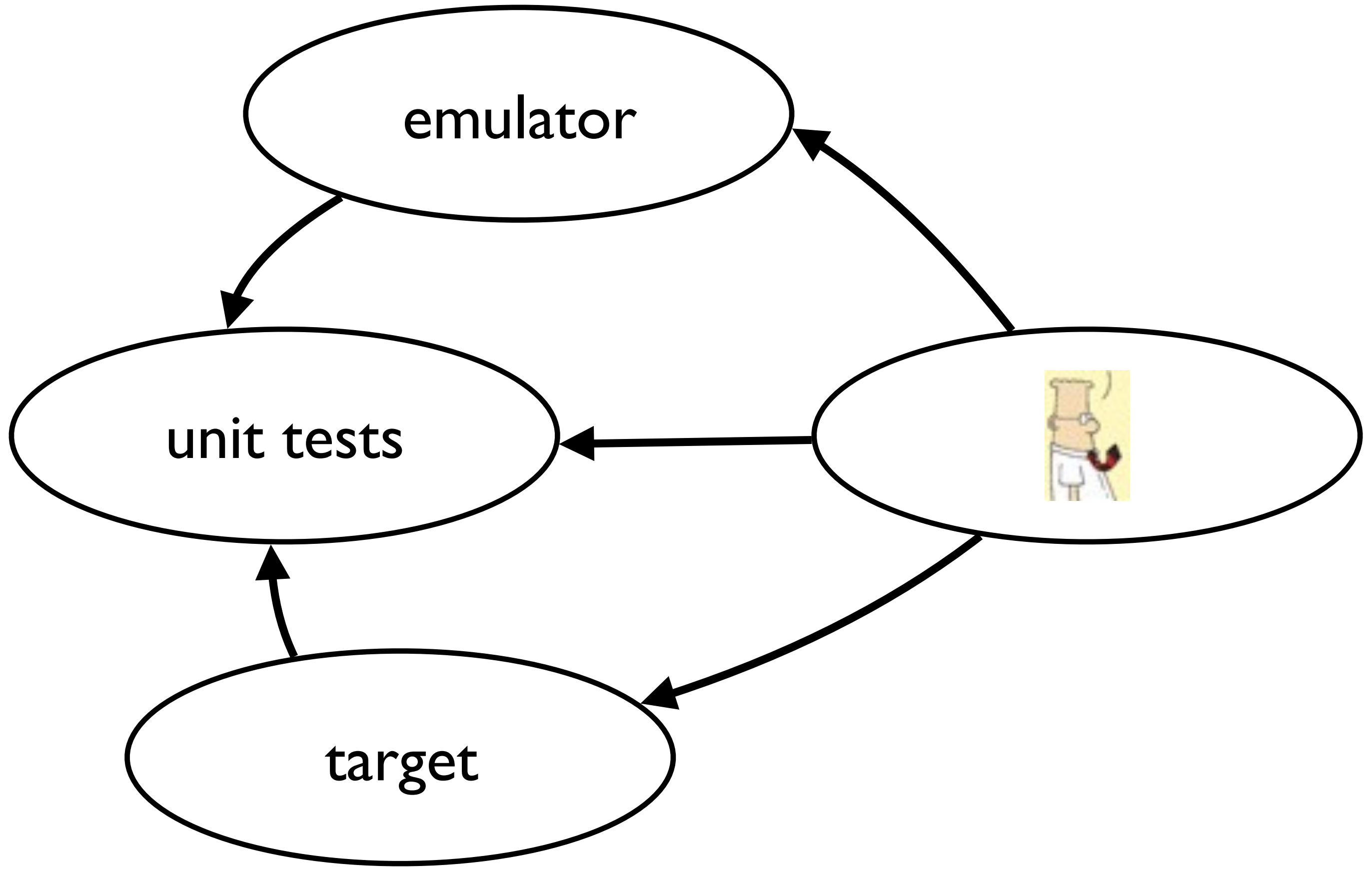
http://www.youtube.com/watch?v=oetF3UTIwbc

An example of a continuous integration and deployment system

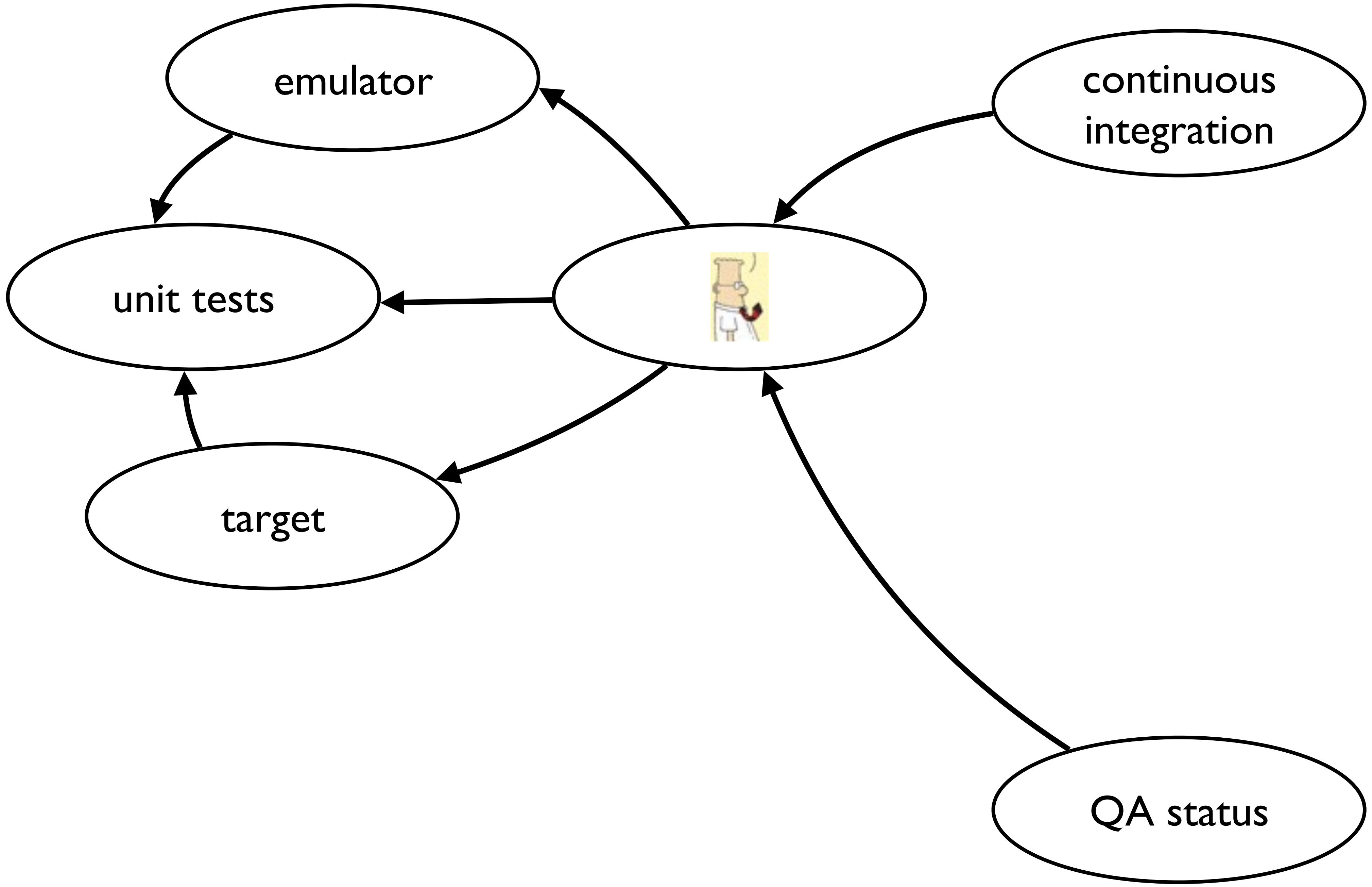
Continuous integration and deployment system



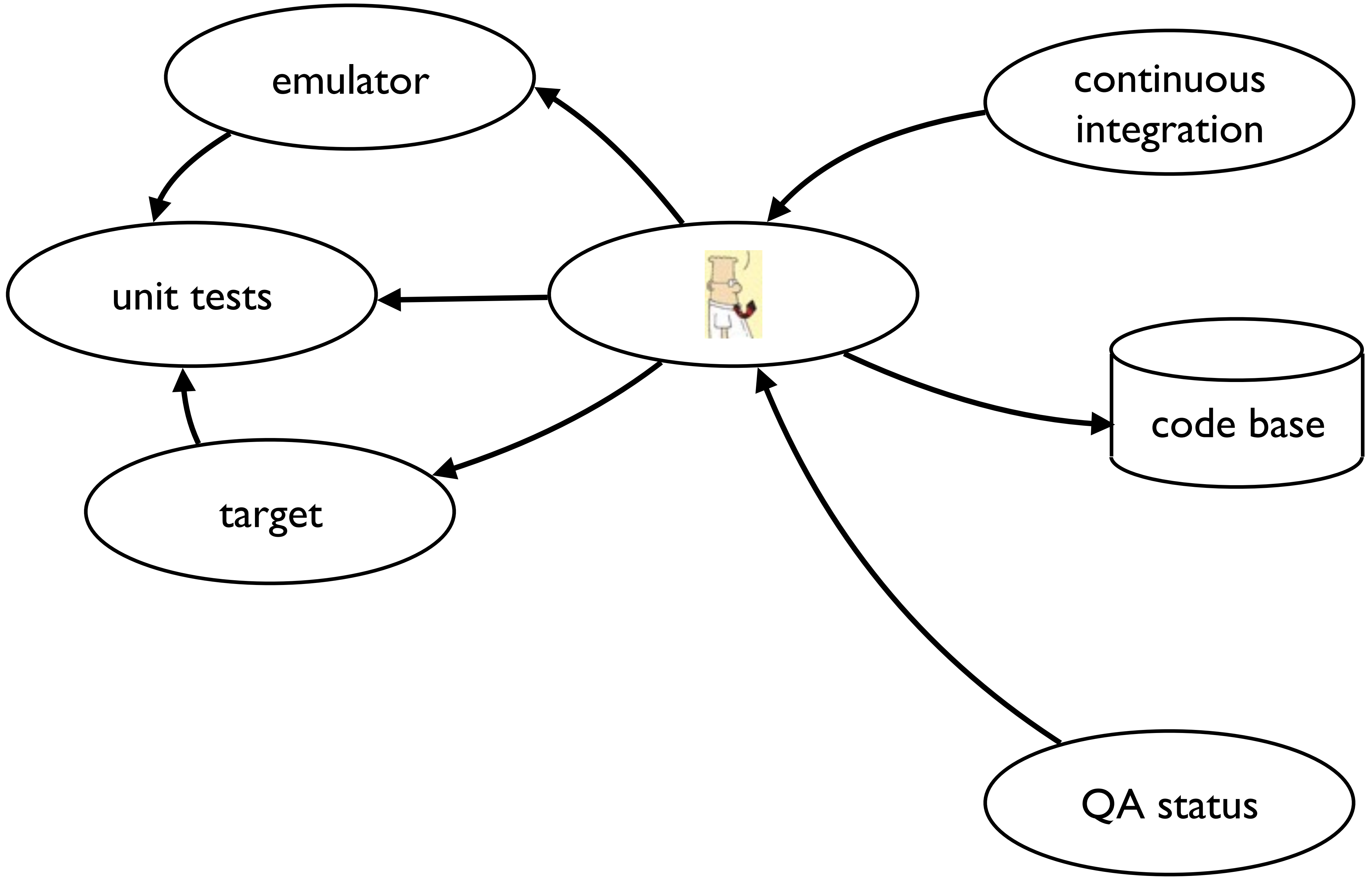
Continuous integration and deployment system



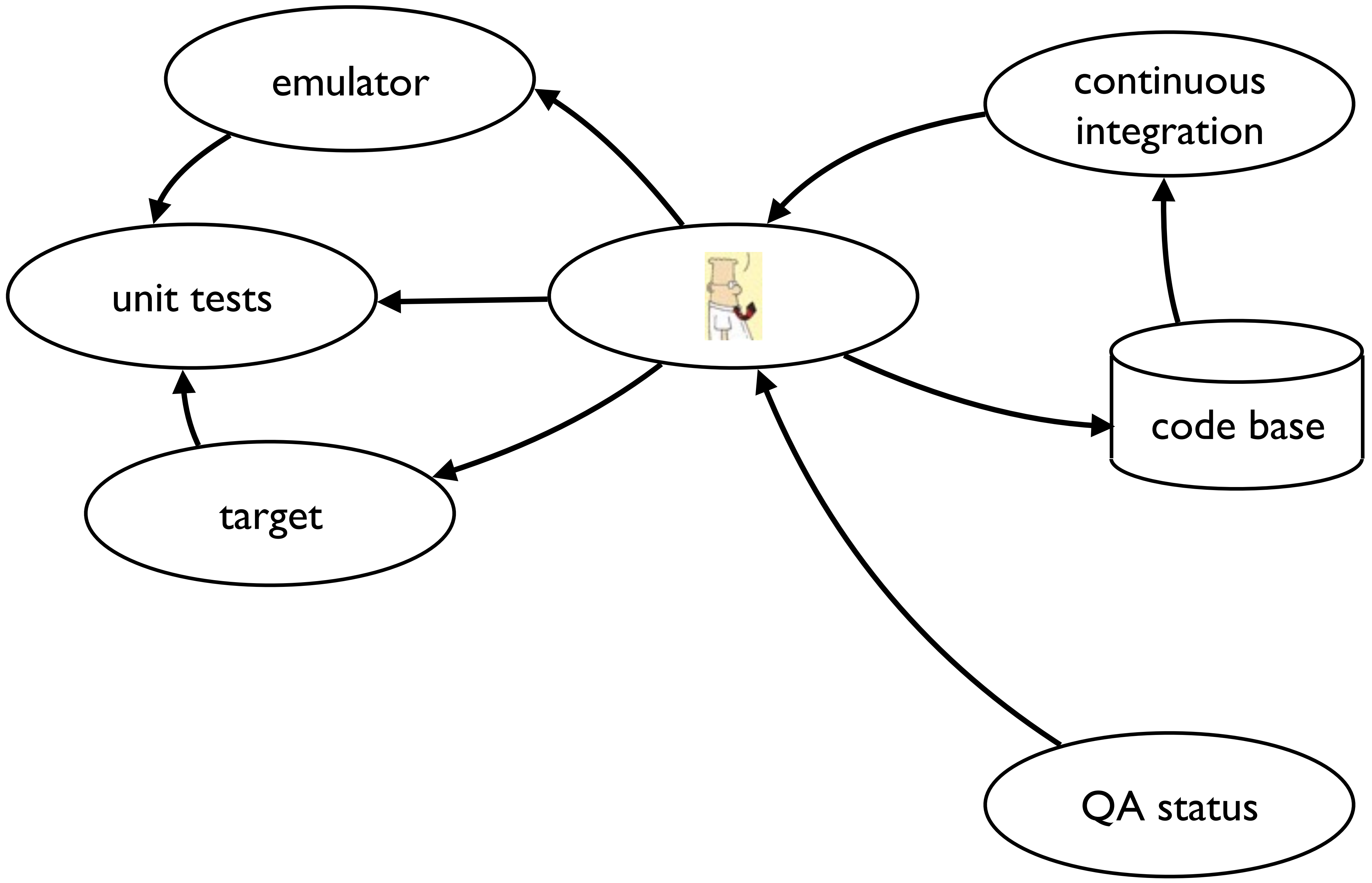
Continuous integration and deployment system



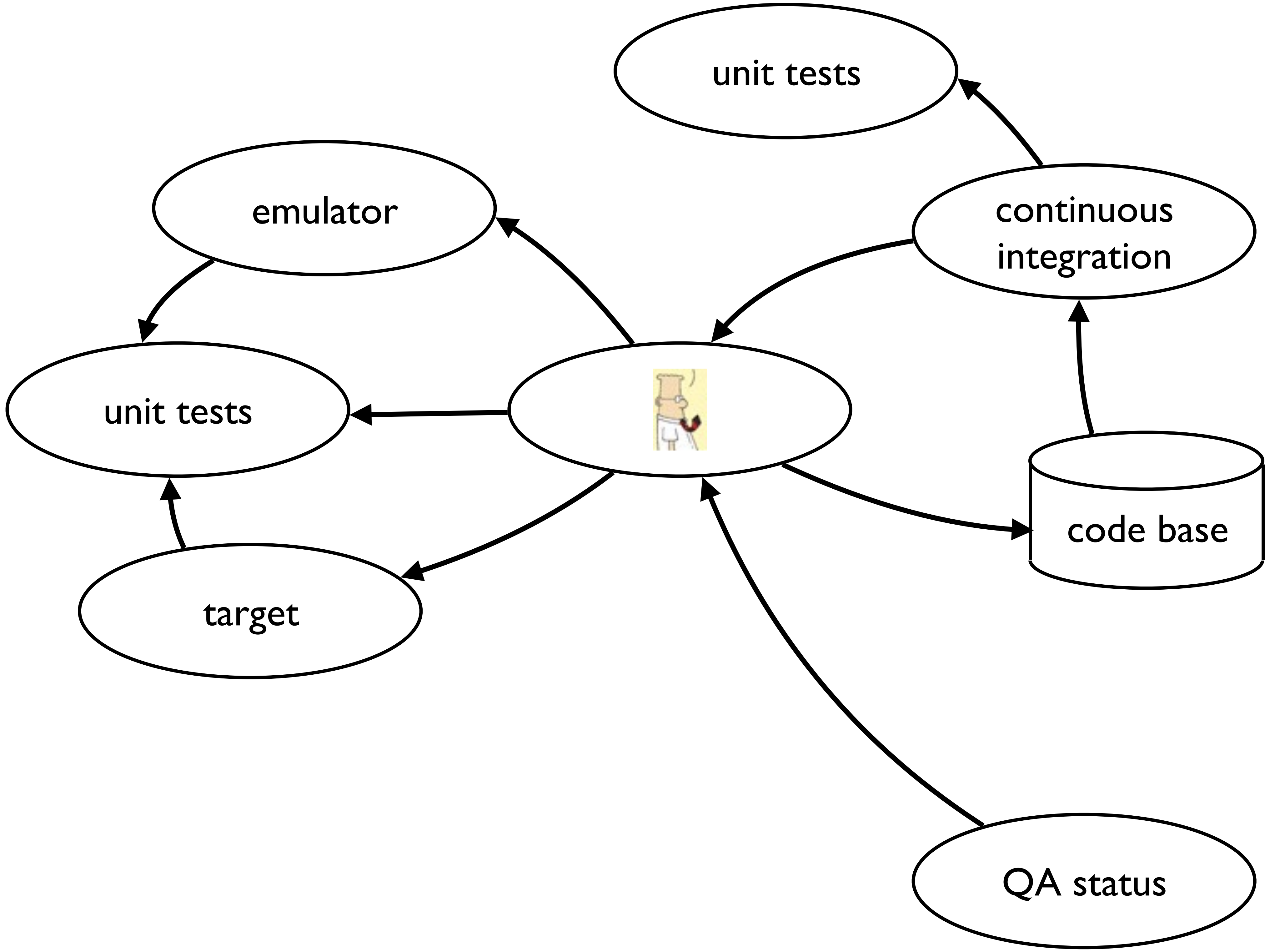
Continuous integration
and deployment
system



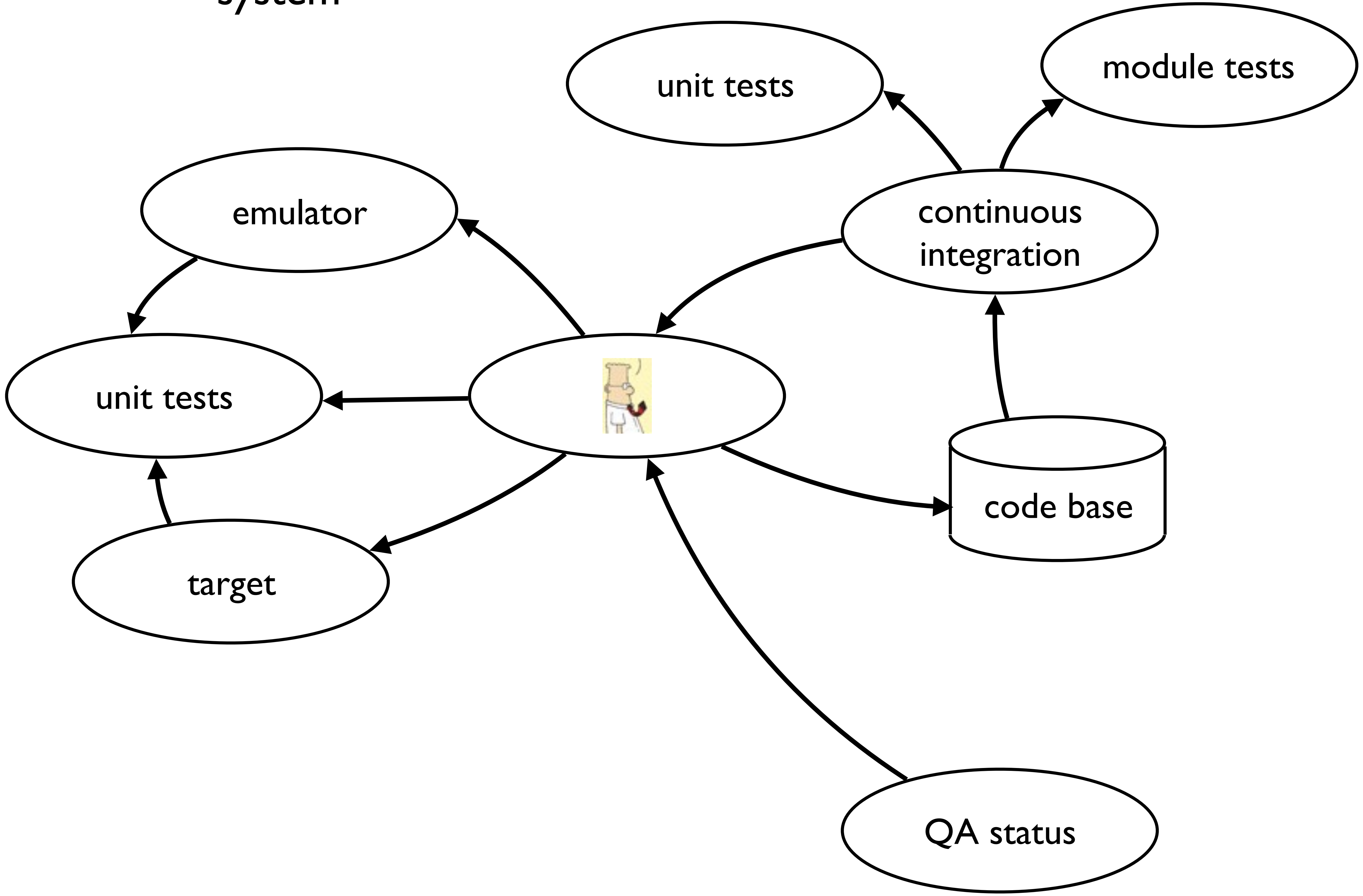
Continuous integration and deployment system



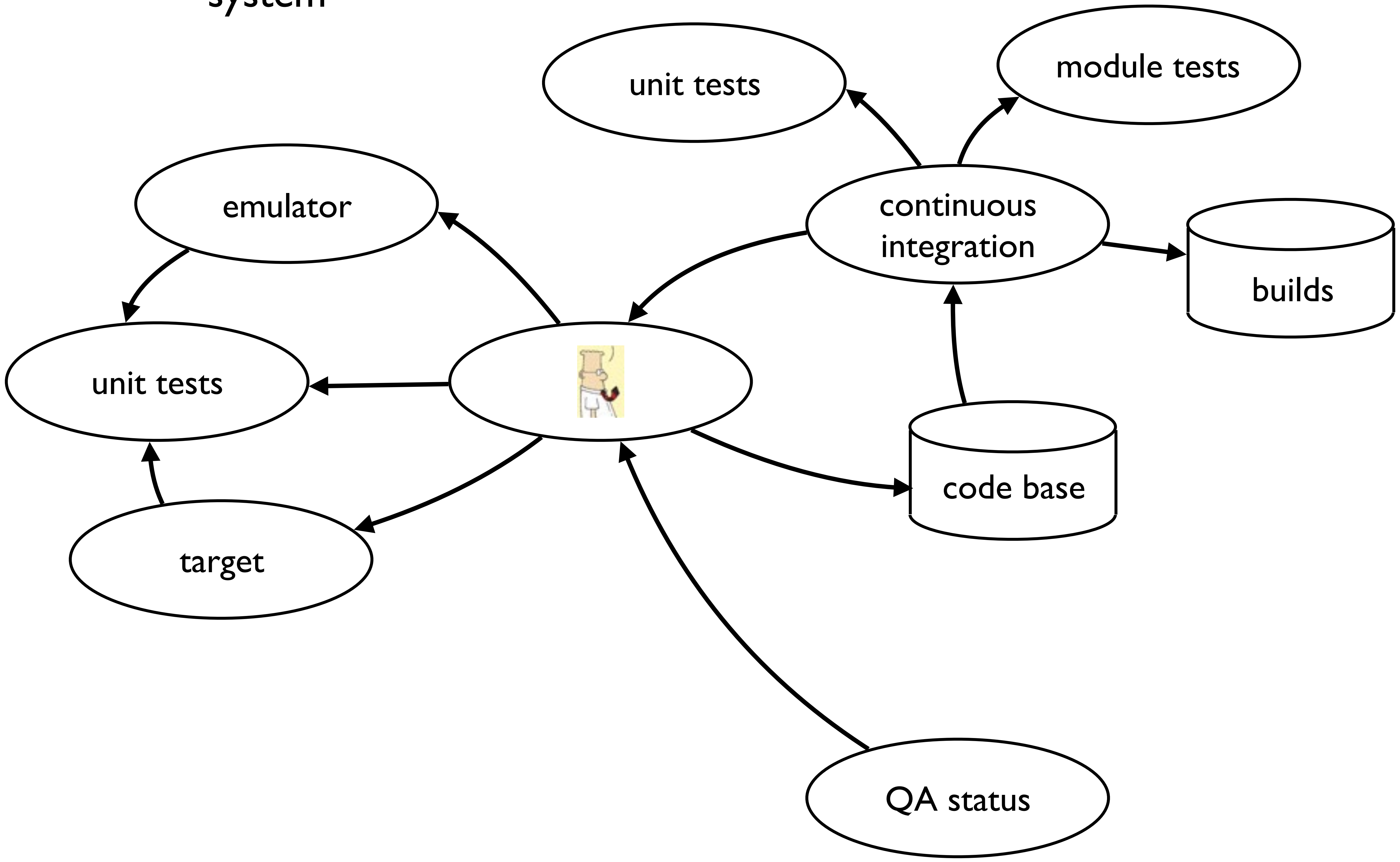
Continuous integration and deployment system



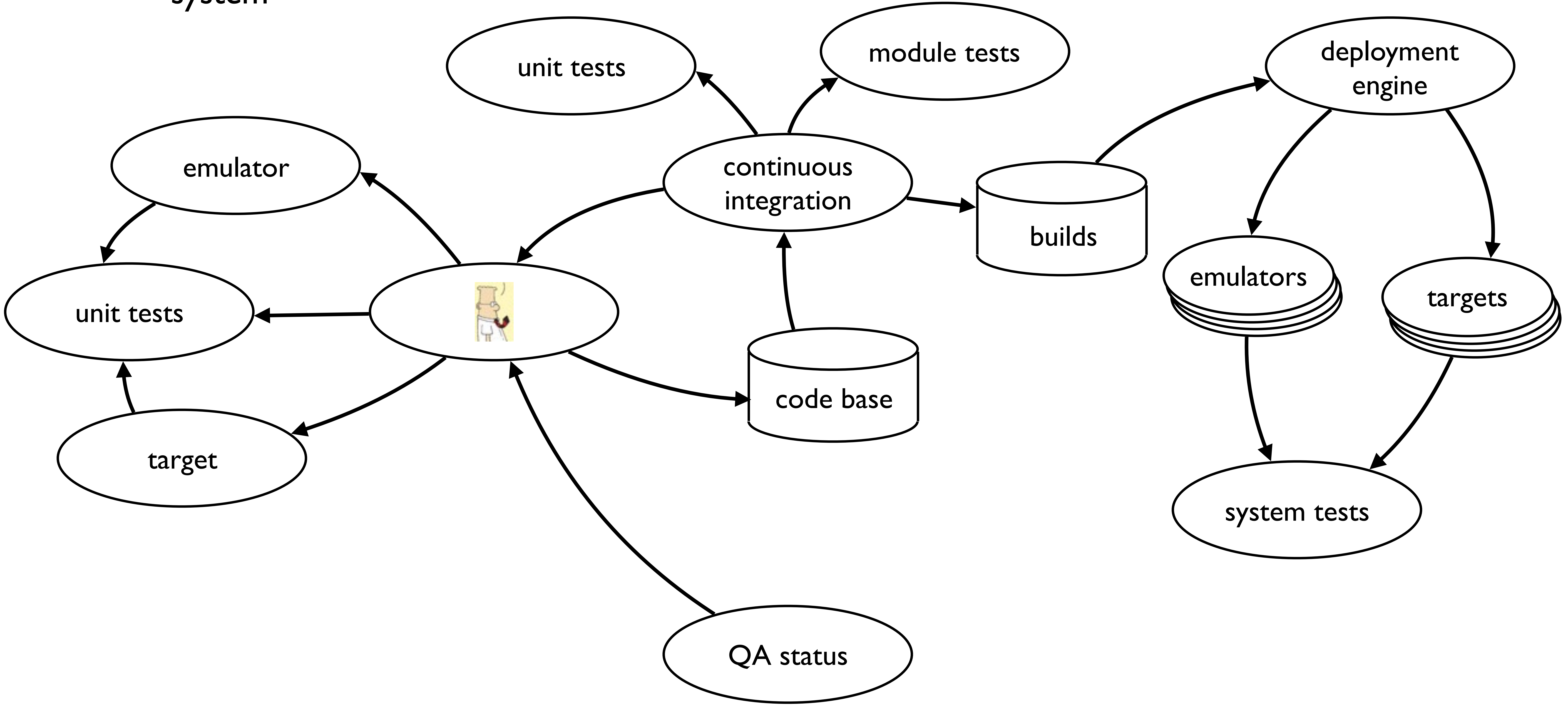
Continuous integration and deployment system



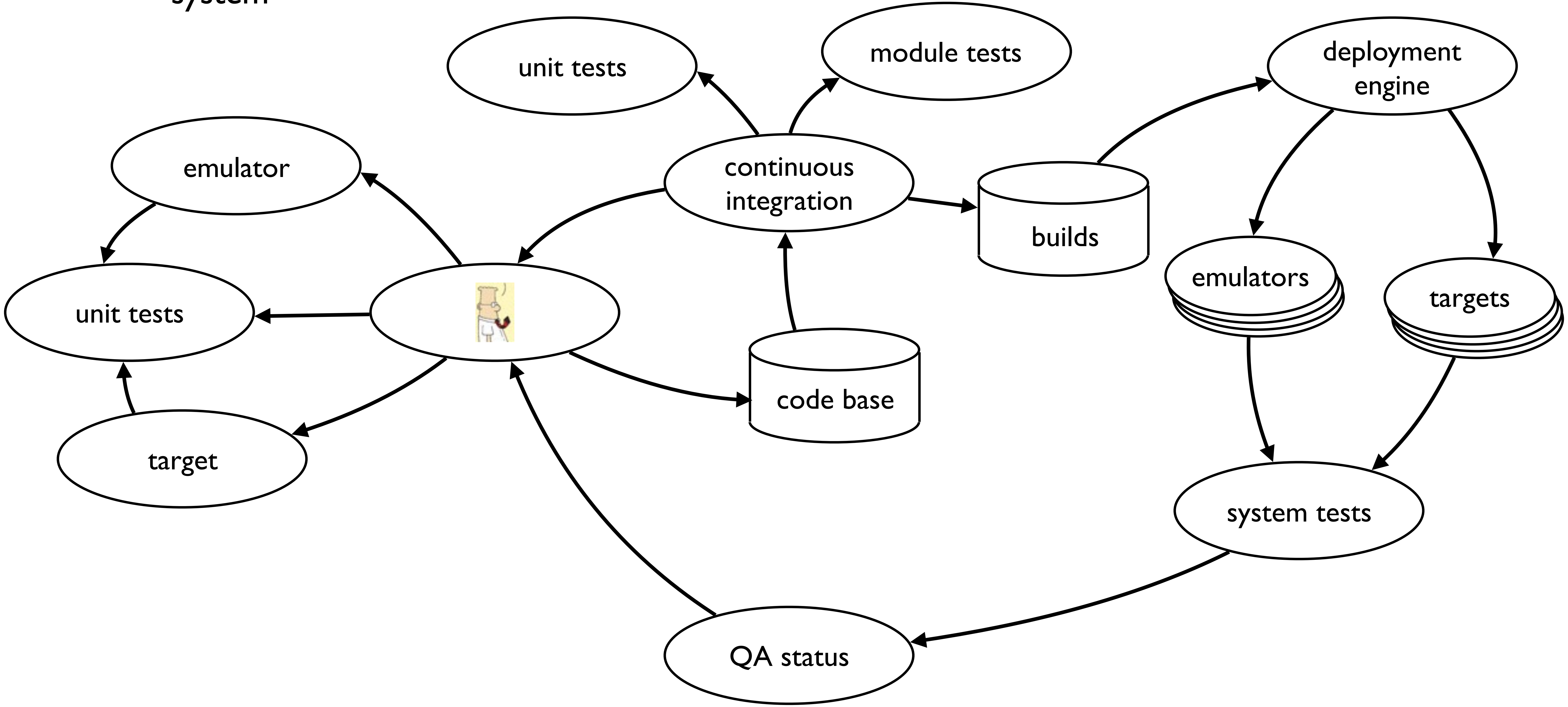
Continuous integration and deployment system



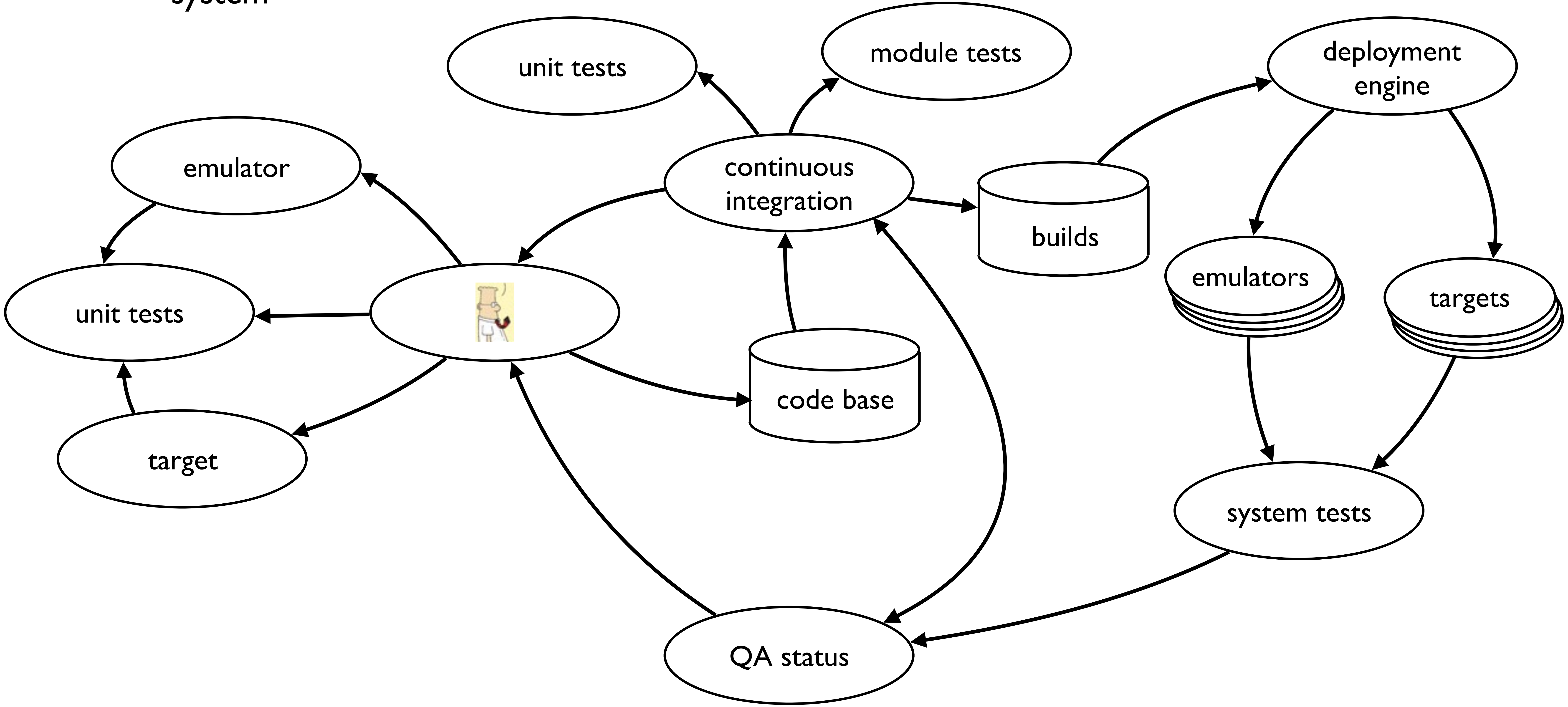
Continuous integration and deployment system



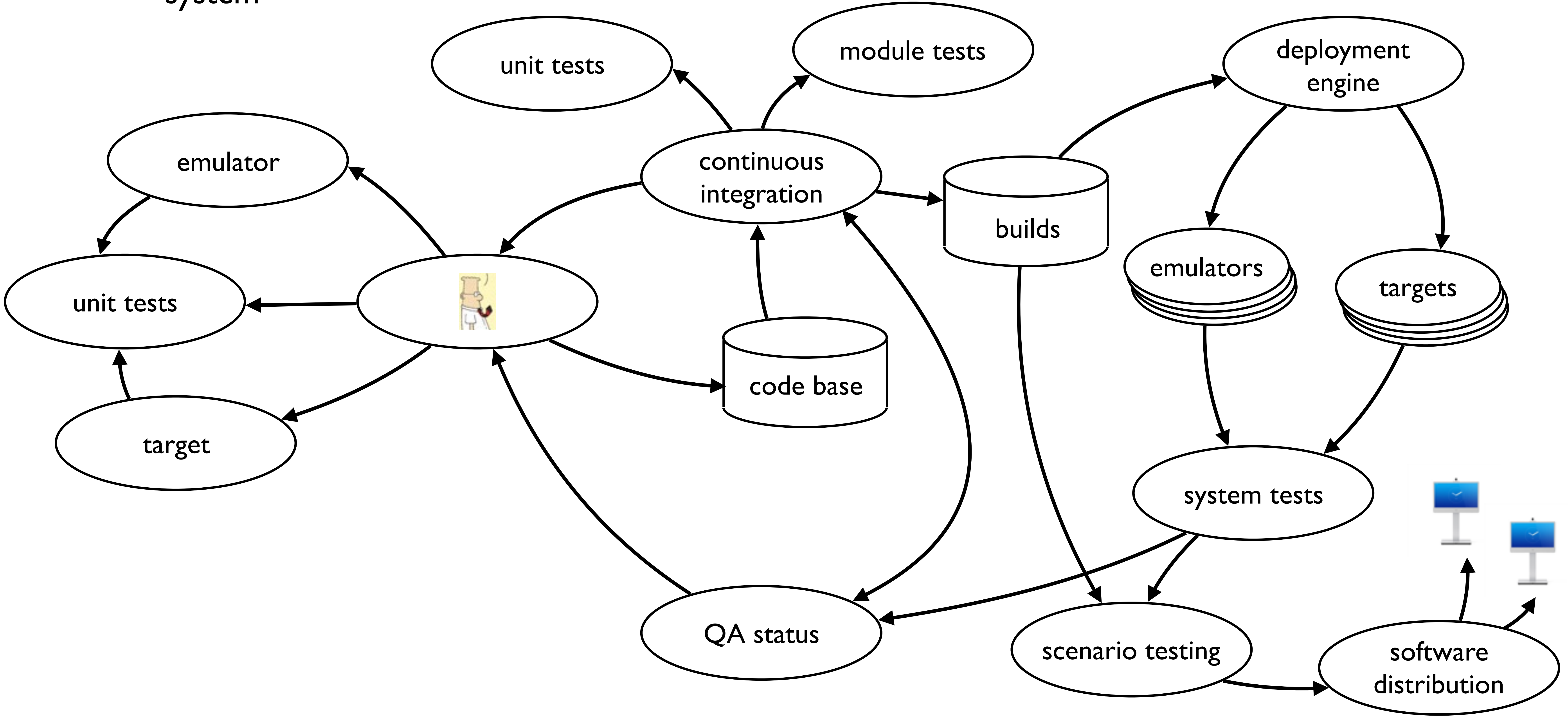
Continuous integration and deployment system



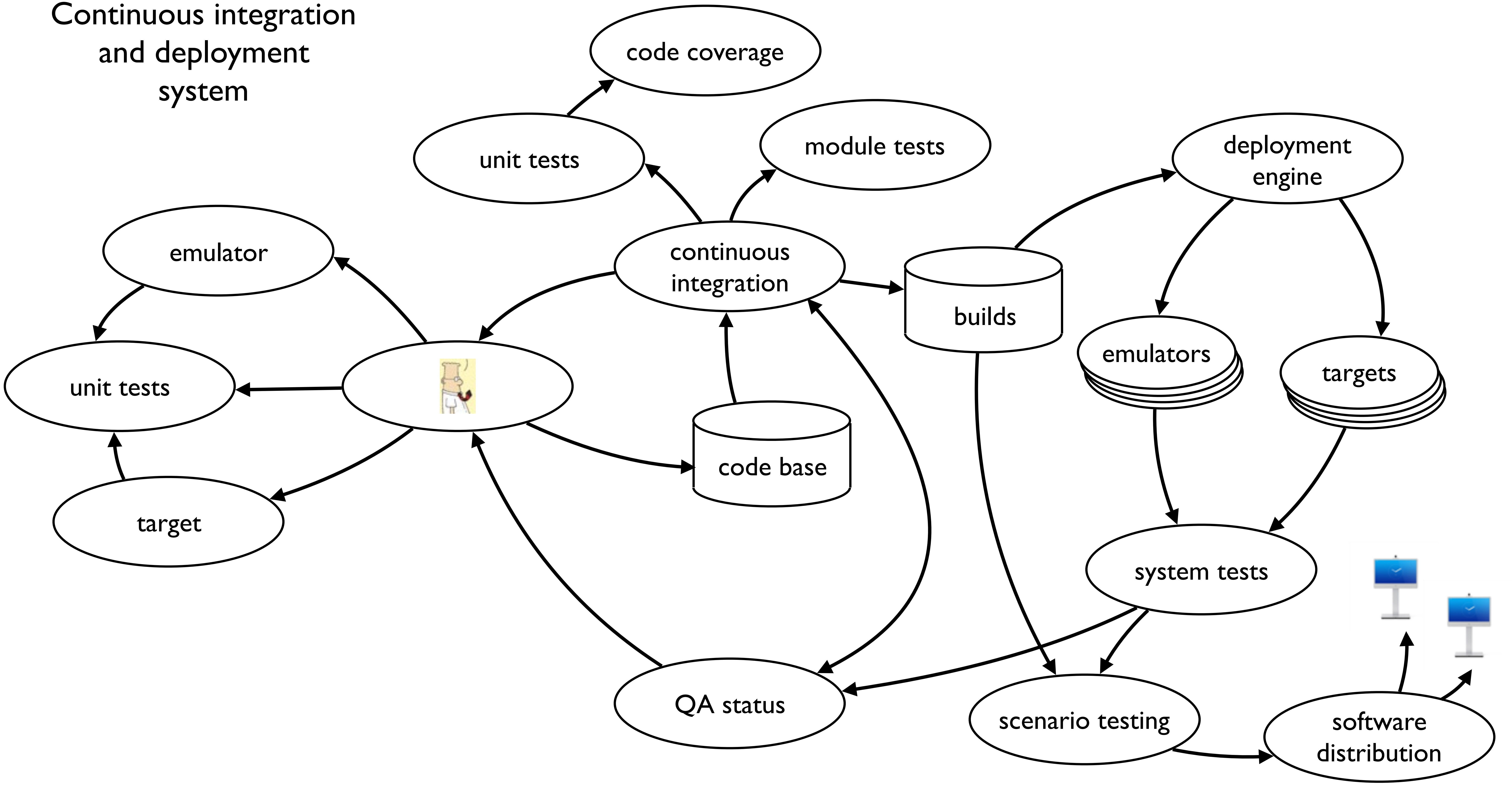
Continuous integration and deployment system



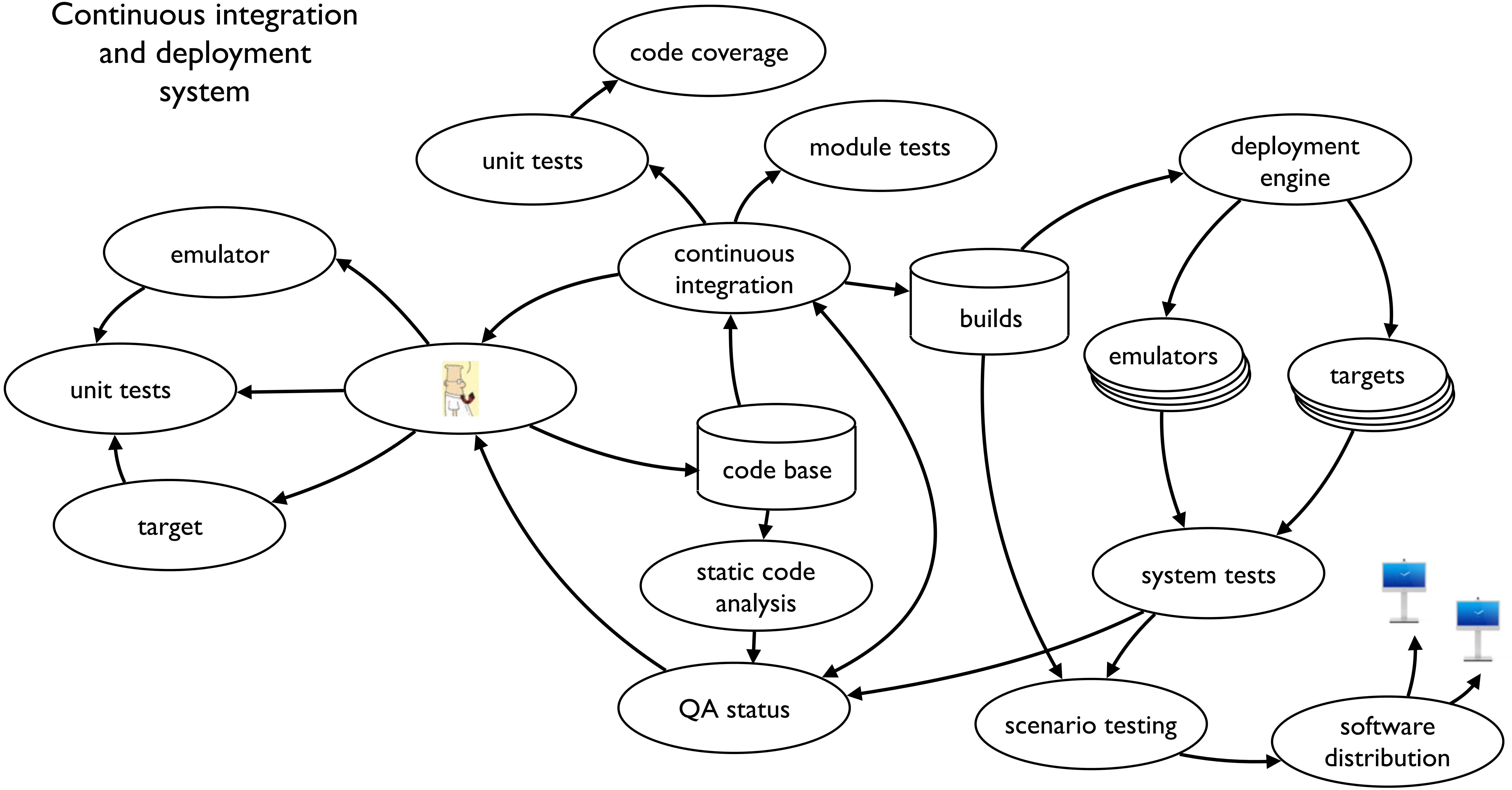
Continuous integration and deployment system



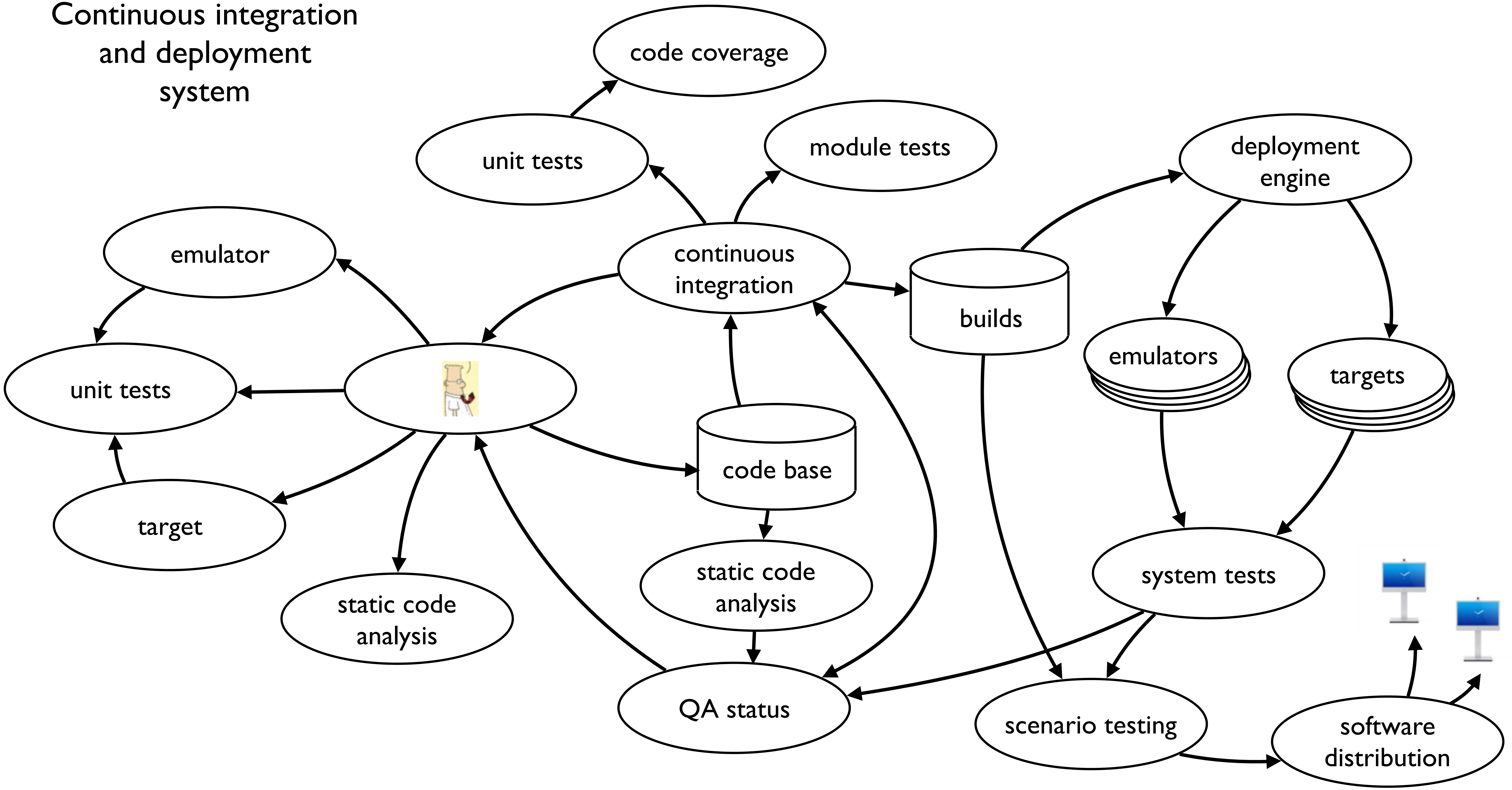
Continuous integration and deployment system



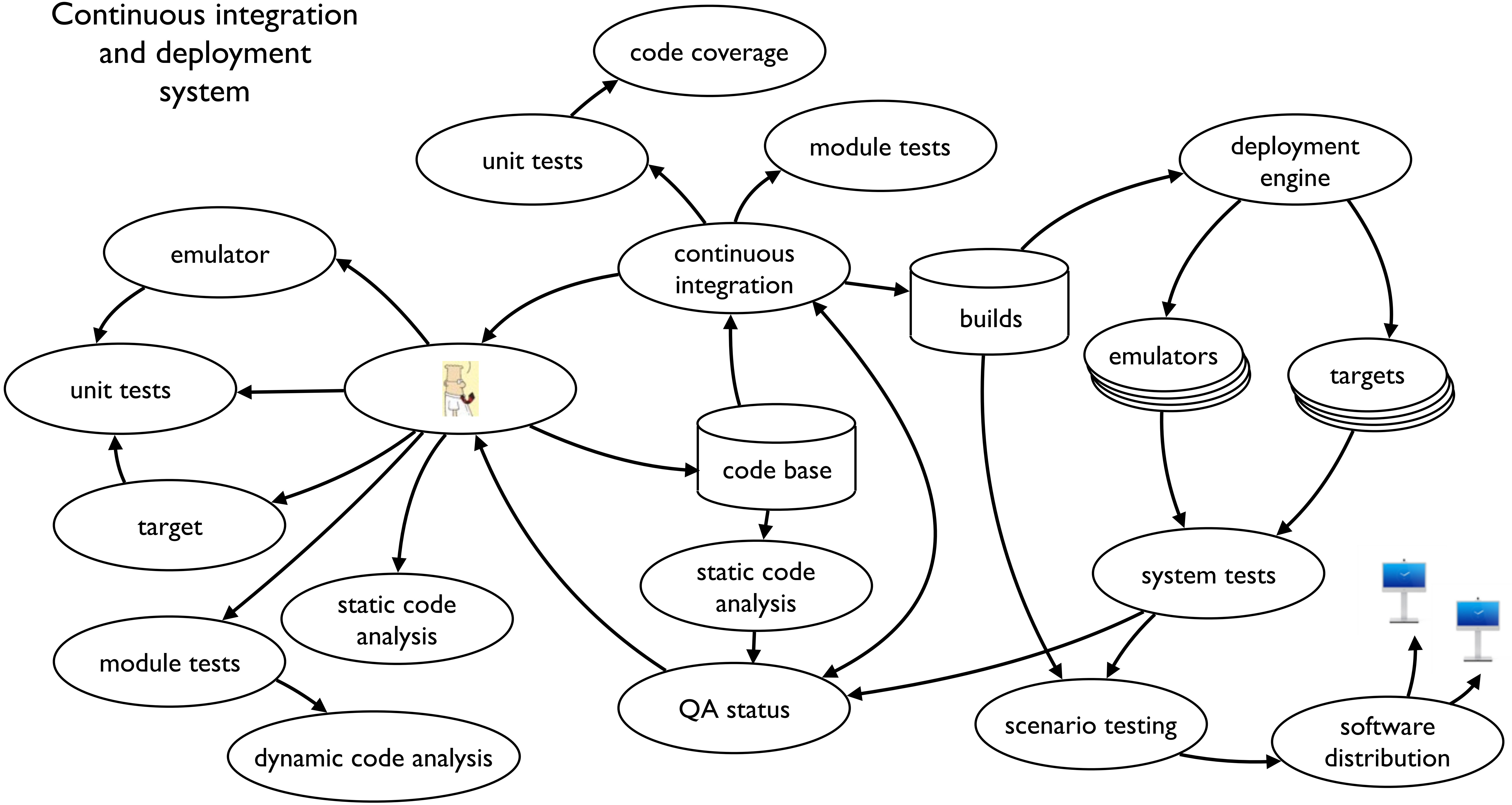
Continuous integration and deployment system



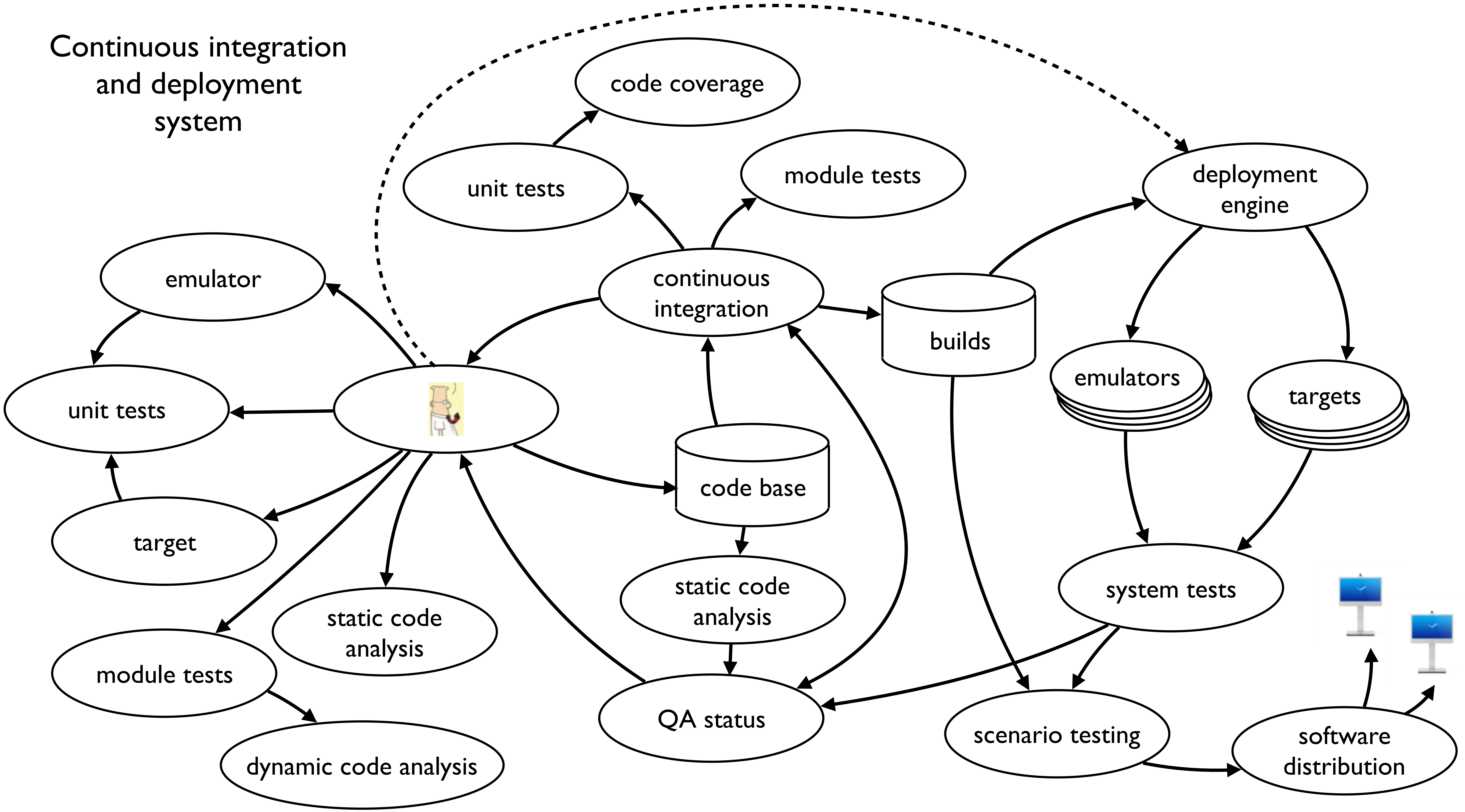
Continuous integration and deployment system



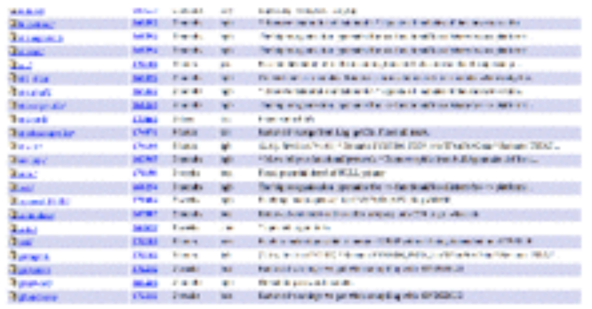
Continuous integration and deployment system

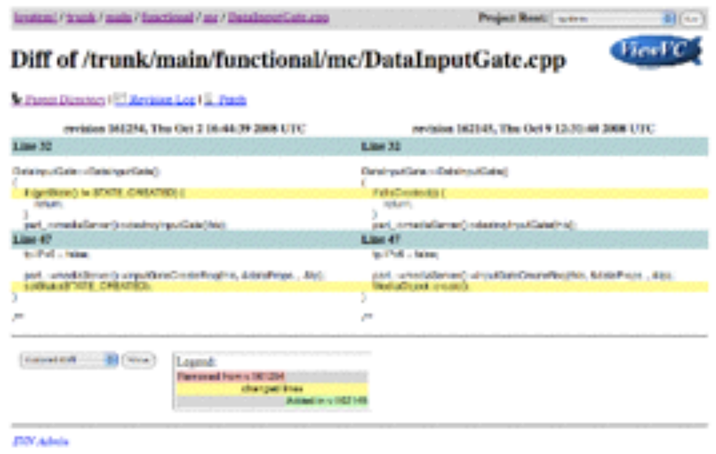


Continuous integration and deployment system



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


viewvc


Diff of /trunk/main/functional/mc/DataInputGate.cpp

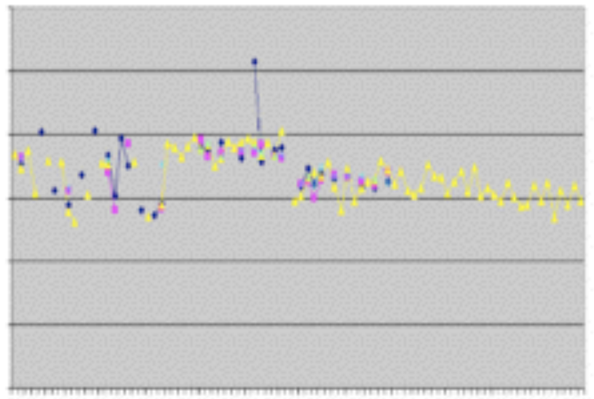
diff from viewvc

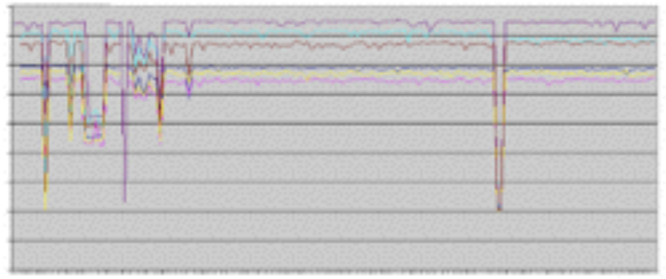
audio delay trend

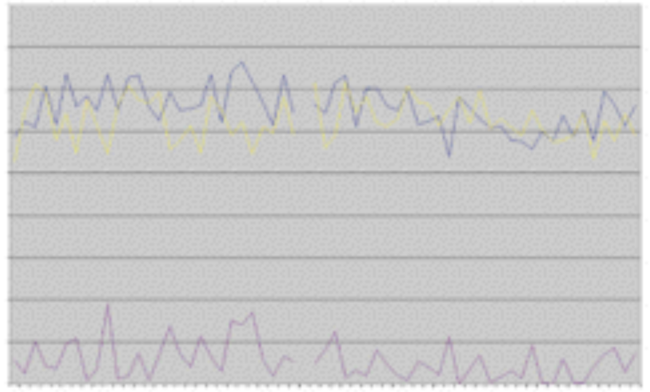
bugzilla

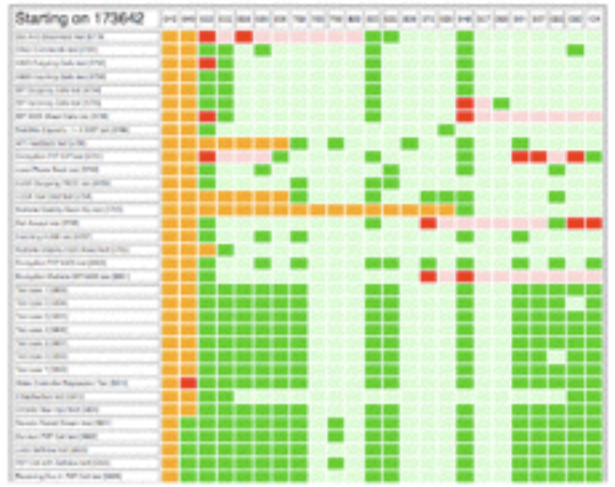
irc channel

continuous integration

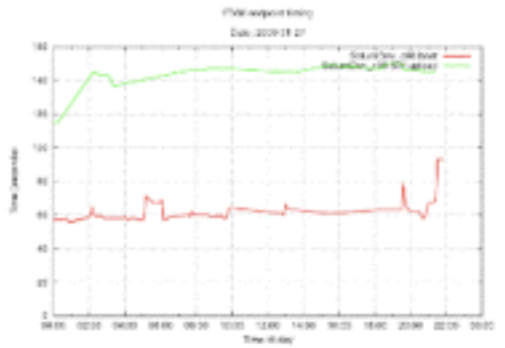

H.264 delay trend

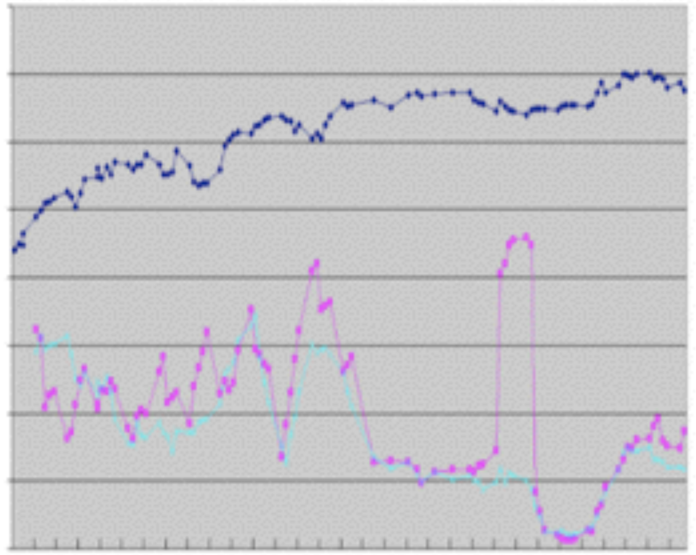

PESQ trend

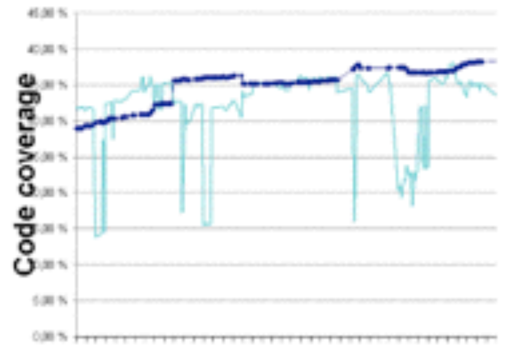

lipsync trend

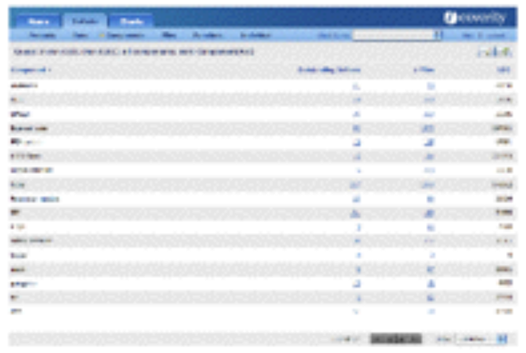

Starting on 173642

system tests

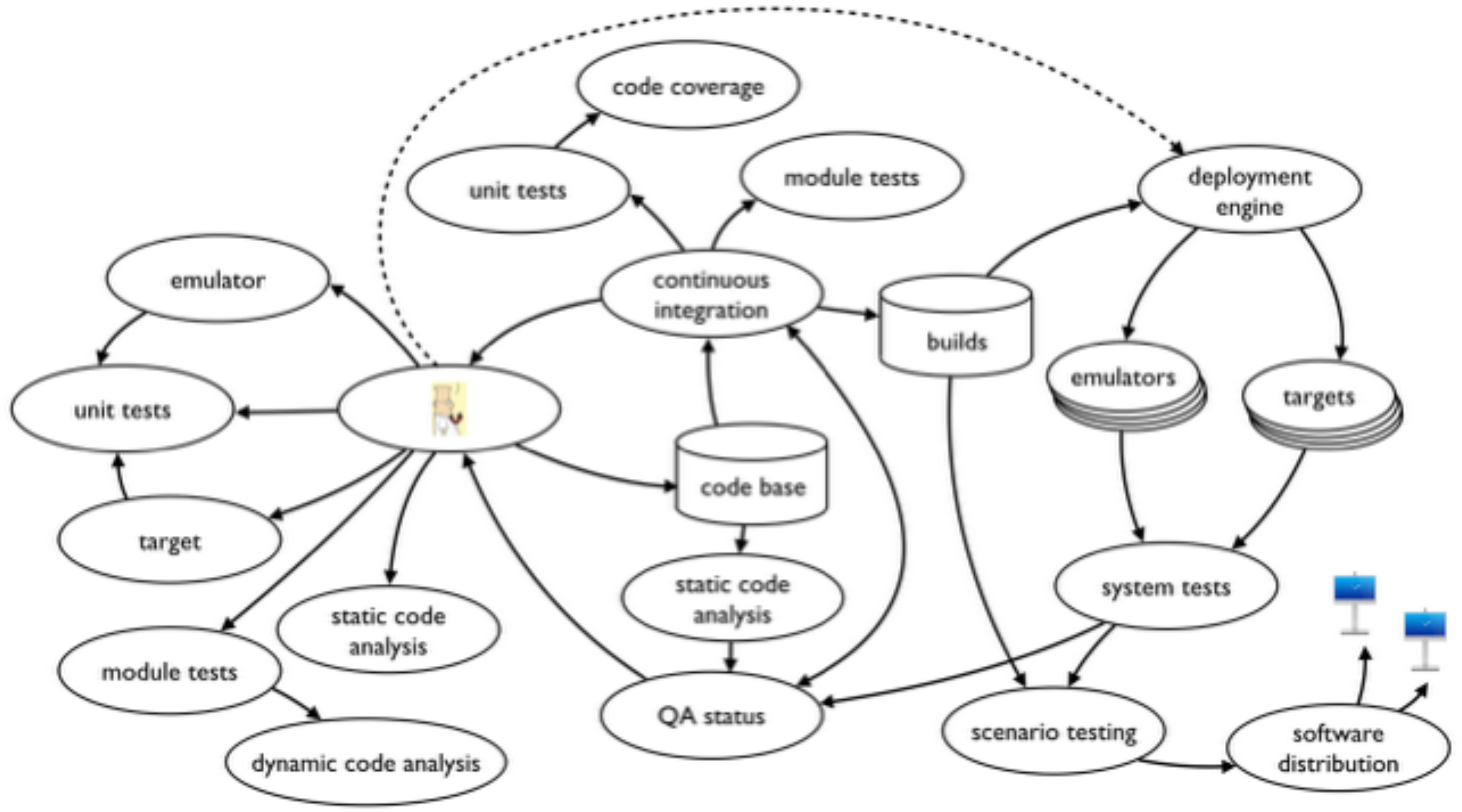

endpoint timing


defect trends




coverity

QA Status



We want something like that! Where do we start?

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

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`)

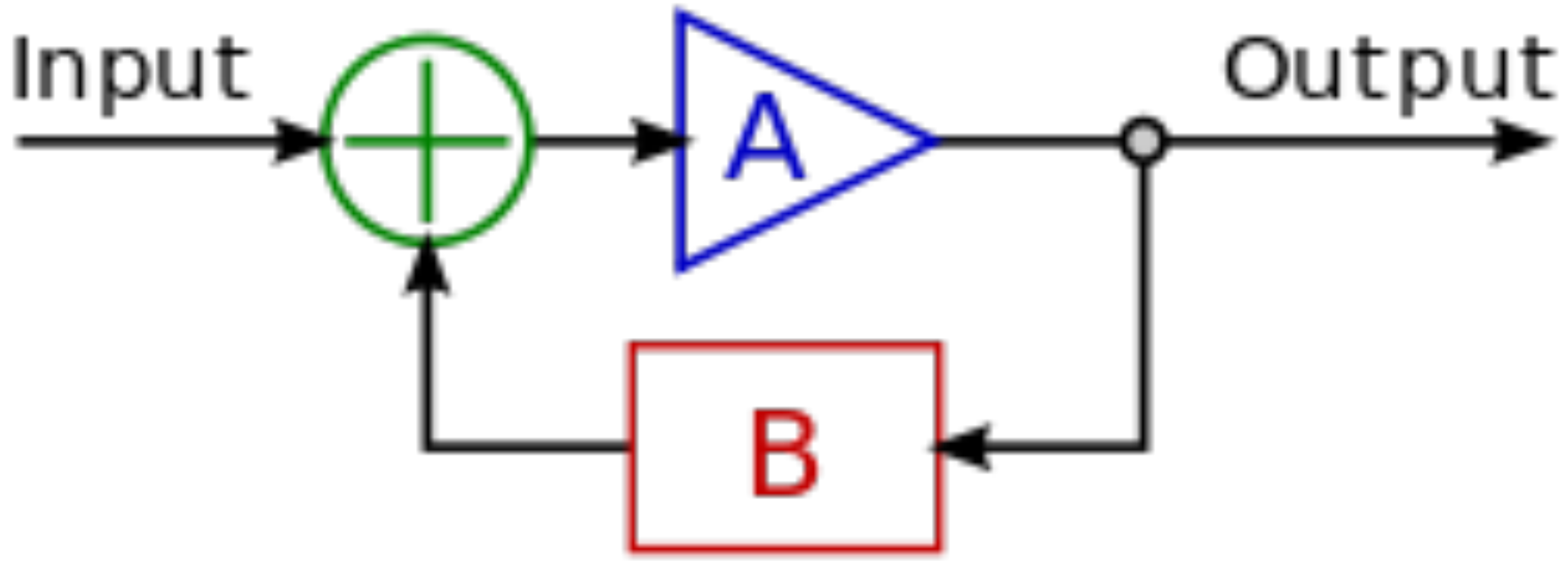
Integrate continuously

- Manual integration vs Automatic integration
- Beware of sandboxes (comfortable developers are leathal!)
- Continuous pain is the key to success
- Test everything, for all commits
- Superfast feedback
- Invest in equipment for fast and complete system testing
- Prune unused metrics and feedback mechanisms
- Slim down your QA department

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

!

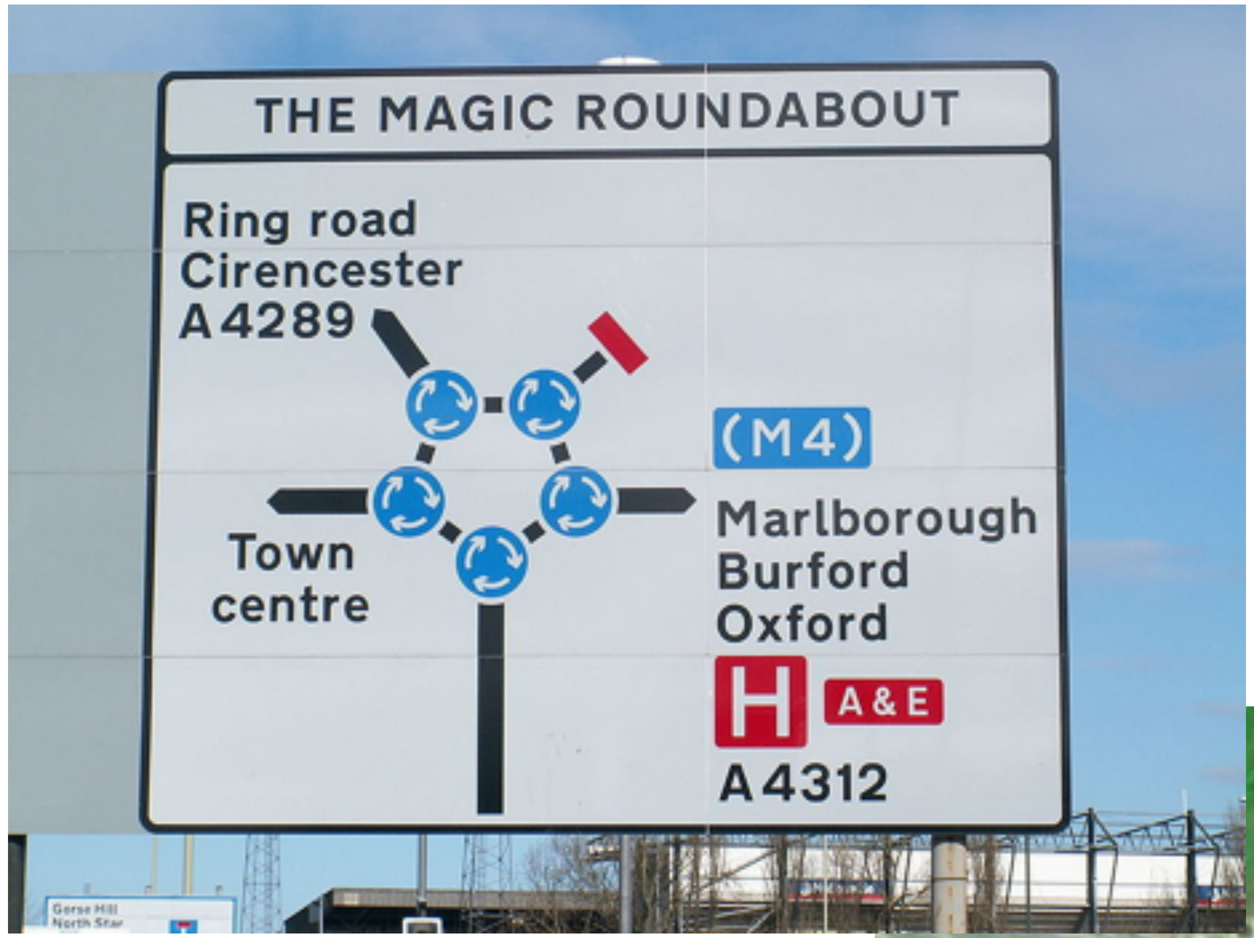

Input
A
Output
B

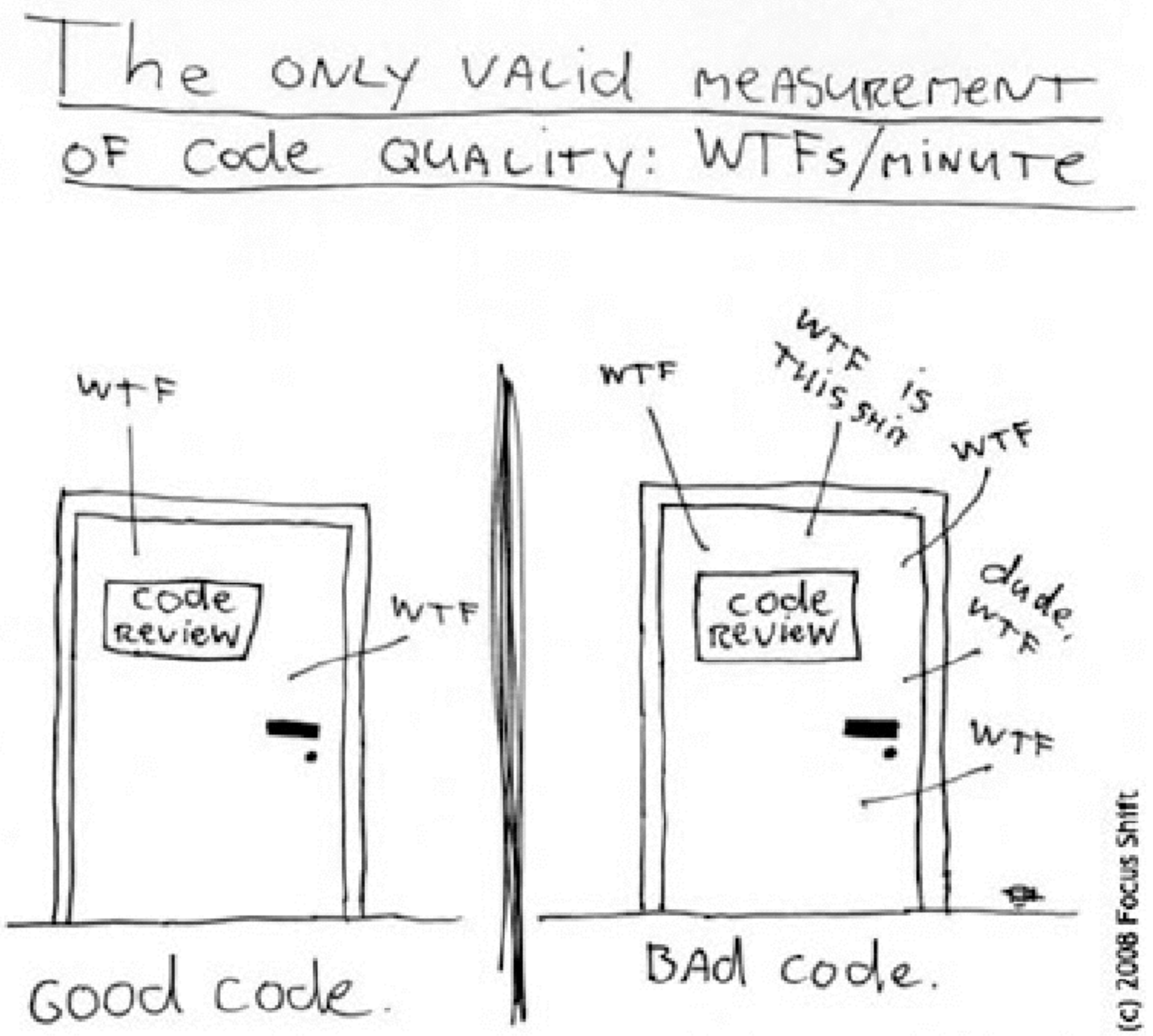
Make sure that everybody is working towards a common goal.

... WELL, SEEMS LIKE HIS PART OF THE BOAT IS ABOUT TO SINK

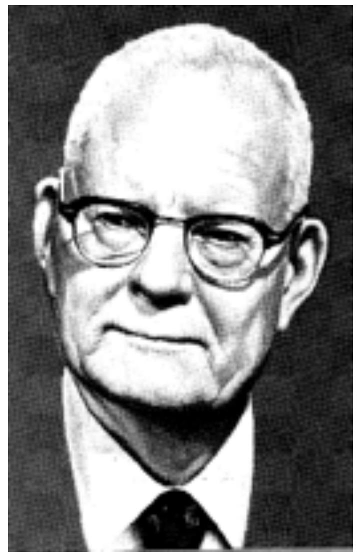
Control does not always work

Focus on flow





(reproduced with kind permission of Thom Holwerda)



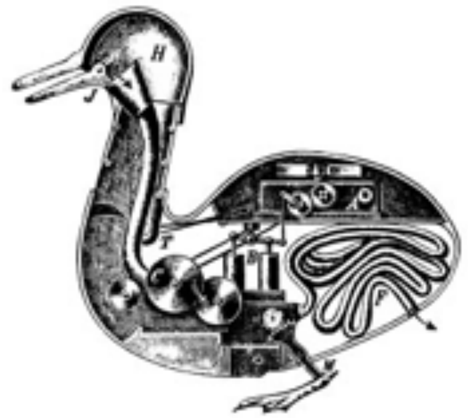
(Demingism)

(Taylorism)

Reductionism vs Systems thinking

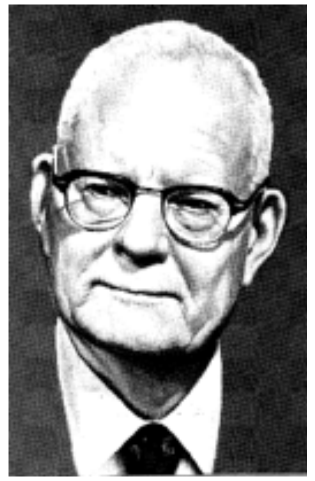
Reductionism is a philosophical position that a complex system is nothing but the sum of its parts, and that an account of it can be reduced to accounts of individual constituents.

Systems thinking is the process of understanding how things influence one another within a whole



(aka, Taylorism vs Demingism)

Frederick Winslow Taylor (1856-1915)



W. Edwards Deming (1900-1993)

Learn to surf, instead of trying to control the waves…

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

(Princess Leia)