CSC7426 : Basics of Software and Data Engineering

J Paul Gibson, D311

paul.gibson@telecom-sudparis.eu

http://jpaulgibson.synology.me/Teaching/TSP/CSC7426/

Requirements Creep and designing for the future

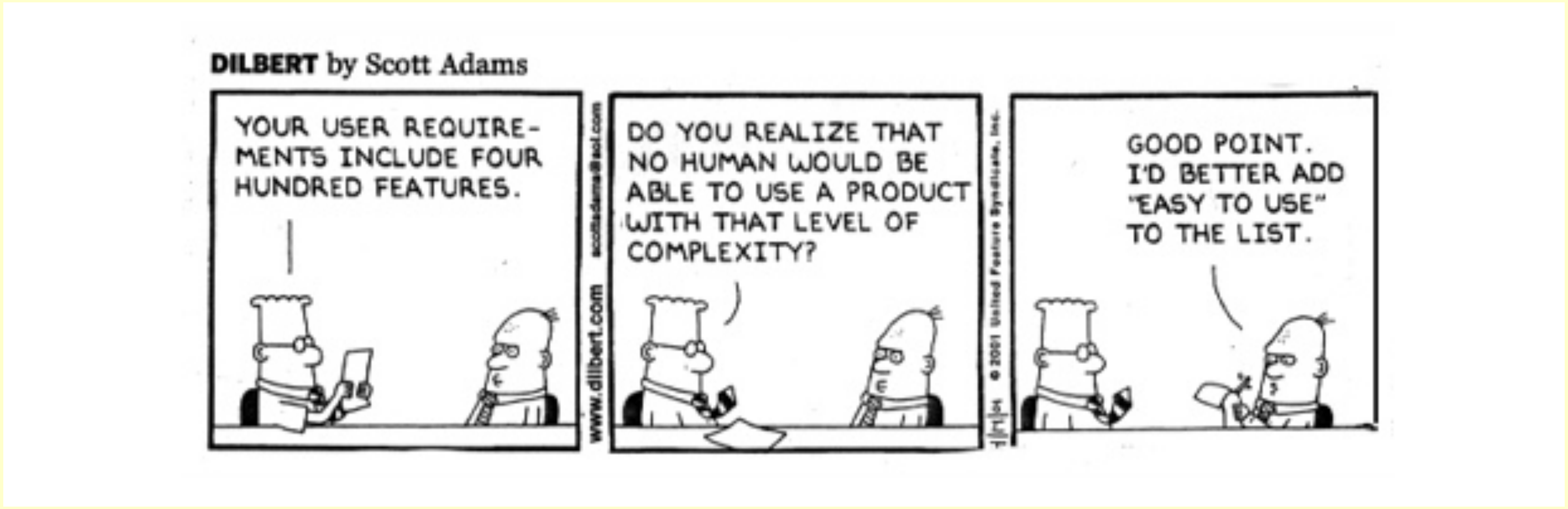
http://jpaulgibson.synology.me/Teaching/TSP/CSC7426/L2-RequirementsCreep.pdf

Requirements Creep

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?

How would this impact the design of your system?


DILBERT by Scott Adams
YOUR USER REQUIREMENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.
www.dilbert.com
scottadams@aol.com
© 2001 United Feature Syndicate, Inc.

Designs that are maintainable, evolvable and re-usable are more likely to be safe against requirements creep

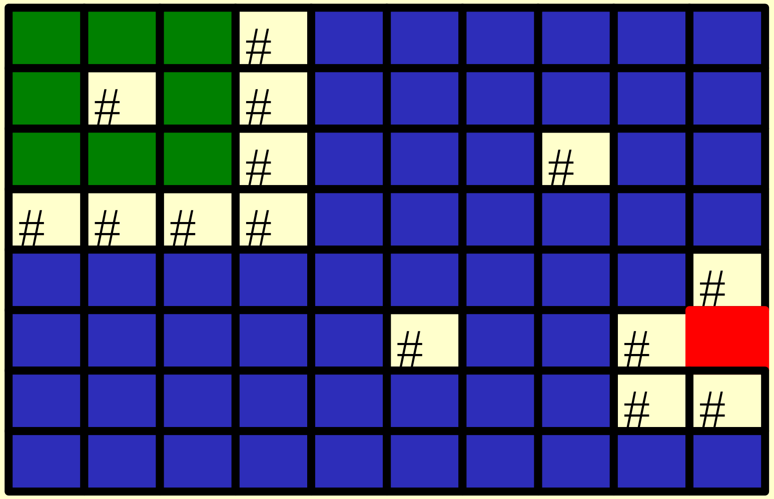
Requirements Creep

Background Reading:

Jones, Capers. "Strategies for managing requirements creep." Computer 29.6 (1996): 92-94.

“Frequent changes in requirements are not always caused by capricious clients (although sometimes they are). The root cause of requirements volatility is that many applications are attempting to automate domains that are only partly understood. As software design and development proceeds, the process of automation begins to expose these ill- defined situations. Therefore, although creeping requirements are troublesome, they are often a technical necessity."

The Robot Problem: A first requirements specification



In a 2-dimensional grid/plane (`N x M`) there are either walls or spaces.

We represent the walls as '# ' in the diagram (with spaces coloured into different partitions). A key part of your design is the data structure you use to implement this.

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

Functional Requirement 1 - You need to calculate the minimum number of robots that are needed in order to be able to visit all spaces in any given grid (of walls and spaces), ie the number of partitions.

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

Design for the future

You are to write a program that solves the robot problem, together with the test code that demonstrates that it works correctly

Your program design should be ‘ready’ for the addition of new requirements/features or changes to existing requirements

Try to predict what the ‘client’ could *reasonably* ask you to add or change, and evaluate how your design copes with such evolution.

Once you are happy with your solution to this problem, I will give you some additional requirements additions/changes (some new functionality)

DISCUSSION - what would be *reasonable* new requirements?

Design and Code for Maintenance (by others)

You are to implement the new functionality:

1) On your own code
2) On the code of one of your class-mates

Your goal is to include both functionalities in the new versions of the 2 systems.

How easy was it to add the new functionality? Compare and contrast your original code with the code of your classmate.

Problem for next session: I will give you a 3rd functionality to implement - you can choose to do this using your original code or the code of your classmate (whichever is easiest)