VG Software Development Methodology

by Manel Rello

GAMELOFT

A videogame is not a piece of software that needs art, it is a piece of art that needs software

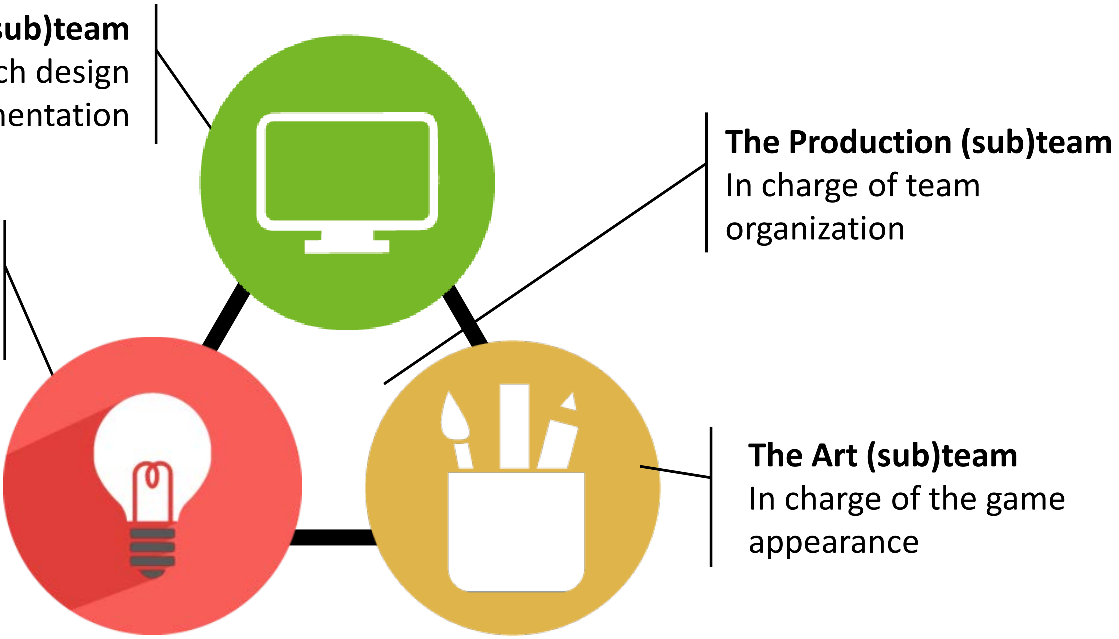
Summary

- Environment
  - Team organization
  - Project Phases
- Prototyping a Game
- Developing a Game
- Updating a Game

Environment

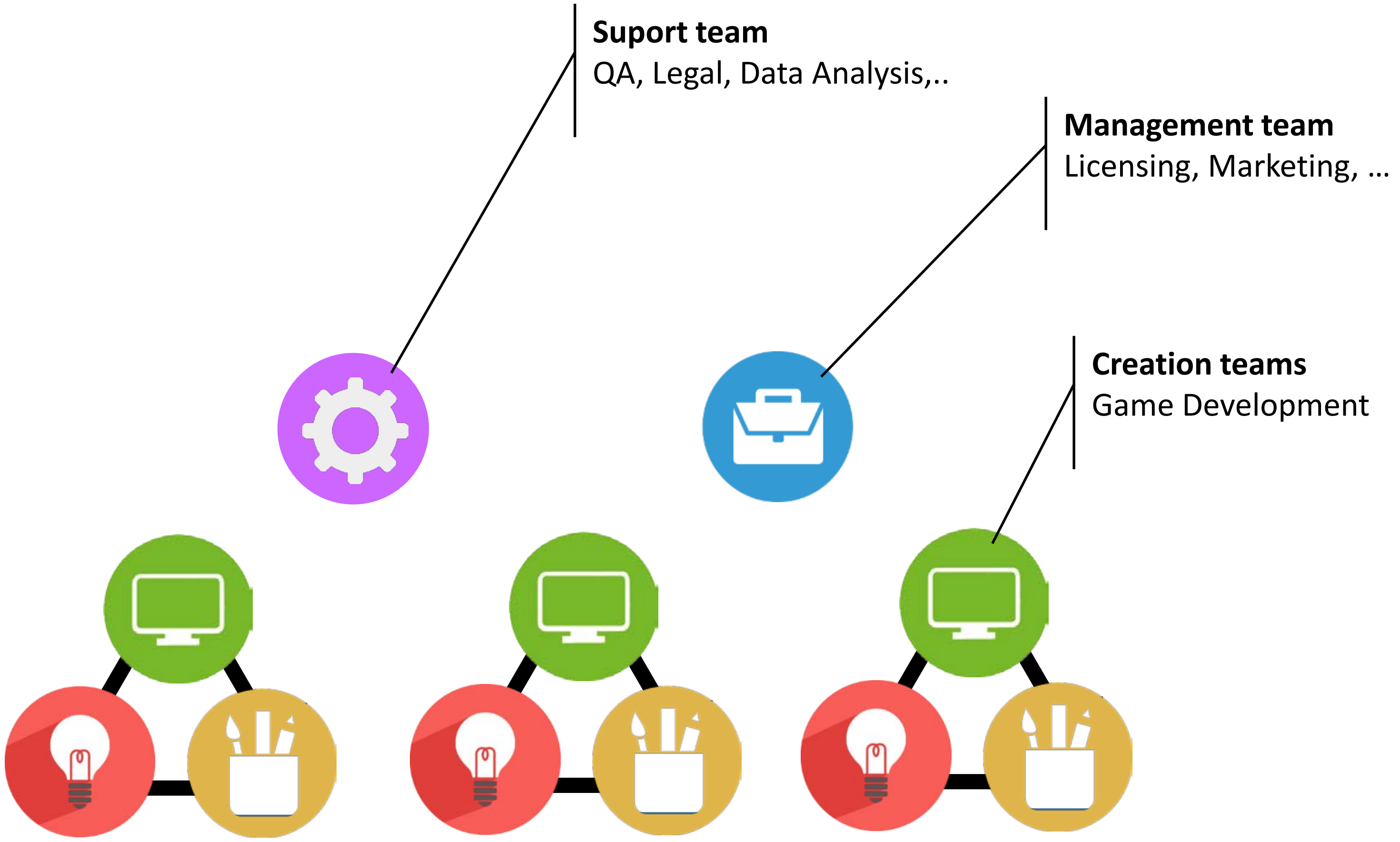
Team Organization

- Each Videogame is developed by a multidisciplinary team


The Engineering (sub)team
In charge of the tech design and implementation
The Production (sub)team
In charge of team organization
The Design (sub)team
In charge of the game mechanics and experience
The Art (sub)team
In charge of the game appearance

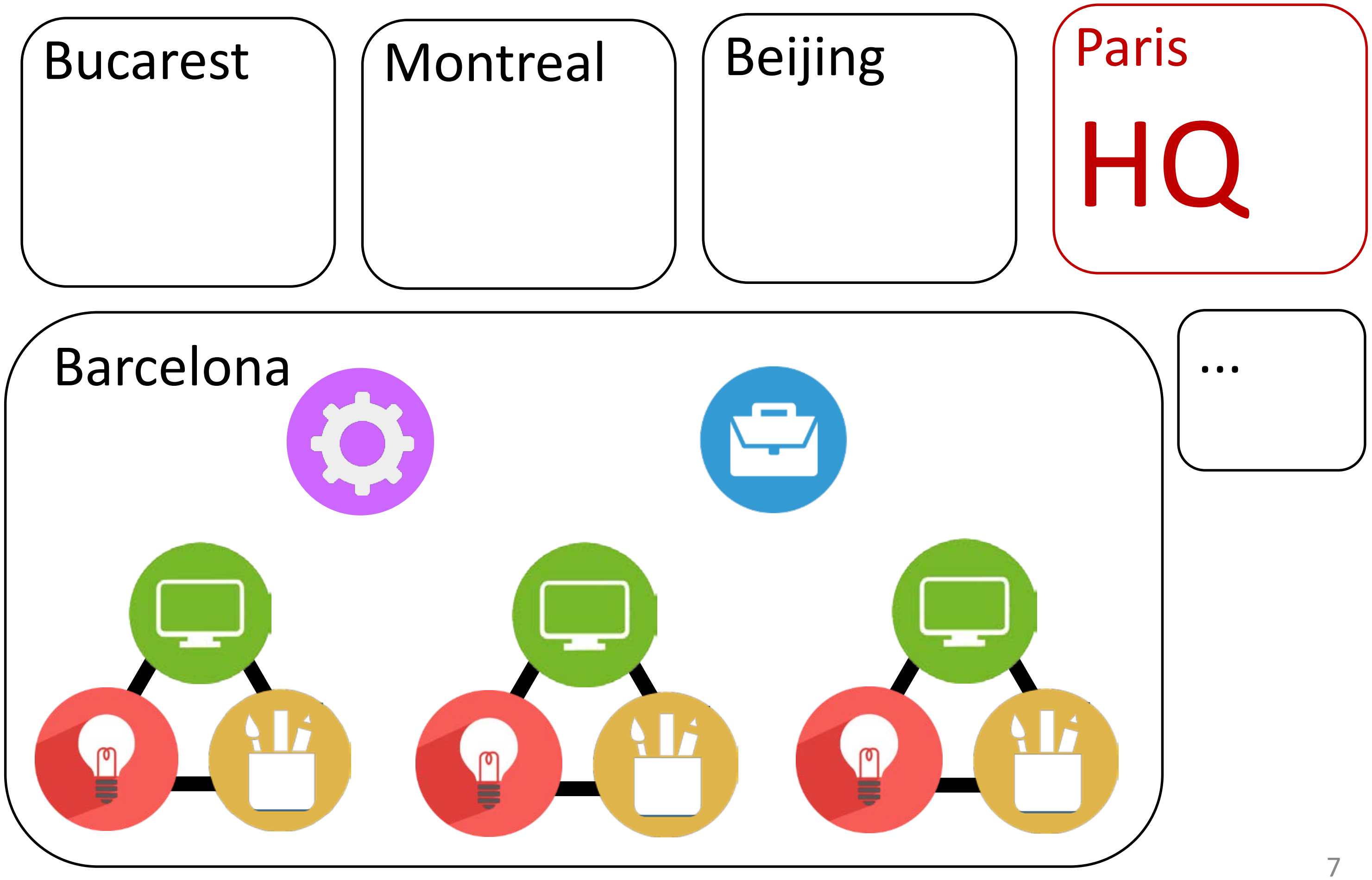
Studio Organization

- Each Creation Studio is composed of several teams of different natures



Company Organization

- Gameloft is composed of several creation studios, one HQ office and some support studios

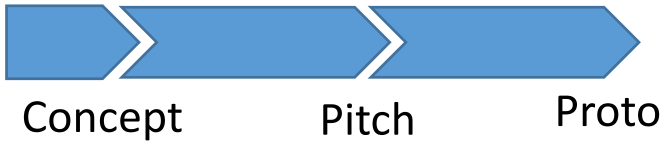


Development Phases and Gates

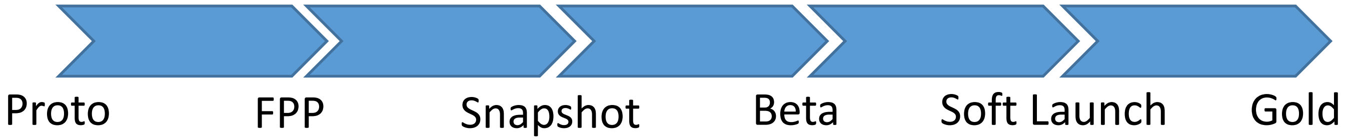
- Each development is divided in phases
- Each phase contains "Exams" called Gates
- If the project fails a Gate, it must repeat it or be cancelled.
- Phases are grouped into 3 stages:
  - Pre-Production
  - Production
  - Post-Production

Development Phases and Gates

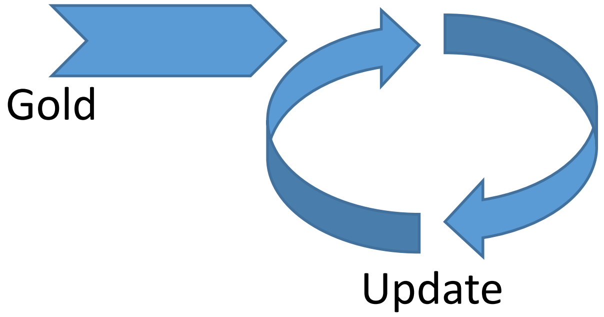
Pre-production



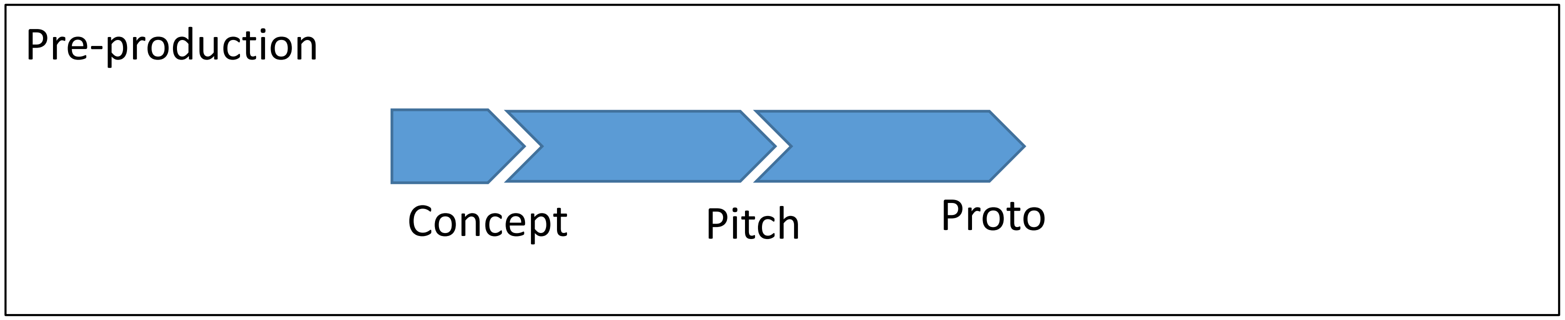
Production



Post-production

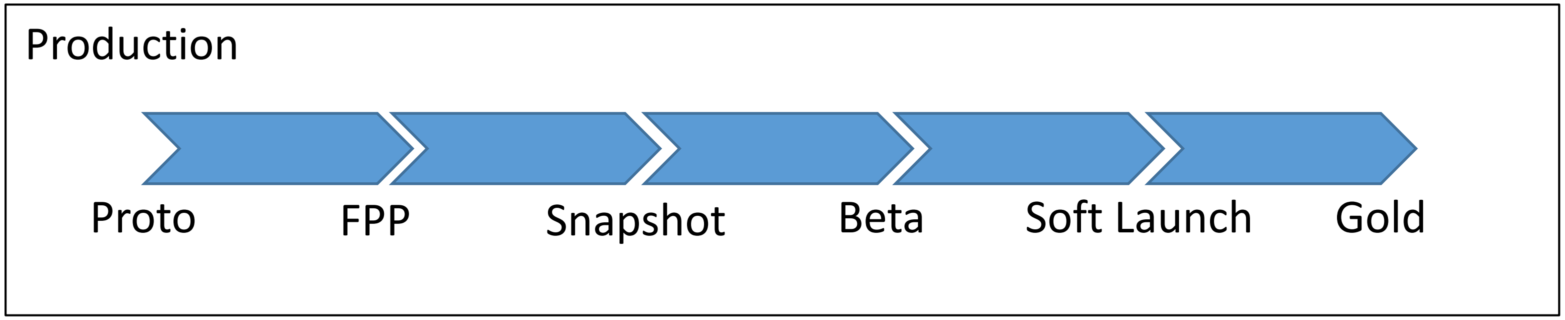


Preproduction



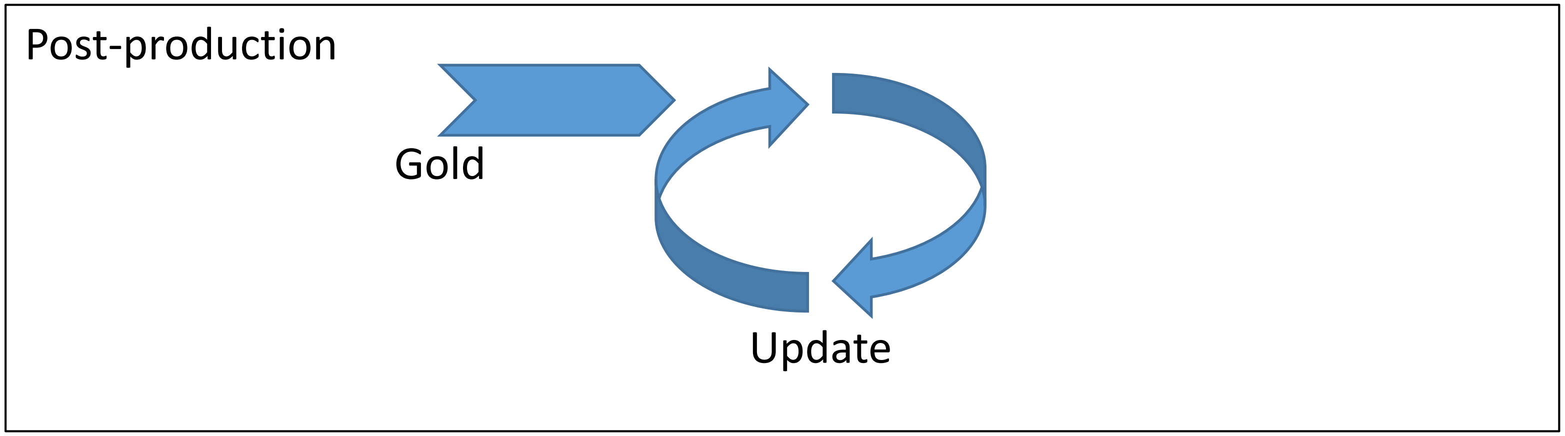
- Must showcase the potential of the game
- Outputs a prototype than can be thrown away
- Does not require the usage of final technology
- Non-critical bugs and glitches are acceptable

Production



- Must craft the final product
- Needs to be methodic
- Must use final technology and cover all cases
- Must be tested, reliable and bug-free

Post-Production



- Keeps the project alive after release
- Adds new features and content
- Keeps players engaged

Prototyping a Game

Properties / Objectives

- Create a playable Demo to showcase main features
- No long-term strategy
- As fast as Possible
- Very-fast iterations and feedback
- No Game Design Documentation
- No Technical Documentation

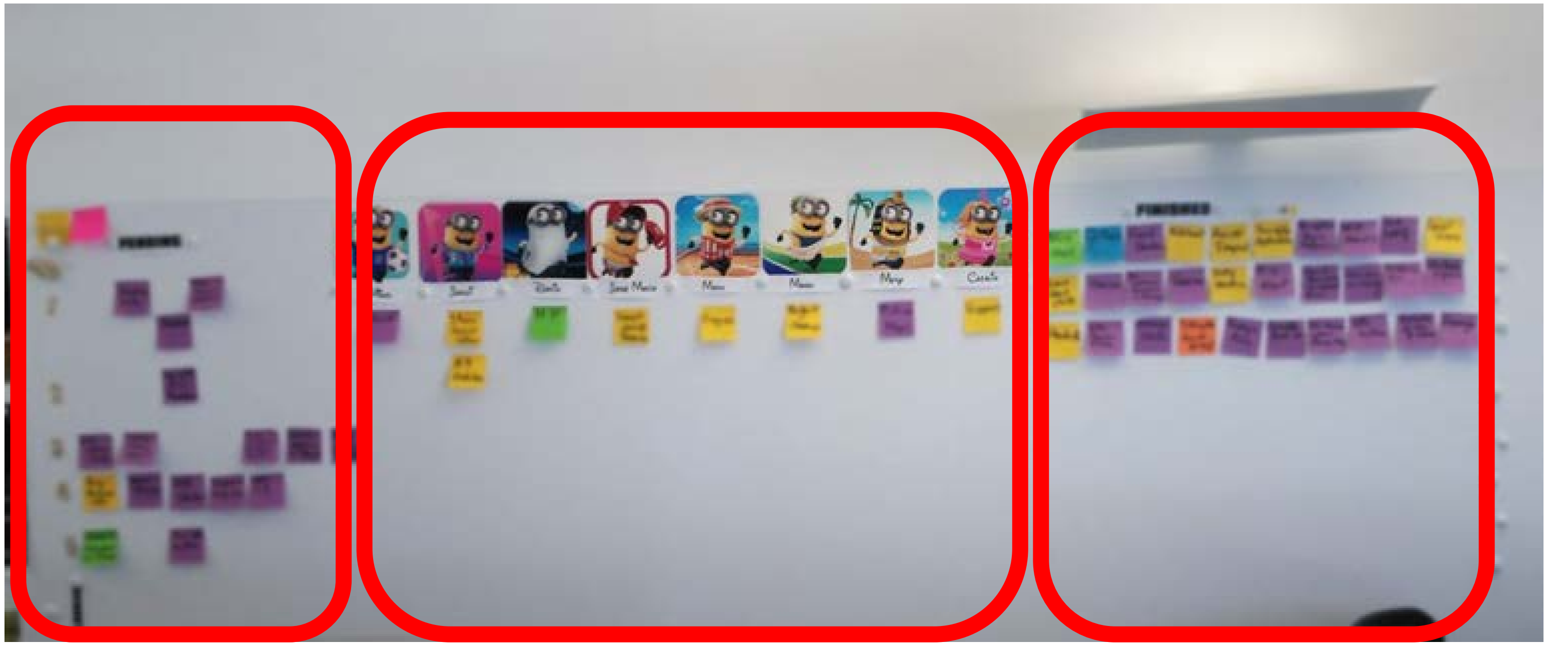
Methodology: “Agile-ish”

- All tasks start in a backlog.
- Once a programmer is free, he is assigned one of the backlog tasks (with the assistance of the lead engineer)
- That programmer develops the assigned task
- Once finished, the results are shown to the design team, who may alter the future features of the prototype.
- At any time, a task may be cancelled or modified

Team Skills

- Everyone can do anything!
- Team leads just try to focus development in the right direction.

Tools: The “Kanban” wall



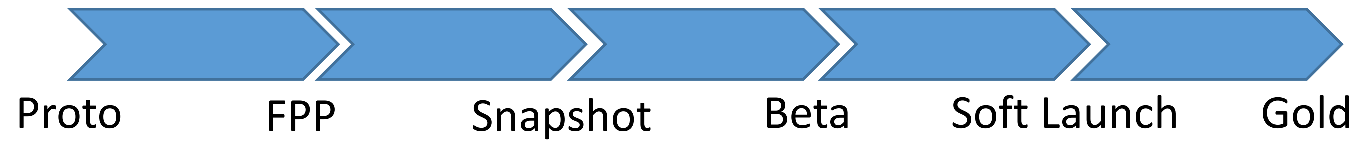
Backlog In-progress Finished

Advantages and Disadvantages

- Advantages
  - Iteration loops are very small.
  - Allows for constant stream of changes.
  - Ideas can be tested and discarded fast.
- Disadvantages
  - The resulting product is not publishable.
  - Incurs a big technical debt.
  - The longer the prototyping phase, the slower it is to advance.

Developing a Game

Properties / Objectives



- Develop a fully-functional and feature-complete game
- Do it under some time constraints
- Features are rigid and stable (almost)
- Develop a complete design Documentation
- Develop a complete technical Documentation

Methodology: Classic Iterative

- The project goals have already been stablished
- Implementation follows waterfall methodology for each of the gates
- After each gate, a new development iteration starts
- The gate/version system ensures the advancement on the project
- Documentation is written and validated

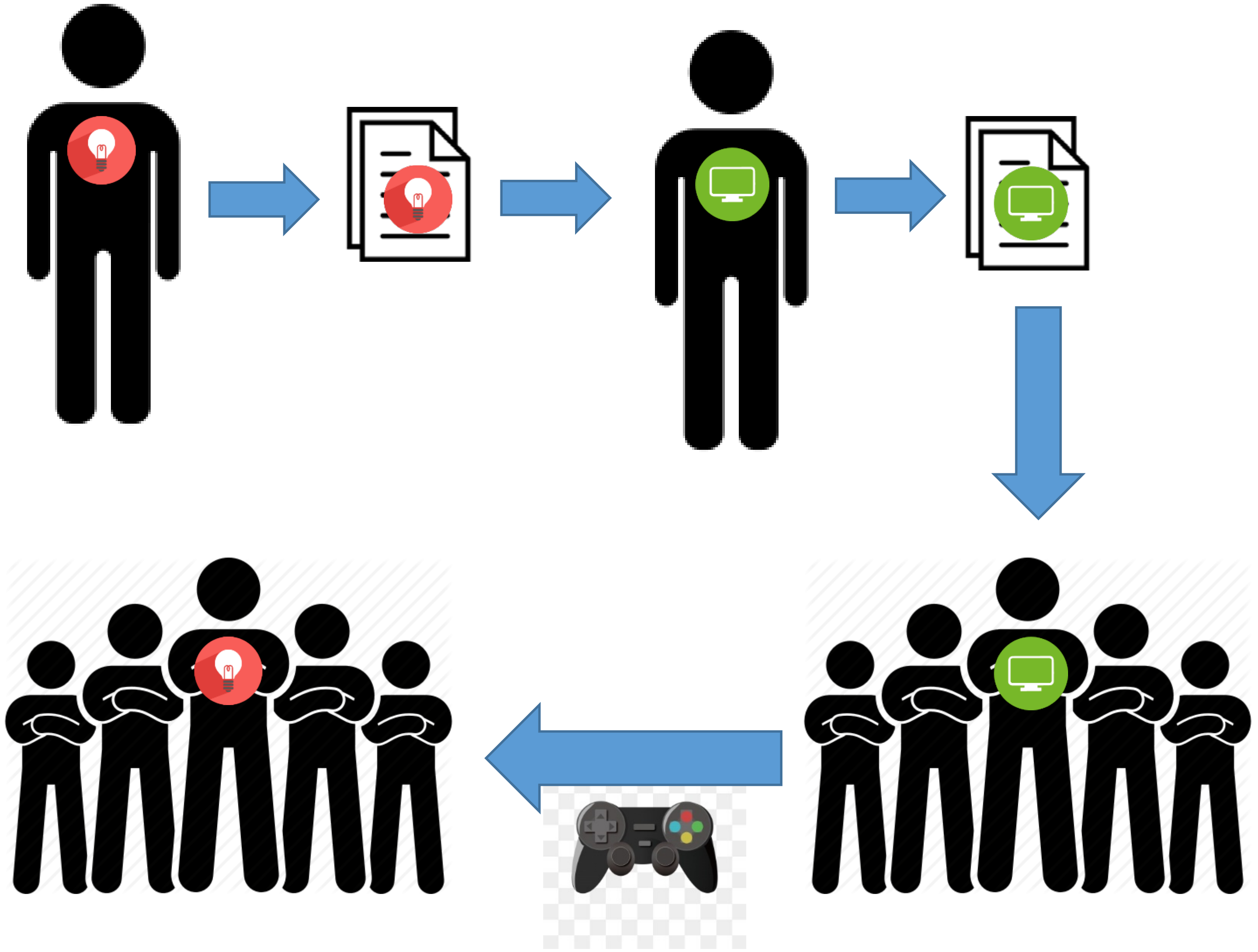
Lead job: Organizing the team

- Expertise is important
- Each team membre has some strong skills that must be put to use properly

The engineering teams

- Lead engineer [1]
  - Checks design documents
  - Designs architechture
  - Organizes the team
  - Implements features
  - Trains the Interns
- Principal engineer [0..3]
  - Designs architechture
  - Designs modules
  - Implements features
- Engineers / Programmers [0..*]
  - Design modules
  - Implement features
- Intern engineers [0..*]
  - Implements features

Lead job: Design Doc Checking



Lead job: Organizing the team

- Main questions:
  - Where are we?
  - Where do we go?
  - What is the team doing now?
  - What will the team be doing next?

“The lead lives in the future!”

Lead job: Designing Architechture

- 4-layer architechture
  - Input/Visuals
  - Gameplay
  - Economy/Logic
  - Data

- Designed by the lead/principal engineer.
- Customized UML

Modules

- Key pieces designed by the principal engineer and one(or more) engineers.
- Detailed design and implementation is done by engineers and programmers.

Code

- Code is usually previewed or reviewed by the principal engineer.
- Then it is either approved or rejected.
- Any engineer / programmer is allowed to suggest/implement modifications as he/she sees fit
- Restrictive ownership is discouraged in the team.
- Code refactor is performed by the principal

Debugging

- The project has continuous Integration mechanisms (Jenkins)
- Unit tests
- Quality Assurance (Testers) team during the last phases of the project.
- Every team member must fix bugs

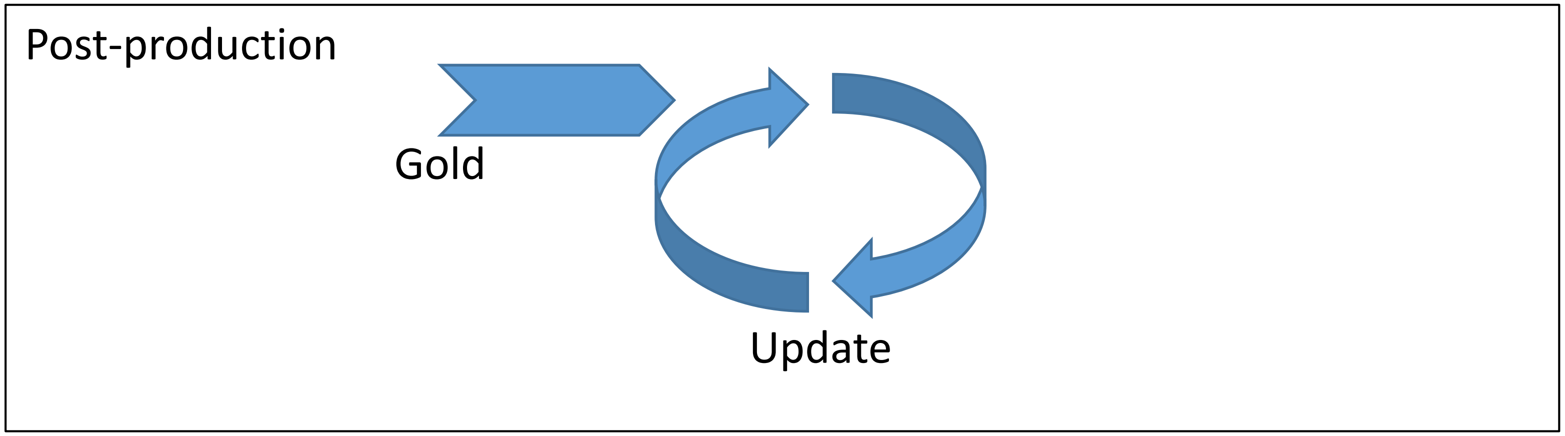
Advantages and Disadvantages

- Advantages
  - Solid codebase development.
  - Architecture is sound and stable.
  - Smaller technical debt.
  - Long term vision.

- Disadvantages
  - Progress is slower than during the prototype phase
  - Less adaptable to changes
  - Higher Bureaucracy
  - Hard dates

Updating a Game

The Update Process



- Every X weeks, a new version of the game must be released
- Each iteration is a full complete cycle
- Many iterations can be developed at once

Properties / Objectives

- The game has already been released and players are already playing it
- Changes in game design are checked and validated deeply before starting implementation
- Evolution of the product is data-driven

Methodology: Classic Iterative

- Each Update is considered as a new product
- For each Update a whole development cycle is executed
- There are no specific Gates during evolution.