Metoda priorytetyzacji MoSCoW

(M) – Must (S) – Should (C) – Could (W) - Won't

Project ships

Minimum Requirements

All these need to be met to pass. Please note, no automated checks for games means no passing. There are corresponding advanced requirements for some of the minimum ones. Fulfilling the advanced one counts as also fulfilling the basic one.

Main non-functional ones

1. (M) It must work (if it doesn't, it's disqualified): if I cannot play, it doesn't work.
2. (M) Project mantra followed.
3. (S) Code quality requirements are followed (see Code Quality section, below)
4. (M) External logging framework (not log4j version 1)
5. (M) Java FX simple GUI (or GUI through a browser).
6. (M) Network game, client-server architecture (web-app is also accepted)
7. (M) Both players are human players (playing against a computer is also accepted)
8. (S) Project should be driven! Estimates, milestones, versions, TDD, DDD, BDD, walking skeleton...

Game UI

Required screens are:

1. (M) Start / config screen - serves to launch the game and choose the settings
2. (M) Main / round screen - where the actual game happens

(C) Other screens are optional (for example ship placement as a separate screen).

Functional requirements.

General:

1. (M) AT LEAST one game
2. (M) 10x10 board (extra points if configurable)
3. (M) Fleet consists of: 4-mast ship, 2 3-mast ships, 3 2-mast ships and 4 1-mast ships (extra points if configurable).
4. (M) Wihanner s ships remaining while the loser has none. Draw is impossible (unless extra features come into play).

L10N:

1. We are bi-lingual: Polish and English are fine.
2. In future, we want to add more languages, make this easy!
3. Game messages (including UI elements or exception messages) are to be read from a file for a chosen language.
4. Choosing the language depends on a configuration variable AND there should be a GUI option to do that as well.

5. Changing the language mid-way through with immediate effect is worth bonus points (optional feature).

Ship placement:

1. (S) Manual
2. (M) Randomized
3. (M) Ships cannot touch - no adjacent field to a ship can have a ship (all directions count)
4. (W) Ships can be *broken*, multiple times (provided the length allows it)
   (Client approved our proposition during meeting on 25 June 2018 to program nuke feature instead of “Ships can be broken”)
5. (M) Diagonal placing is disallowed, only horizontal and vertical.
6. (C) Bonus points if I can re-randomize ships or manually adjust randomized fleet

Playing:

1. (M) Player has two boards: for his fleet for his shots/hits.
2. (M) You fire until you miss. It should be possible (if one is lucky enough) to sink entire fleet without giving the opponent the chance to return fire.
3. (M) Shot results: miss or hit. Both are marked on the hit-board.
4. (M) Upon a hit: mark the damaged place and ask for another shot.
5. (M) Upon a miss: mark on hit-board.
6. (M) Upon ship-sinking: once all masts of a ship are hit, ship sinks. Once the ship has sunk, mark all adjacent fields as "missed", since none of them can have a ship anyway.
7. (M) Sinking the last ship, that is, winning.
8. (M) One large additional feature (each team picks one).

Code quality and team setup

1. (M) Holy master - everything on master is holy, this is what is being checked by customer
2. (M) CI server - before anything gets pulled into master, it is integrated with master by CI server, it runs tests, checks, etc.
3. (M) Reviewers - pull-requests to master that are handled by CI server are then reviewed internally by teammates (in case of pair programming internal review is not needed)
4. (M) Outside reviewers - once team says yes, outsiders come in (each team chooses external reviewers). Two external reviewers need to say "code is OK" before it can be pulled to master.
5. (M) All code is on GitHub
6. (M) Jenkins (or equivalent) is used as CI server
7. (M) Maven is used by Jenkins and team
8. (S) Maven has Findbugs, JaCoCo and Checkstyle integration
9. (S) Dependency convergence must be 100%, verified with maven-enforcer plugin
10. (C) All dependencies that are NOT newest are recorded (along with reason why) in the dependencies
11. (S) Dependencies are newest or reason for why is in the docs, versions are kept up to date with Maven versions plugin.
12. (S)Sonar is used to keep quality gates (just internally is fine).
13. (C) Acceptance tests are welcome, one per feature is required
14. (C) Documentation should be provided, explaining program architecture (diagram is necessary, CRC diagrams are most welcome)
15. (S) Code quality – non-OO code is tolerated in little amounts.
16. (S) 60% unit test code coverage (lines).

Small additional features

Most are marked above (optional, bonus points).

1. (C) Configurable log destination: console, file, separate files.
2. (C) Transcript of the game happens (messages exchanged, shots fired, results, etc.) to a separate window.

Large additional features

Each team picks one: web-app, AI, salvo, torpedo, fleet sails, nuke, persistence.

1. (M) Salvo - ships shoot via salvo
2. (M) Torpedo - one-mast ships can fire torpedoes