Exercise 2: I require a design

Hand-out: 12 April 2004
Due: 19 April 2004

Please solve this exercise within your project team.

1. Requirements Document for the Game

This assignment is associated with your game idea (your project).

You will work as a project team and will have one week to finish the documenting, but start as soon as possible. One tends to underestimate the workload even if he/she is told not to undervalue the importance of the requirements documenting.

For this assignment, you need to:
1. Reach a complete and consistent understanding of the game that your team will work with later on;
2. Analyze the requirements for the game project to establish exactly what an end-user of the system wants;
3. Validate and document the requirements specification for your project following the IEEE standard;
4. Submit your file of the requirements documentation on time to your assistant.

Some suggestions:
1. The requirements should specify ‘what’ but not ‘how’;
2. When there is updating, make your modifications clear in the updated version.
3. Based on IEEE standard, make sure that nothing important is missing from the requirements specification.

2. Modularity

Class FIGURE below represents a figure on a chess board. The figure has a type (pawn, king, queen, ...) and a current position on the chess board. The modifier ‘move_to’ moves the figure to a new position on the chess board. Of course, depending on the current position and the figure-type, only a few target positions are legal at any point in time. Look at the code and find the mistakes in its design. Explain what is wrong and provide a better solution. The solution (just as the code below) does not need to be complete, but the concept of the solution should be clearly visible.
indexing
description: "Objects representing figures on a chess board."

class
FIGURE
create
make
feature {NONE} -- Initialization
make_pawn is
  -- Create new pawn figure.
  do
    type_id := pawn_type_id
  ensure
    is_pawn: type_id = pawn_type_id
  end

feature -- Access
x: INTEGER
  -- Position of figure on X axis of chess board
y: INTEGER
  -- Position of figure on Y axis of chess board
type_id: INTEGER
  -- Type number of current figure

feature -- Constants
pawn_type_id: INTEGER is 1
king_type_id: INTEGER is 2
...

feature -- Status
is_valid_type_id (a_type_id: INTEGER): BOOLEAN is
  -- Is `a_type_id' a valid figure type number?
...

is_valid_position (an_x: INTEGER; a_y: INTEGER): BOOLEAN is
  -- Is (`an_x', `a_y') a valid position on an chess board
  -- whose range is from (`1', `1') to (`8', `8')?
...

is_valid_destination (an_x: INTEGER; a_y: INTEGER): BOOLEAN is
  -- Is (`x', `y') to (`an_x', `a_y') a valid
  -- move for the current figure?
require
  is_valid_position: is_valid_position (an_x, a_y)
do
  if
    type_id = pawn_type_id
    then
      Result := ...
  elseif
    type_id = king_type_id
    then
      Result := ...
  end

feature -- Movement
move_to (an_x: INTEGER; a_y: INTEGER) is
  -- Move current figure from (`x', `y') to (`an_x',
  `a_y').
require
  is_valid_position: is_valid_position (an_x, a_y)
  is_valid_destination: is_valid_destination (an_x, a_y)
...

invariant
  valid_current_position: is_valid_position (x, y)
  valid_type_id: is_valid_type_id (type_id)
end