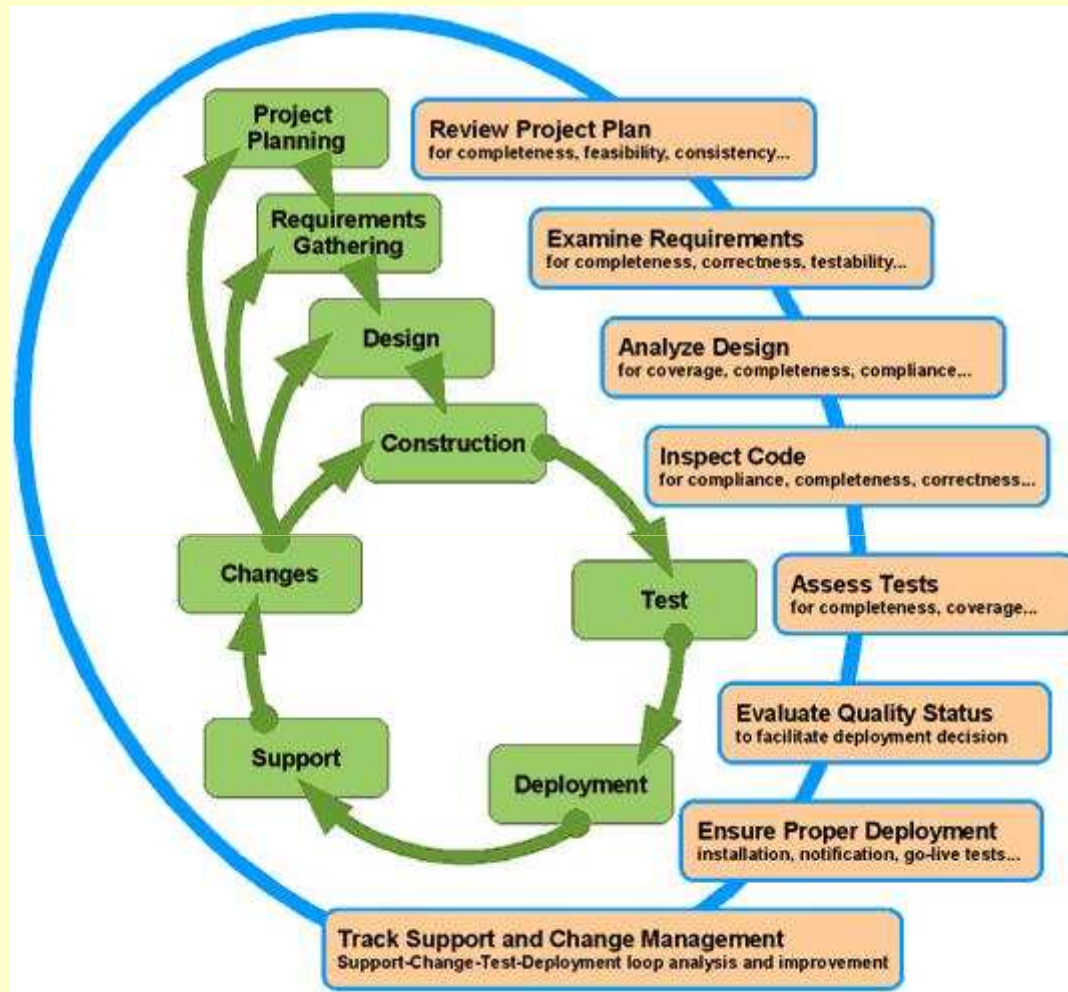


Software Engineering :
A PBL Session

(L2-RobotProblem)

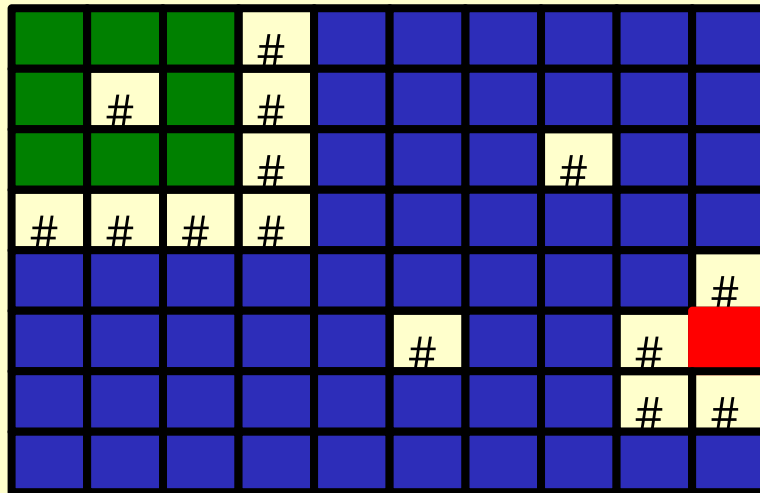
Software Life Cycle and Software Process



This problem is concerned with looking at the life cycle when requirements creep during the development process.

What sort of procedures would you put in place if you knew that this would happen in advance?

The Robot Problem: An overview of problem domain



In a 2-dimensional grid/plane ($n*m$) there are either walls or spaces.

We represent the walls as ‘#’ in the diagram (with spaces coloured into different partitions)

In such a grid we can place robots who can move horizontally and vertically but cannot move on top of a wall.

You need to calculate the minimum number of robots that are needed in order to be able to visit all spaces in the grid.

In the example above, there are 3 partitions and so we need 3 robots.

The Robot Problem: Requirement1

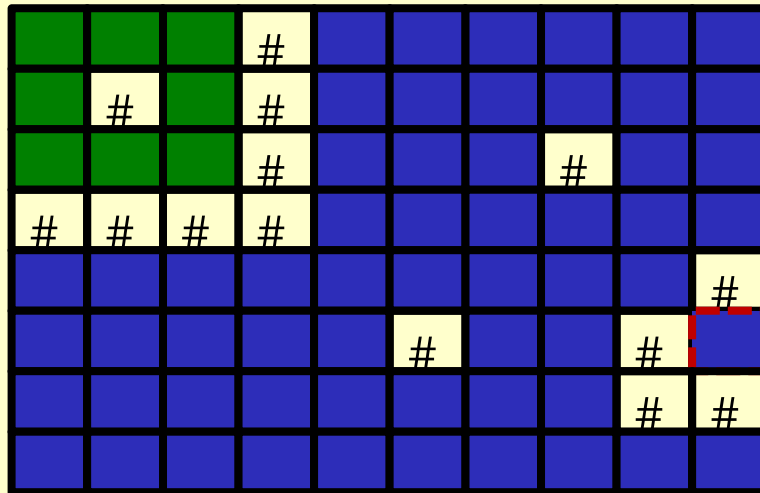
Given as input a 2-dimensional array of booleans (with true representing the fact that a wall is at the given co-ordinates), you are to output an integer which represents the number of partitions in the graph.

You are also to document all steps in the life cycle (including design and tests)

You are to work in teams of 3 or 4 developers. You have 3 hours. All teams must agree on a common programming language to be used.

The Robot Problem: Requirement2

A second type of robot can also move diagonally

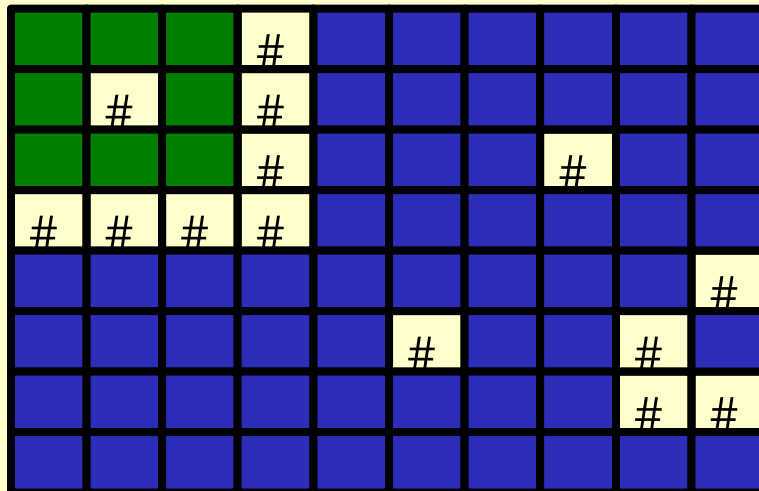


In this case, the number of partitions is now 2 as the previously **separated** square is now connected to the partition in **blue**

You are to extend the system to calculate the partition values for both types of robot

The Robot Problem: Requirement3

Robots are limited by the amount of distance they can travel. As a useful bound on this value, you are to calculate the size of the largest partition



In this example (with the 2nd type of robot) the largest partition is coloured blue and its size is 58