Bored Games Framework

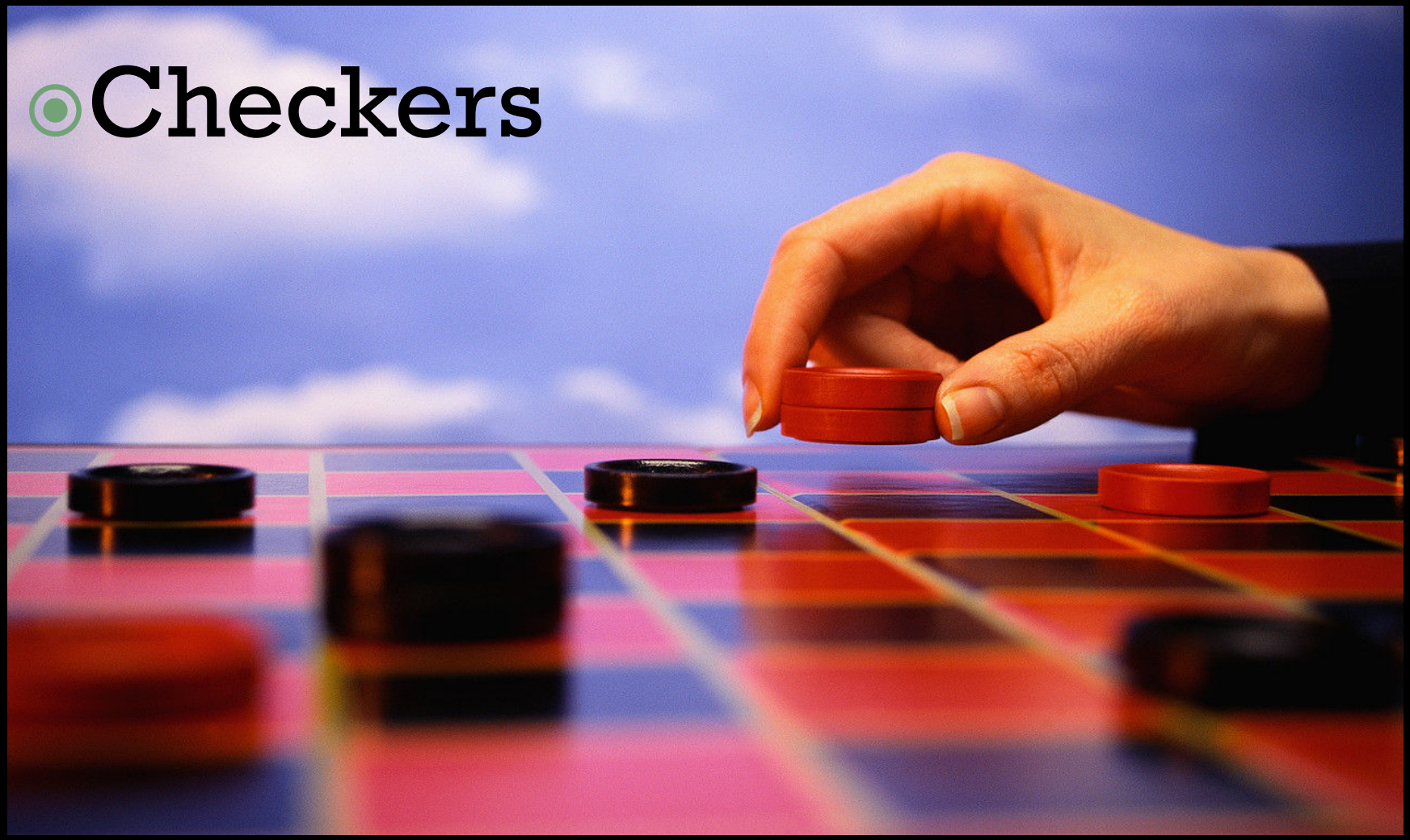
Analysis of Software Artifacts (17-654)
Carnegie Mellon University 2009

Who did What?

- Crux – GUI
- Nash – GUI
- Pidgin – Plug-ins
- Chronos – Framework

Implemented Game

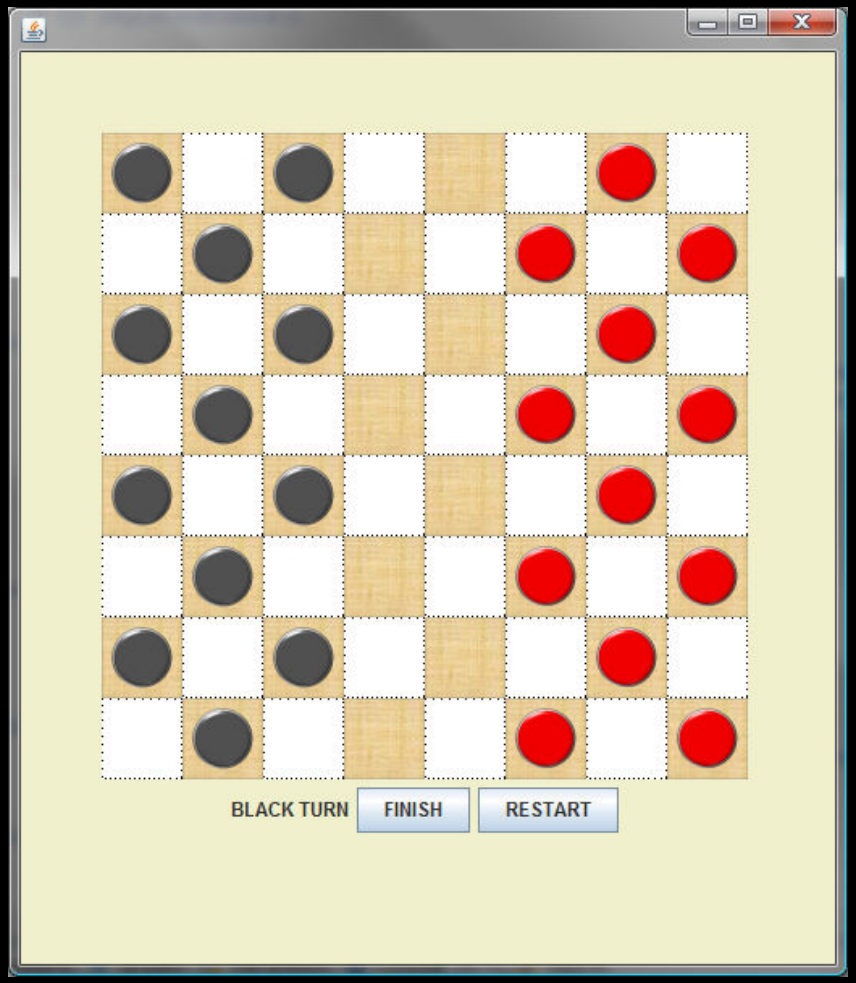
- Checkers



Crux Team (GUI)

- Yong Gyu Kim
- Dohoon Kim
- Bokuk Seo
- Sang-Hyun Lee

Demo



Specification Experience

- We can get information to implement from JavaDoc.
- But it is hard to know proper sequence.
- In sequence diagram there are only two case of usecase
- Direct use of game plug-in module.
- We also referred game plug-in spec.

Plural

- Used to check state and verify null check of input value
- UI has not many methods to verify through plural
- Hard to figure out access permission
- We can't check variables or conditions in detail
- Hard to check results from plural

Nash Team (GUI)

- HunJae Lee
- SeongYong Lim
- SeJoon Oh
- Duri Kim

Demo

- NASH UI plug-in
  + Pidgin Game plug-in
  + Chronos Framework

Specification Experience 1

- Occurred issues when using the framework interface
  - In Checkers, a player can have MULTIPLE MOVES in one turn
    - The framework is supposed to determine the player's turn in original design.
    - But, we implemented that UI plug-in determines which player has the next turn based on the current situation given by game plug-in and the framework.

Specification Experience 2

- Run Plural on our UI plug-in
  - We designed the Plural implementation with three different perspectives(dimensions)
    - Initialize the framework
    - Initialize our UI plug-in
    - Check the states according to the game workflow
  - When a function is related to more than two dimensions and we want to check the same access permissions at the same time, Plural does not allow us to use like that.
  - Example

```
@Full(value="GAMEWINDOWSTATE", requires = "READY", ensures = "CONSTRUCTED")
@Full(value="GAMEPLUGINSTATE", requires = "READY", ensures = "CONSTRUCTED")
public void startup() {
```

Testing Experience 2

- Integration challenges
  - Owing to framework specification, there was no big chaos.
  - But there were some adjustments for the sequence that UI plug-in calls the framework methods.

- System/Component test
  - Without game plug-in, we could not perform our tests properly.
  - The framework needed more separation b/w game and UI plug-ins.

Pidgin Team (GUI)

- Carlos Simões
- Higino Silva
- João Carlos Almeida
- Miguel Graça Oliveira (Presenting)

Experience

- Initial implementation very easy
- Some limitations on spec:
  - Not a complete decouple between Game and UI
  - Alternating moves constraint (later overcome)
- Testing
  - Framework forces the initialization of the board
  - Not easy to simulate complex game situations
  - Difficult to assess the responsibility of the defects found

Plural

- @Perm(requires = "#0 != null")
- **public CheckersBoard(CheckersGame theGame) { …**

- Board b = **new CheckersBoard(this);**
- Shows a warning: "Must not be null" ☹

- Plural is good for simple checks
- Too much effort for complex situations
- Too little documentation

Chronos Team (Framework)

- Ignacio Cavero
- João Brito
- João Viegas
- Miguel Sousa (presenting)

Framework Design Experience

- Some issues not detected during design
- Some changes done on framework had some impact on the design, but without affecting the GUI development.

Testing Experience

- Four teams:
  - Tests can only start after development is complete, with four teams this becomes more difficult
  - Coordination is not easy
  - Two GUI teams
- Framework is called by the GUI, this makes testing more complex

Plural

- It is a tool still in development
- Lack of documentation
- Problems found:
  - Inheritance support
  - Methods allowed for multiple states
  - State changes based on previous multiple states
  - Local analysis and partial system annotation
- Long learning curve

Questions & Answers