openCX – a case study

FEUP-MIEIC-ESOF-2019-20

Ademar Aguiar

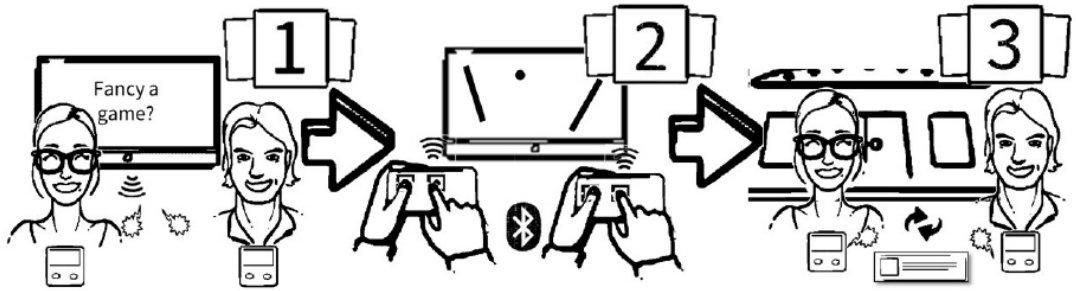
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

openCX

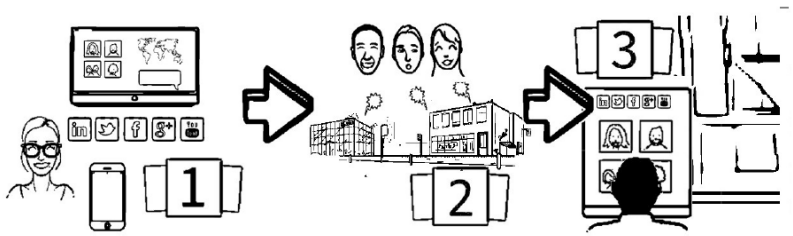
- openCX is an integrated set of apps aimed to improve the collective experience (CX) of participants attending professional conferences, group meetings and public spaces in general.
- It explores technologies in the areas of Internet of Things (IoT) and multimedia visualization applicable to distributed services and for creating augmented spaces.

- Interactive Agenda

- Interactive Badges



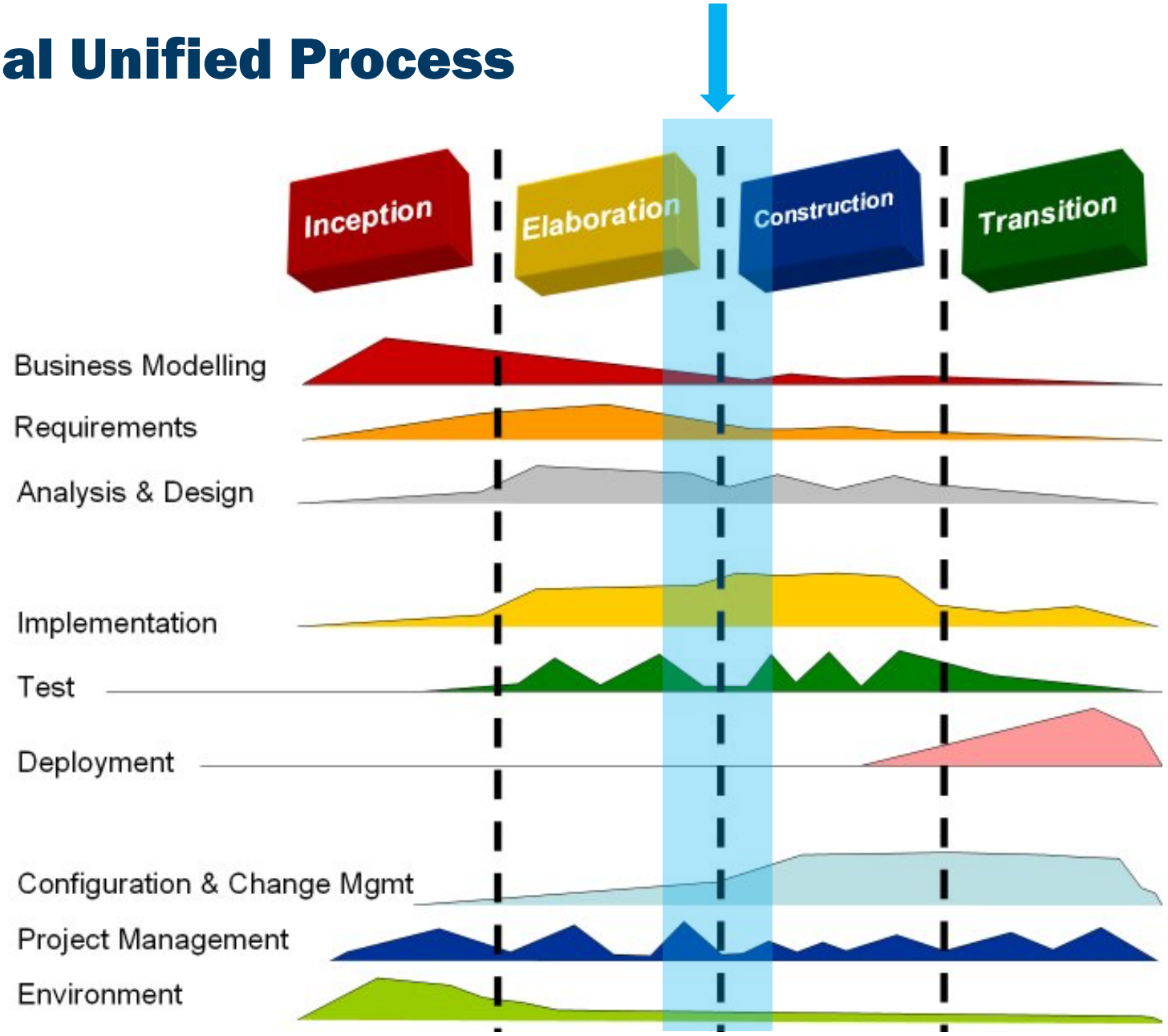
- Social Media Visualizer



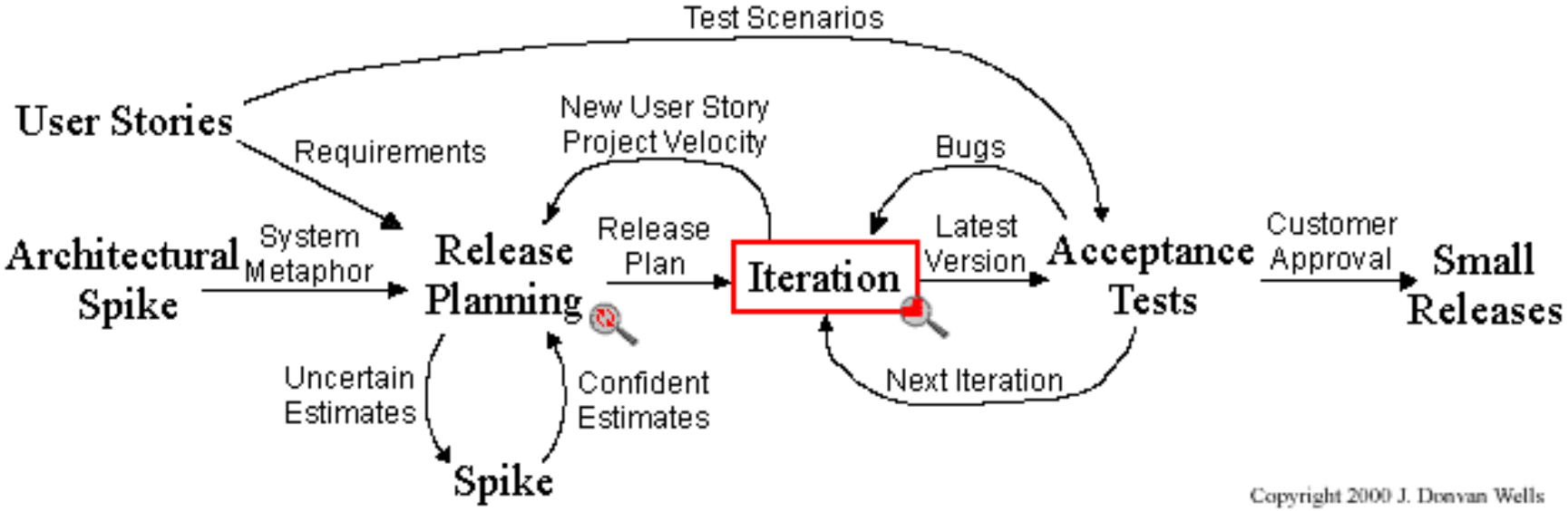
Outline

- Status: RUP and XP perspectives
- Key XP Practices for openCX
  - Simple Design
  - Pair Programming
  - Refactoring
  - Test Driven Development
  - Continuous Integration
  - Collective Code Ownership

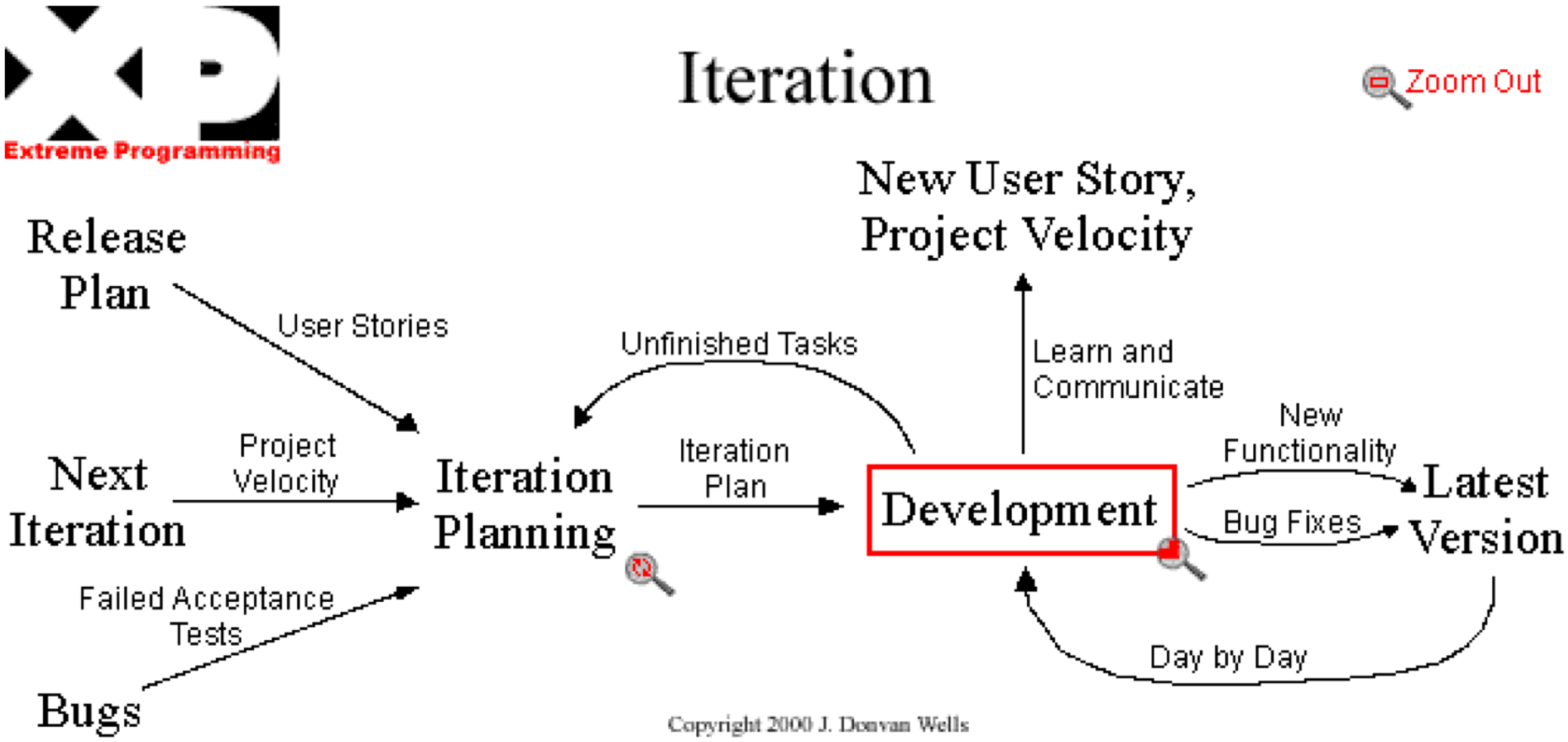
Rational Unified Process



Extreme Programming Project



Copyright 2000 J. Donvan Wells



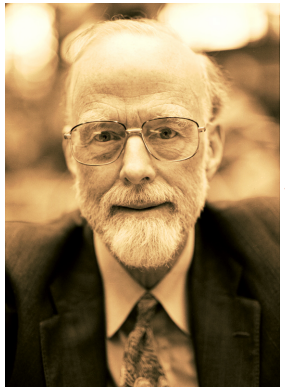
Release planning: Relase content & date

Simplicity

- "**Do the simplest thing that could possibly work**" (DTSTTCPW) principle
  - Elsewhere known as KISS (Keep It Simple, Stupid)
- A coach may say DTSTTCPW when he sees an XP developer doing something needlessly complicated
- "**You ain't gonna need it**" (**YAGNI**) principle
- Simplicity and communication support each other (how?)

Simple Design

- Use the simplest possible design that gets the job done
- The requirements will change tomorrow, so only do what's needed to meet today's requirements (remember YAGNI)
- Avoid big up-front design



Tony Hoare, Turing Award Lecture, 1980

I conclude there are two ways of constructing a software design:

one way is to make it so simple that there are obviously no deficiencies,

and the other way is to make it so complicated that there are no obvious deficiencies.

Pair Programming

- How it works:
  - Two programmers work together at one machine
  - Driver enters code, while navigator critiques it
  - Periodically switch roles and pairs
  - Requires proximity in lab or work environment
- Advantages:
  - Serves as an informal review process
  - Helps developing collective ownership and spread knowledge
  - Improves quality (less defects, better design), whilst maintaining (or improving) productivity
- Note: If you don't do XP/PP, at least do peer reviews

Refactoring

- Recently updated to “Design Improvement”
- Refactoring = improve the structure of the code without changing externally visible behavior
- E.g., refactor out any duplicate code generated in a coding session
- Refactoring and automated tests go hand-in-hand (why?)
- Simple design and continuous design improvement (with refactoring) go hand in hand (why?)

Test-driven Development

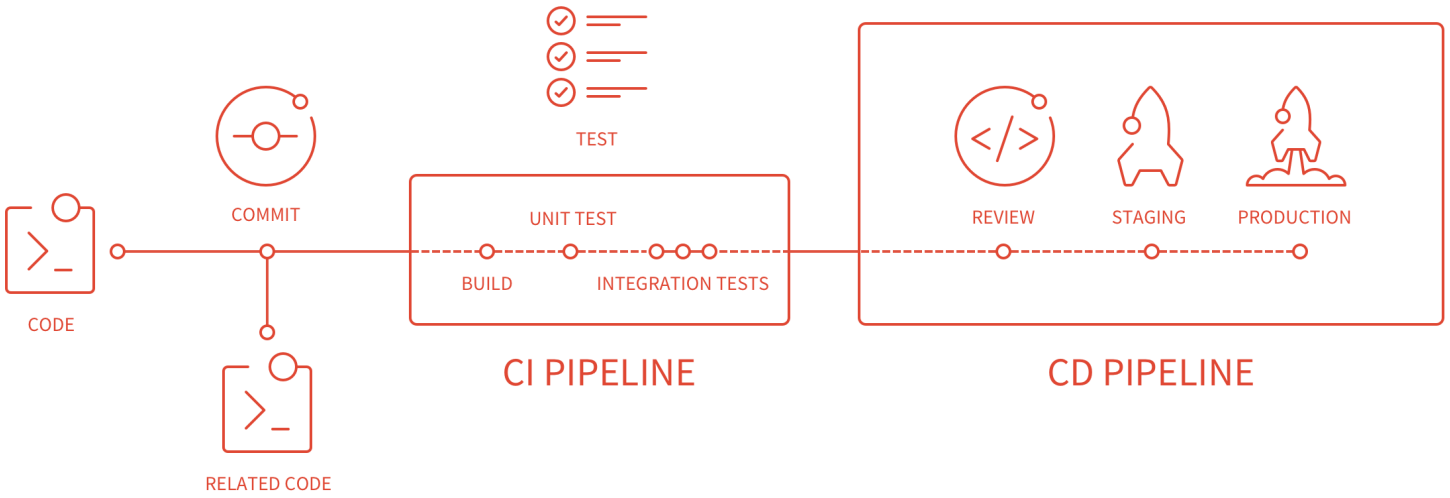
- **Test first:** before adding a feature, write a test for it!
  - If code has no **automated** tests, it's assumed it doesn't work
  - Also create test for bugs discovered, before fixing them
- **Unit Tests** (or **developer tests**): testing of small pieces of functionality as developers write them
  - Usually for testing single methods, classes or scenarios
  - Usually automated with a unit testing framework, like JUnit
  - Experiments show that TDD reduces debugging time
- **Acceptance Tests** (or **customer tests**): specified by the customer to check overall system functioning
  - A user story is complete when all its acceptance tests pass
  - Usually specified as scripts of UI actions & expected results
  - Ideally automated with a UT or AT framework, like FIT

Collective Code Ownership

- What it means:
  - No single person "owns" a module
  - Any developer can work on any part of the code base at any time
- Advantages:
  - No islands of expertise develop
  - All the developers take responsibility for all of the code
  - Pressure to create better quality code
  - Change of team members is less of a problem

Continuous Integration

- All changes are integrated into the code base at least daily
  - As opposed to “big-bang integration”
- Tests have to run 100% before & after integration
- Enables frequent releases



GitLab continuous integration & continuous delivery pipeline

openCX - FAQ

- beacons, beacons, beacons: GPS, QRs, metro-type-lines
- microbit's: bluetooth versus radio??????
- backend: ask and we will do it!
- databases on mobile -> sqlite
- ...