

A photograph of two men in a meeting room. The man on the left has glasses and a beard, wearing a dark jacket. The man on the right is smiling, wearing a dark sweater over a light blue shirt. They are standing in front of a whiteboard covered in colorful sticky notes (yellow, orange, green, blue, purple). The room has fluorescent lighting and a modern office aesthetic. A large, semi-transparent grey graphic element is overlaid on the right side of the image.

Scrum

In theory & practice

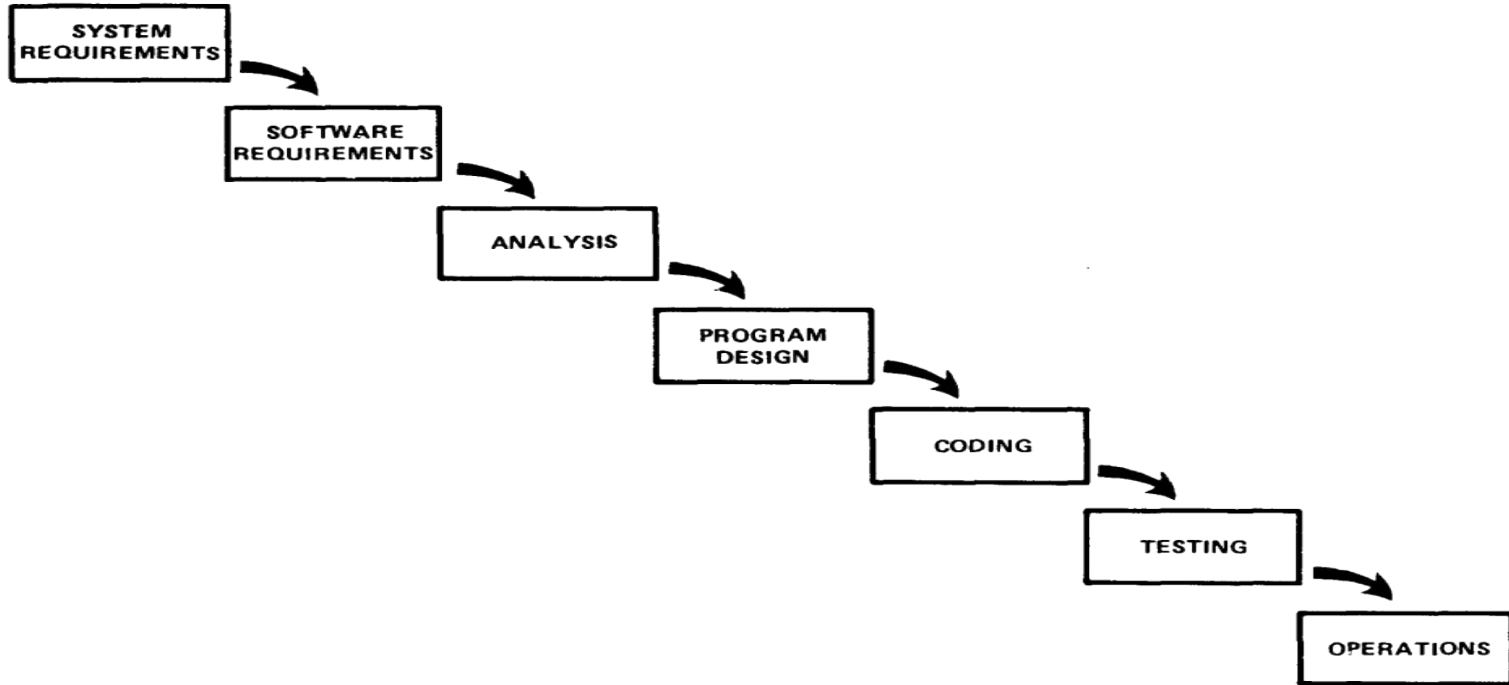
WHO?

- Visa Parviainen
- B.Sc. Metropolia 2010
- CSM 2010
- Scrum projects last 4 years
- Project manager
- Eficode
- Software company
- Software and software development tools & services

Scrum is not for just software projects. You can use it for **everything**.

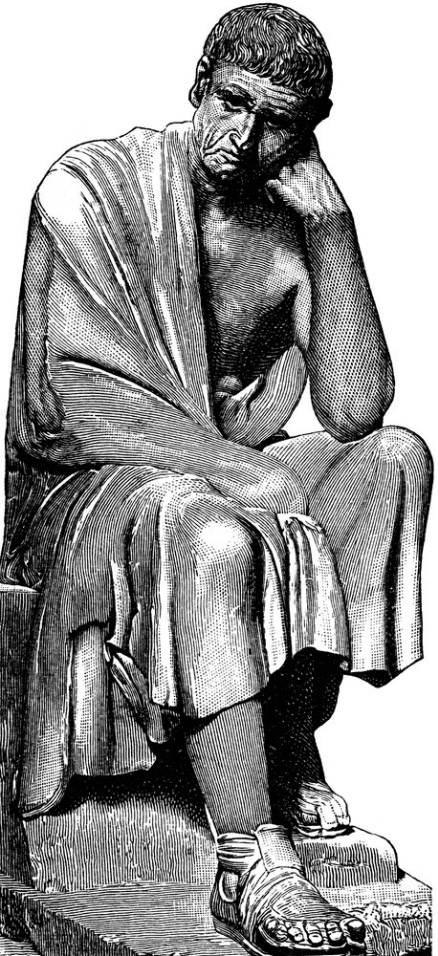


Waterfall...



Picture credit: Dr. Winston Royce

Waterfall?



*“the implementation described above
is risky and invites failure”*

Dr. Winston. W. Royce in *Managing the
Development of Large Software Systems*

Who *is* this Winston fella?

*“He was **the first** who described the Waterfall model for software development”*



Specification is hard.

Now get of your arses.

There are two kinds of
problems in the world.

Simple and complex.
Chili con Carne vs. bouncer



② web client (init)

- API reqs:
 - auth
 - current user (company)
 - company
 - projects
- company plugins
- layout request
- load ('layouts/page')

discuss something
stakeholders during
state transitions!

Software development: A complex problem

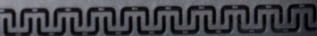
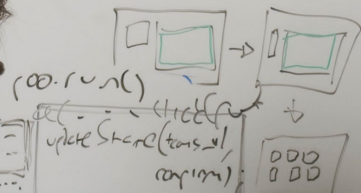
- user behavior
 - sudden changes/gaps/missing actions (e.g. Jerry completed <10% of tasks in 5-min tea this sprint)
 - epic heroic actions (e.g. NASSI completed >60 tasks in one day)
 - non-linear events (!)

- Plugin ID:
comment[content] match company
comment[project] = Okendet
→ plugin key

.scope .scope-project
client = {
 apps: {

□ = view
□ = layout
□ = client

type (method
necessary terms
auto-iterate die())
filetree:
- action (c, u, d)
- objects & subobjects
= all
= type
- type #1

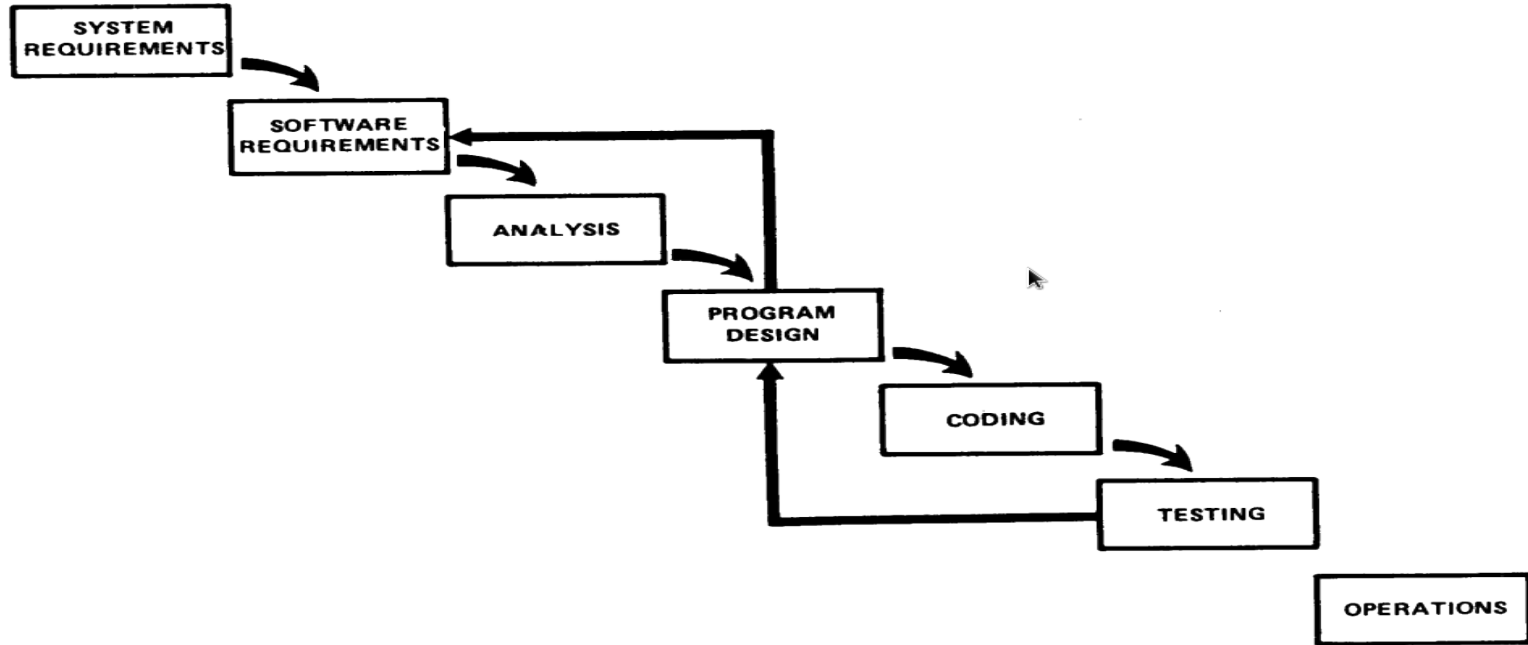




Iterations

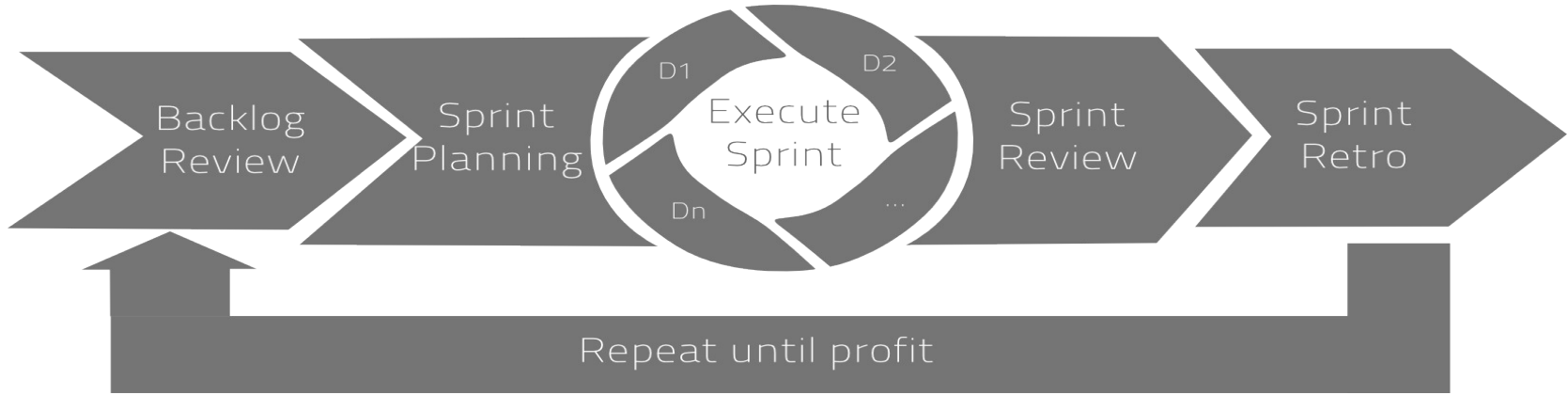
FTW!!!

Looping



Picture credit: Dr. Winston Royce

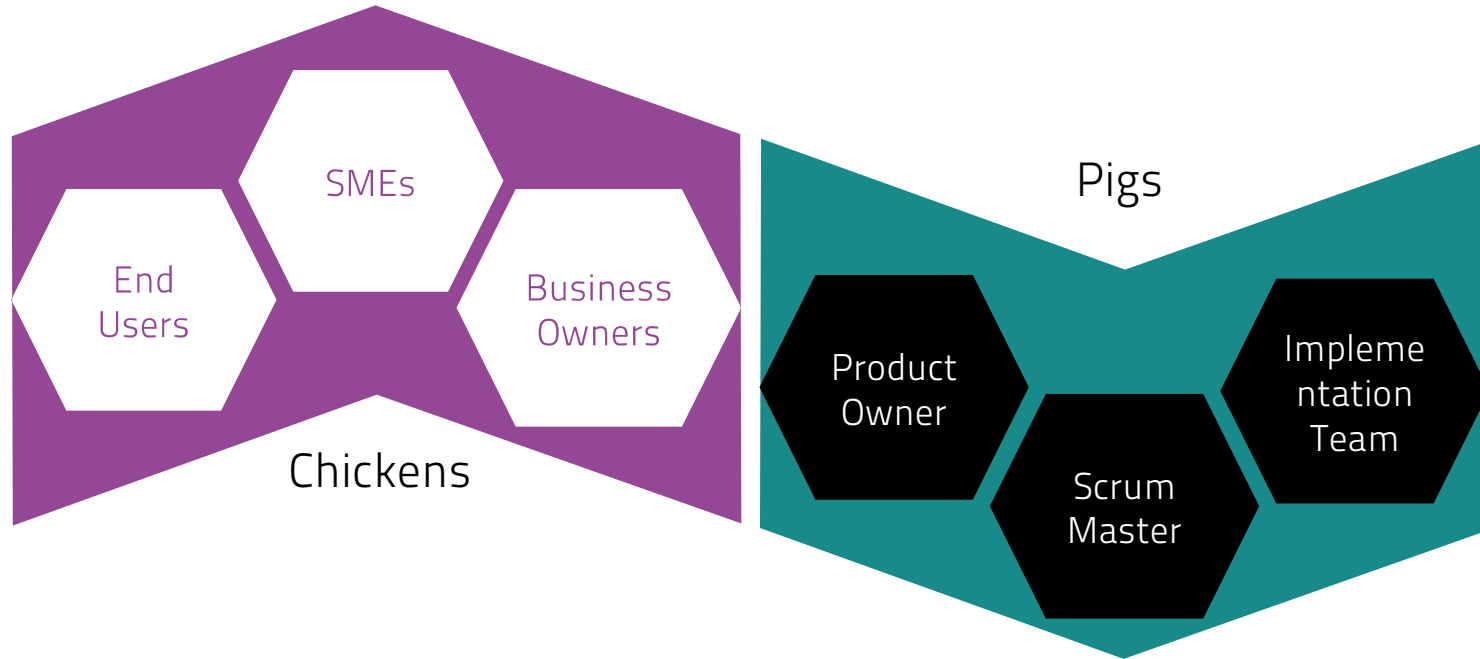
Sprint = Short iteration



What you need...



Those Animals!



User story



Backlog = List of work to
be done





Wall

Let's talk ready

When can we start working on a task?

What's done is done.

But how do you know it is?

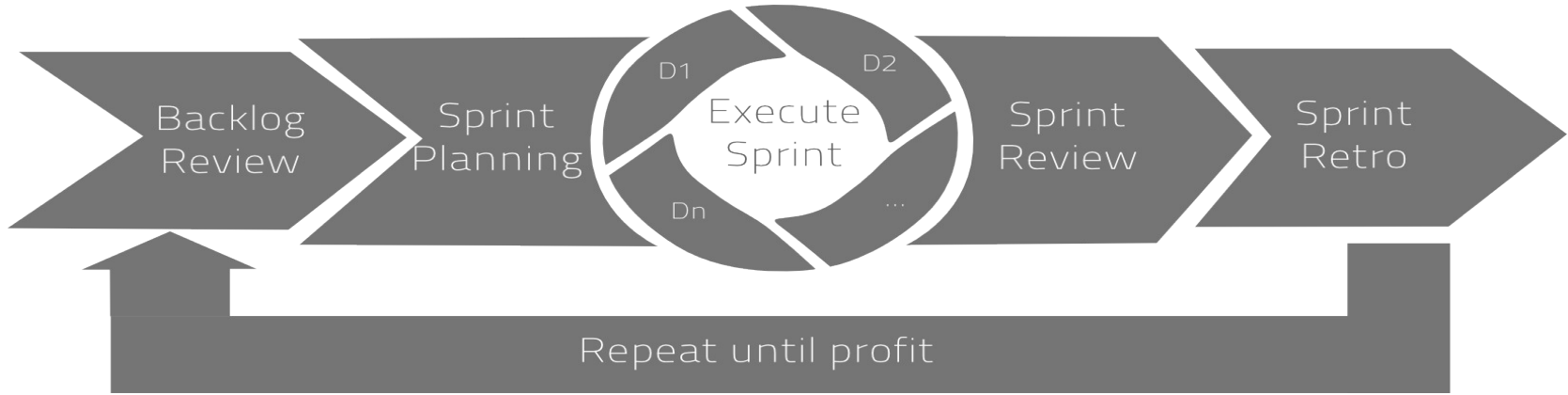
Tools

- Version control (Git ftw. (SVN?))
 - Agree on working methods!!!
- Development environment
- Testing & demonstration environment
- CI-environment
- Workstations
- Information sharing: Wikis, issue management systems etc.



Meetings **prior** to sprint

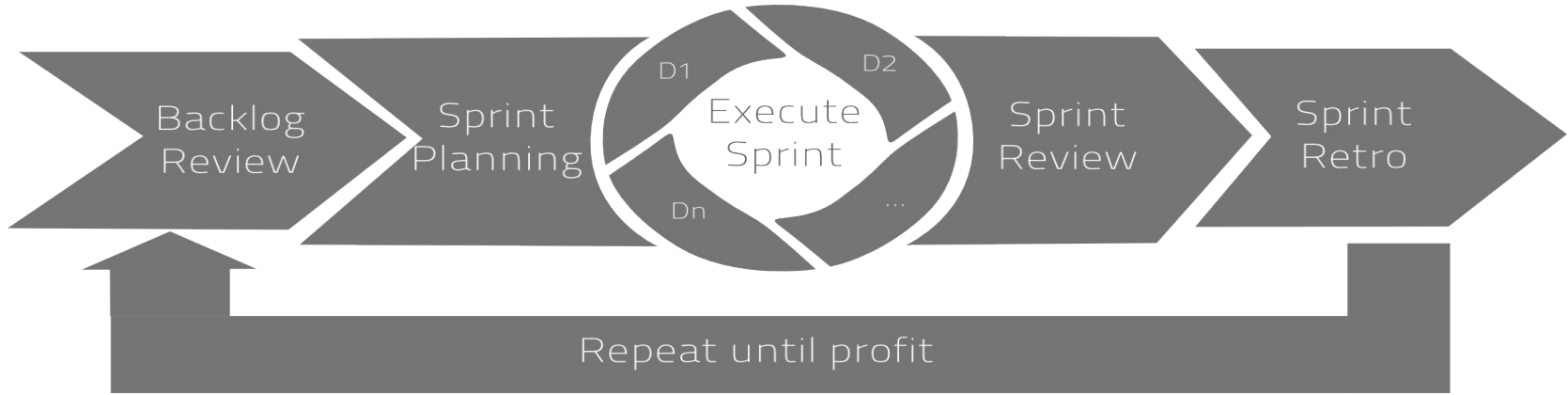
Short Iterations



Backlog Review

- Product owner's own show
- Vision of product into user stories
 - Forming acceptance criteria
- Prioritization of existing user stories
 - Determines what happens next

Short Iterations



Sprint planning

Team decides together what will be done next

The people who **actually do the work** decide how much work can be taken

Team **commits** to finishing the work in time

Team

Declines
to take
tasks that
are:

too vague

contain
too many
unknowns

too large
to be
done in
one sprint

Meetings **during** the sprint

Daily Scrum

- Pigs speak, Chickens listen
- What did I do?
- What will I do next?
- Did I run into any trouble or impediments?
- Do I need any help?

Other useful meetings



- AD-HOC chats
 - Good way to create architecture or principles in agile projects
- Workshops
 - Let's brake the product (safety)
 - UI workshops (common agreement on functionality)
 - Architecture and refactoring workshops

Remember: Decide what meetings you want to have before the sprint and stick with it!

Meetings **after** the sprint

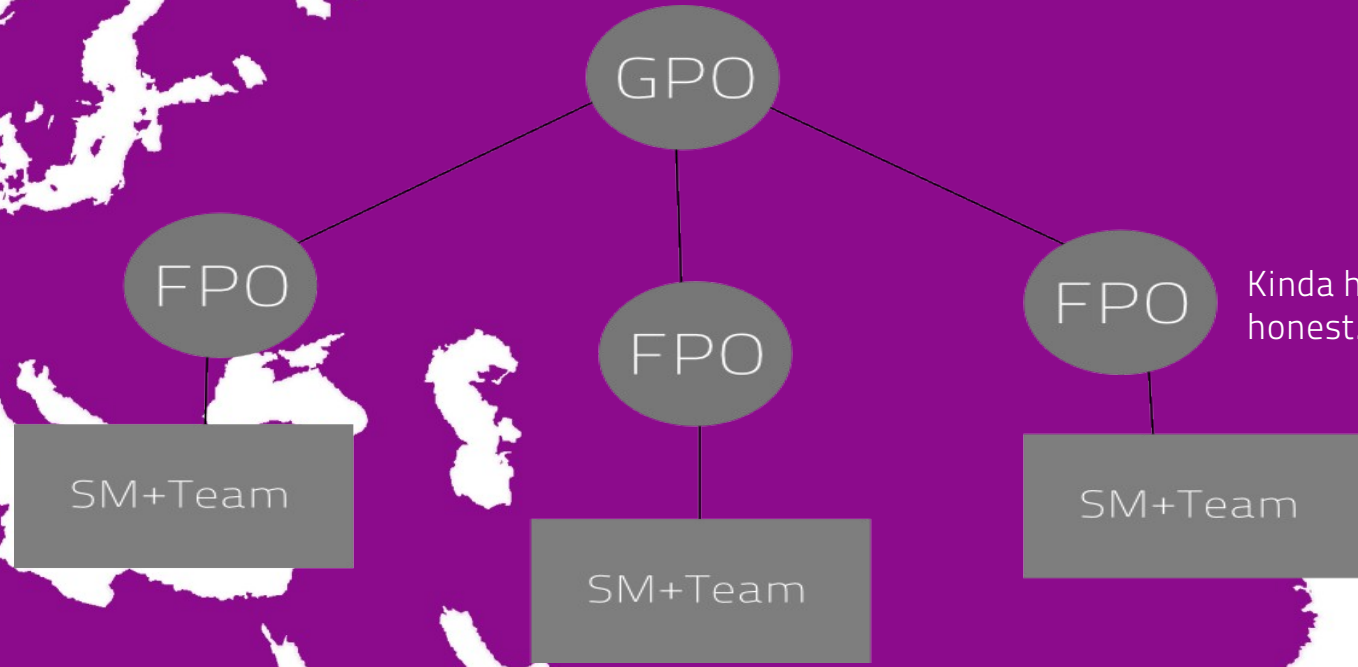
Sprint Review

- Show **for** the Product Owner/Business owners
 - What have we managed to complete
 - Team presents functionality to PO
 - How did we do?
 - Are they done right?
 - PO accepts/rejects the work done, or issues additional changes as new user stories.

Sprint Retrospective

- Team's own show
- What went well?
- What went badly?
- What could be improved?
- From last retrospective, which issues are still current?
 - Have all of them been worked on, if not, why?

Scaling



Practical tips

- Sprint 0 > 0!
- Estimations through story points = yes.
 - SM makes statistics after each sprint
 - Use statistics to estimate how much work for next sprint
- Using your sense in planning = allowed
 - Not too many overlapping stories
 - No stories with too many unmet dependencies

Practical tips 2

- TDD FTW!
 - Feels useless until you refactor the first time (and you will)
 - Good method for learning:
 - Senior writes test, junior codes, senior checks and gives feedback
- ... Which reminds me: Code review FTW!
 - Implement-critique-fix -cycle





The logo consists of a white, rounded pentagonal shape centered on a solid green background. Inside this shape, the word "eficode" is written in a dark green, lowercase, sans-serif font.

eficode



eficode