Welcome to the class

Why not only code it ?

Famous Software Failures

- **AT&T long distance service fails for nine hours**
  (Wrong BREAK statement in C-Code, 1990)

Why not only code it ?

Famous Software Failures cont'd:

- **Mars Climate Orbiter (September 23rd, 1999)**

  The 125 million dollar Mars Climate Orbiter is assumed lost by officials at NASA. The failure responsible for loss of the orbiter is attributed to a failure of NASA's system engineer process. The process did not specify the system of measurement to be used on the project. As a result, one of the development teams used Imperial measurement while the other used the metric system of measurement. When parameters from one module were passed to another during orbit navigation correct, no conversion was performed, resulting in the loss of the craft.

Why not only code it ?

Famous Software Failures cont'd:

- **Hi-tech toilet swallows woman (2001)**

  [Source: Article by Lester Haines, 17 Apr 2001] A 51-year-old woman was subjected to a harrowing two-hour ordeal when she was imprisoned in a hi-tech public convenience. She was captured by the toilet, which boasts state-of-the-art electronic auto-flush and door sensors, which steadfastly refused to release it's victim, and further resisted attempts by passers-by to force the door. Finally the fire brigade ripped the roof off the cantankerous crapper.

Why not only code it ?

Famous Software Failures cont'd:

- **E-mail buffer overflow (1998)**

  Several E-mail systems suffer from a "buffer overflow error", when extremely long e-mail addresses are received. The internal buffers receiving the addresses do not check for length and allow their buffers to overflow causing the applications to crash. Hostile hackers use this fault to trick the computer into running a malicious program in its place.

Why not only code it ?

Software is

- a critically important infrastructure component
- a key enabler
  - militaryly
  - economically
  - scientifically
  - culturally

But usually

- expensive
- of poor quality

Common Software Problems

- **Software can cost hundreds or thousands of dollars per line**
- **Lifetime maintenance costs are higher still**
- **Software is late or fails**
- **Software is not performant (too slow)**
- **Software is incomprehensible**
- **Software is more trouble to use than it is worth**

Past Approaches to Solutions

- **Use more people**
- **Create better programming languages**
- **Write software tools to help create software**
- **Design before writing**
- **Start by baselining requirements**
- **Train people better**

- **Create more chaos**
- **Bad programs can be written in any language**
- **(who finds the error in reasoning in here ?)**
- **Are you designing the right program ?**
- **but they change !**
- **…to do what ?**

The Software Crisis

Summary:

Millions are spent for an incomprehensible tool that comes late just to cause trouble, and we don't have answers

or:

THE SOFTWARE CRISIS (1968)

History

1968: NATO Software Engineering Conference in Garmisch (Germany):

Why cannot bridge-building techniques be used to build operating systems *(‘engineering’) ?*

The Nature of Software

- It is a component of a larger system that “fits” with hardware, people, mechanical devices
- It transforms data using computers
- It has a complex structure
- It is usually very large, expensive, and lengthy to build

The Nature of Software

- Software is extremely malleable – we can modify the product all too easily
- Software construction is human-intensive, there are no real costs of materials
- Software is intangible: no laws of physics are applicable
- Software is not detectable by any of the five human senses

The Nature of Software

But:

The are characteristics analogue to physical engineering processes

Studying such analogs can be useful:

- **Help us learn about computer software**
- **Find points of similarity**
- **Suggest successful approaches to be emulated**
- **Avoid known mistakes**

Engineering Example

Building a house:

- **Land and finances**
- **g… si…**
- **Architect will define number of floors and rooms, o…**
- **t…**
- **C…**
- **Entering**
- **L…**
- **Fixing minor problems, leaking in the roof …**

System Feasibility

Software Plans and Requirements

Product Design

Detailed Design

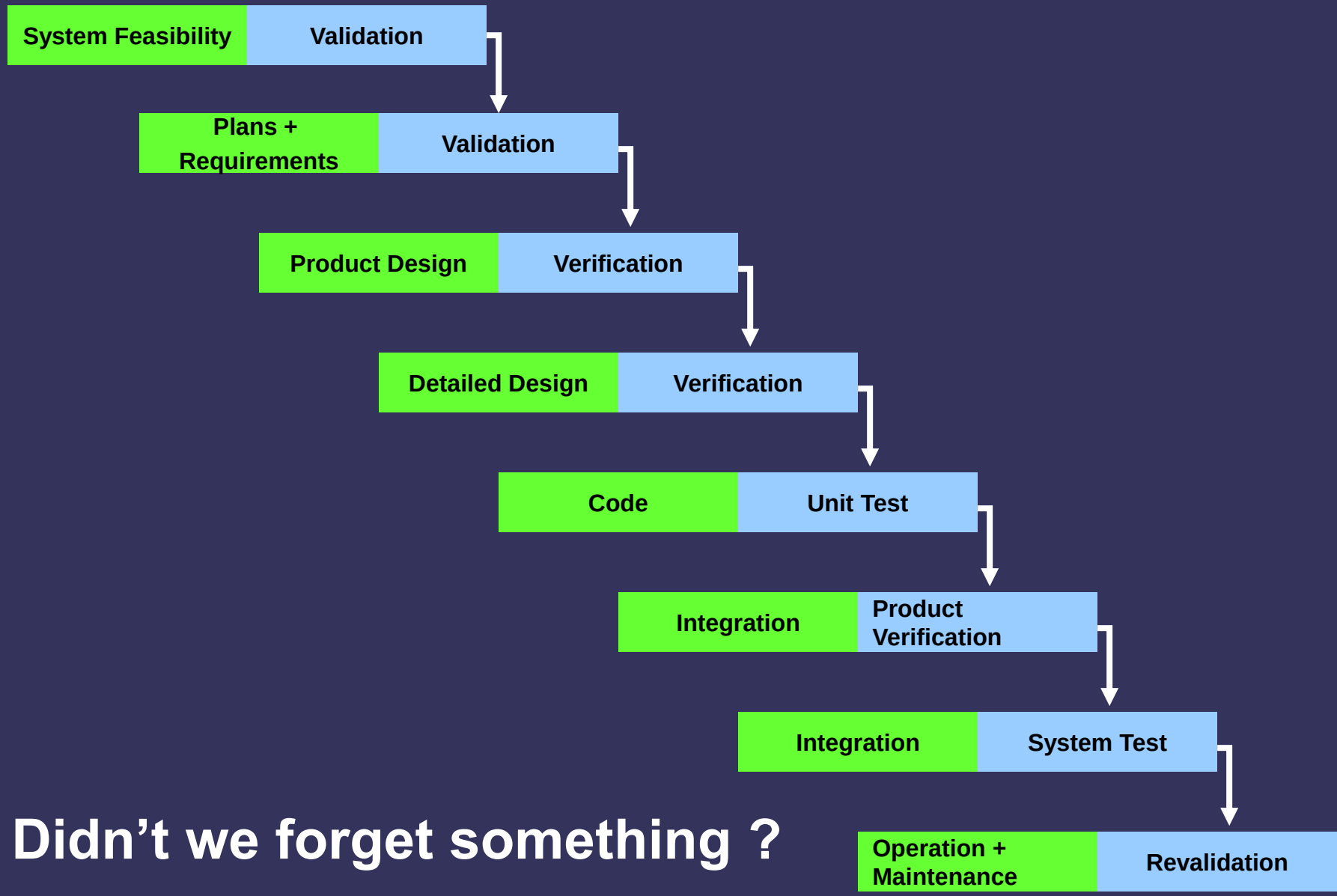
Code

Integration (Product Verification)

Integration (System Test)

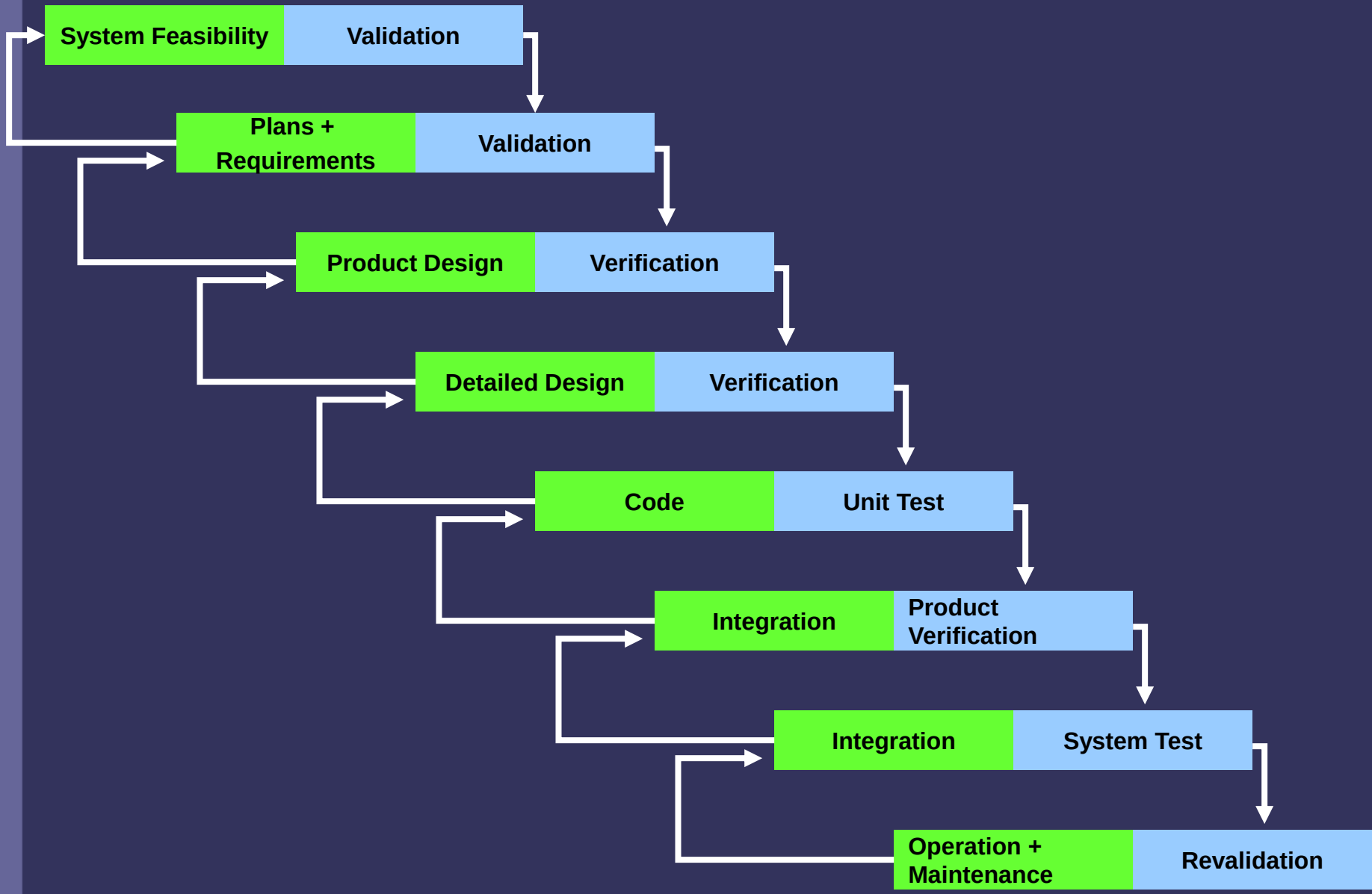
Operations and Maintenance

The Waterfall Model

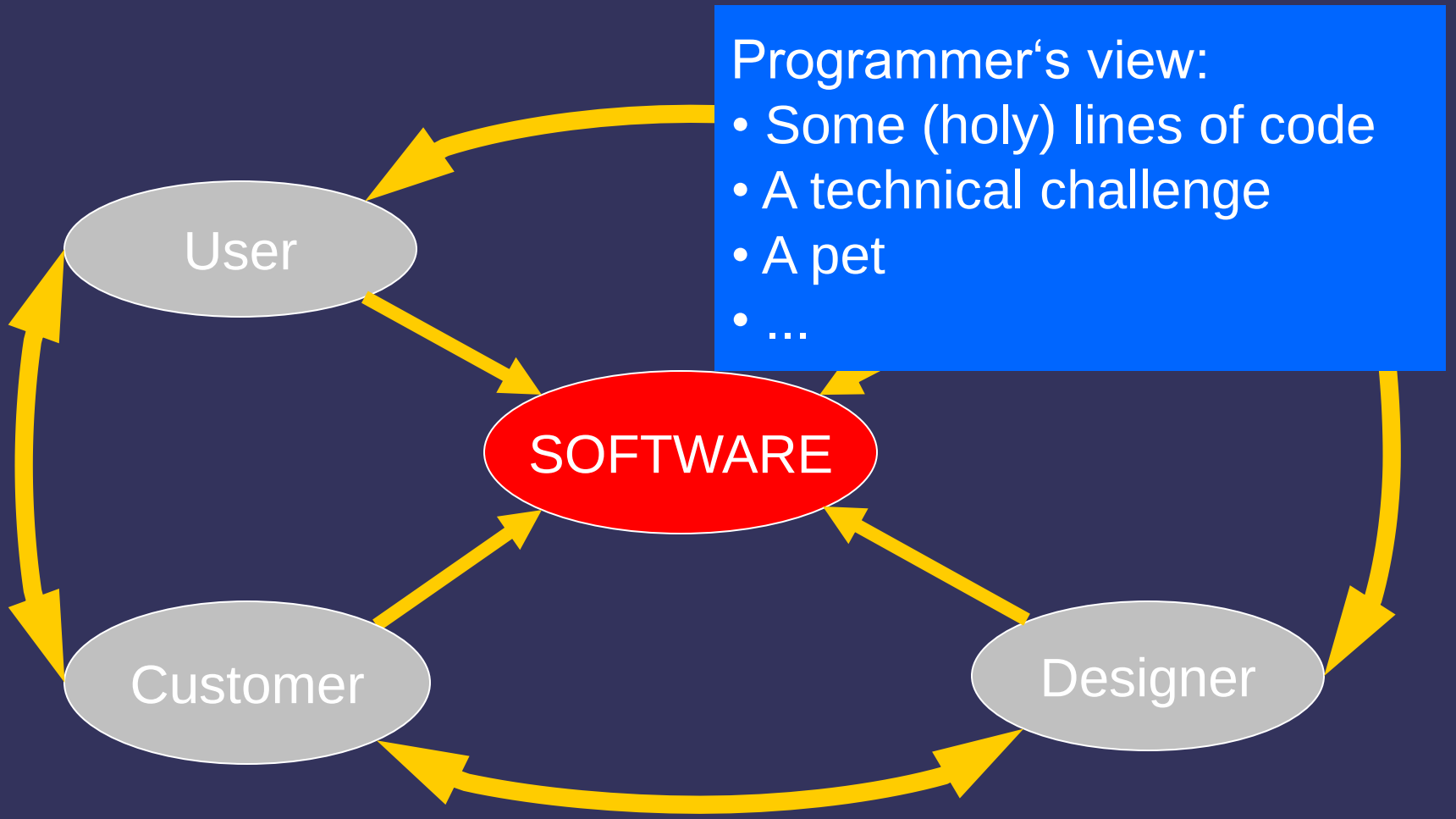


Didn’t we forget something ?

The Waterfall Model

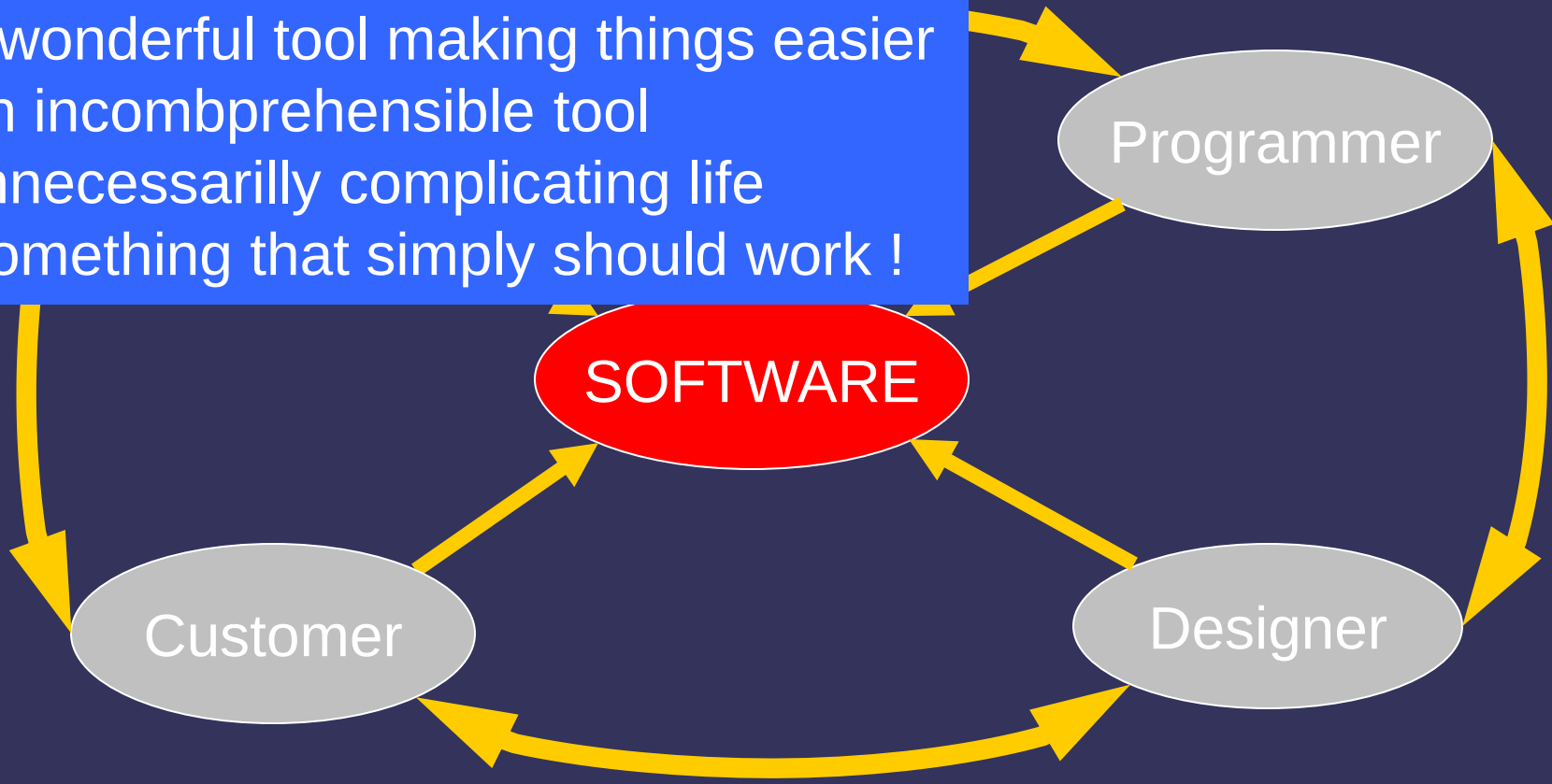


The Human Factor

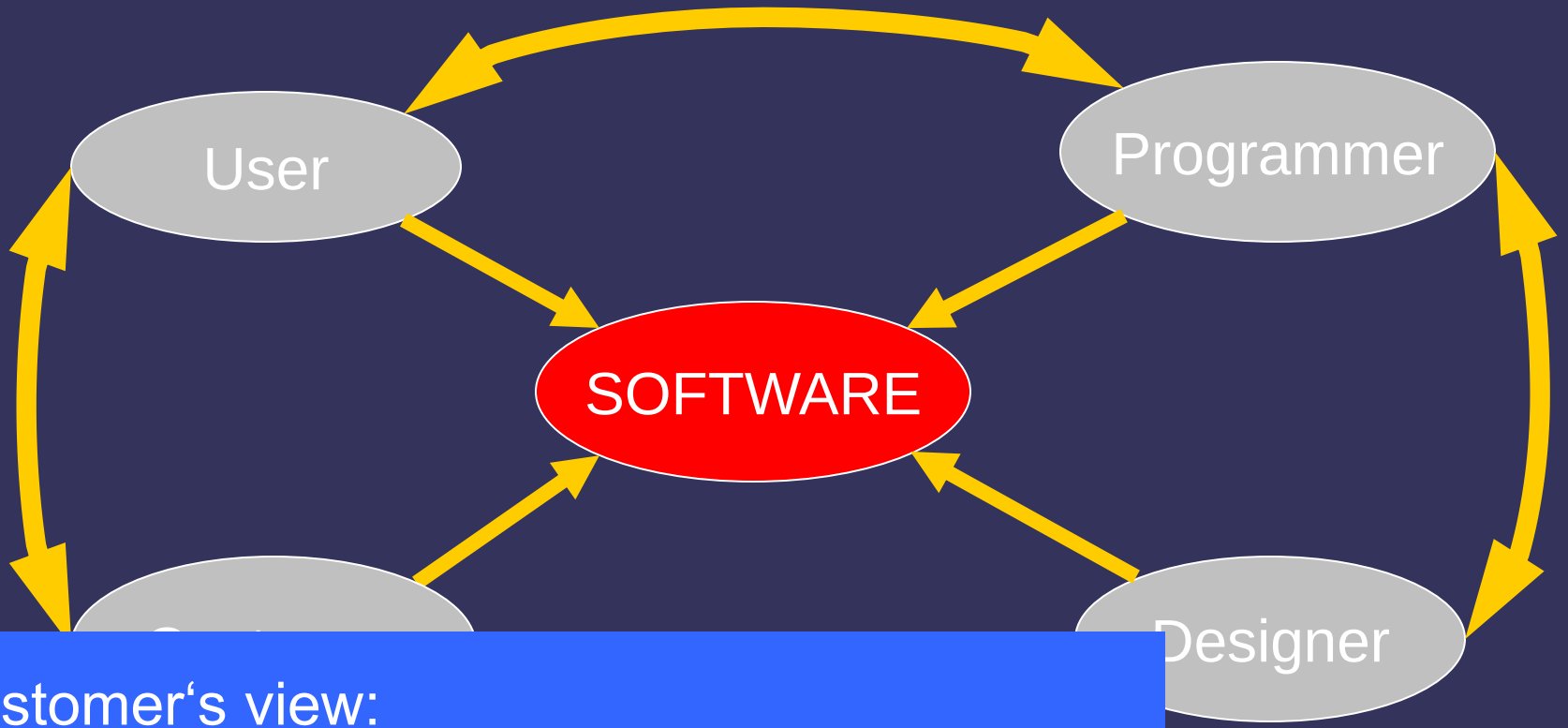


The Human Factor

User‘s view:
- A miracle
- A wonderful tool making things easier
- An incombprehensible tool unnecessarilly complicating life
- Something that simply should work !



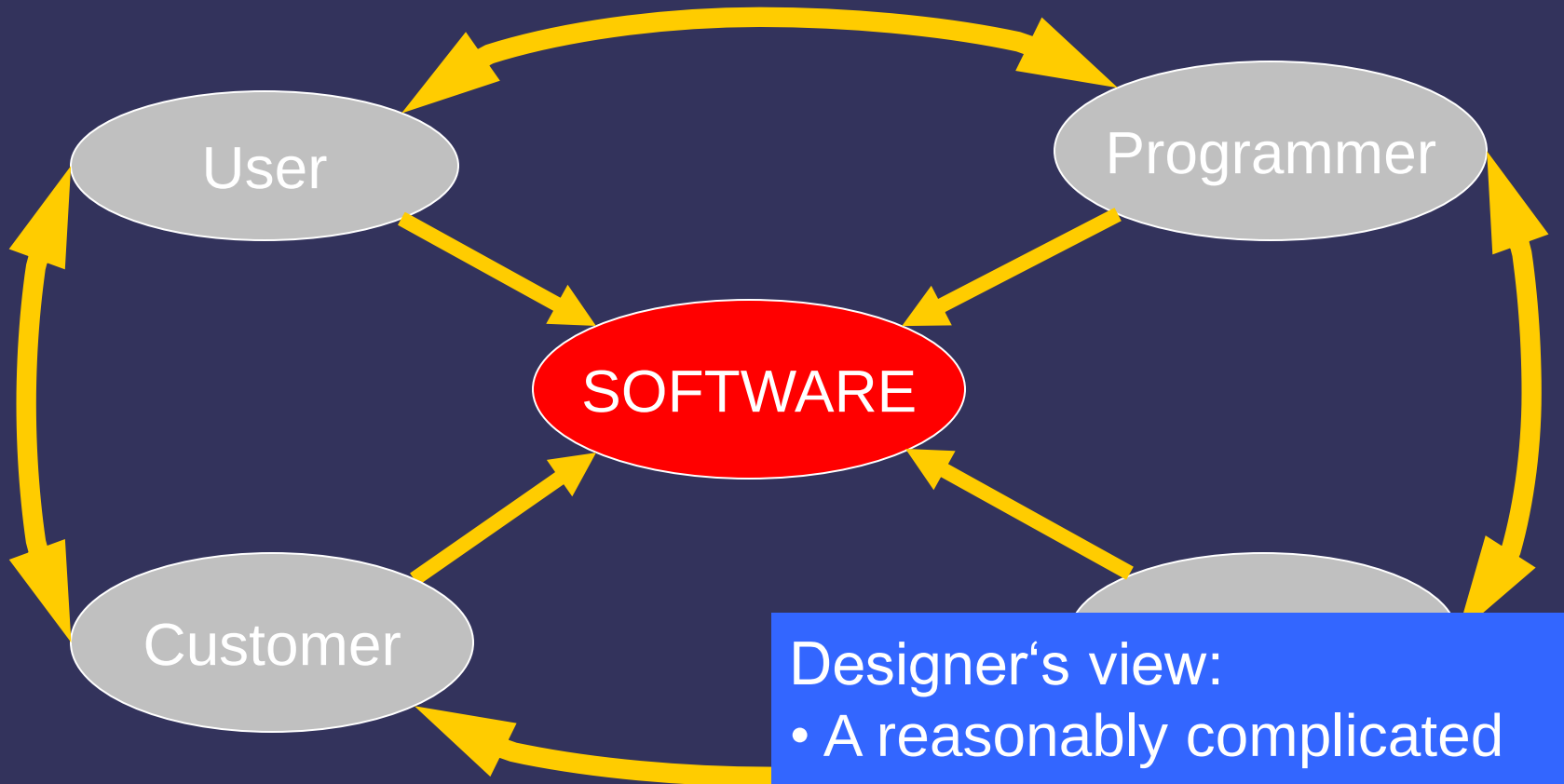
The Human Factor



Customer‘s view:

- A hopefully affordable tool to enhance profit.

The Human Factor



Designer‘s view:

- A reasonably complicated tool to fulfill the needs
- A technical challenge

Review of Waterfall Model

Weaknesses:

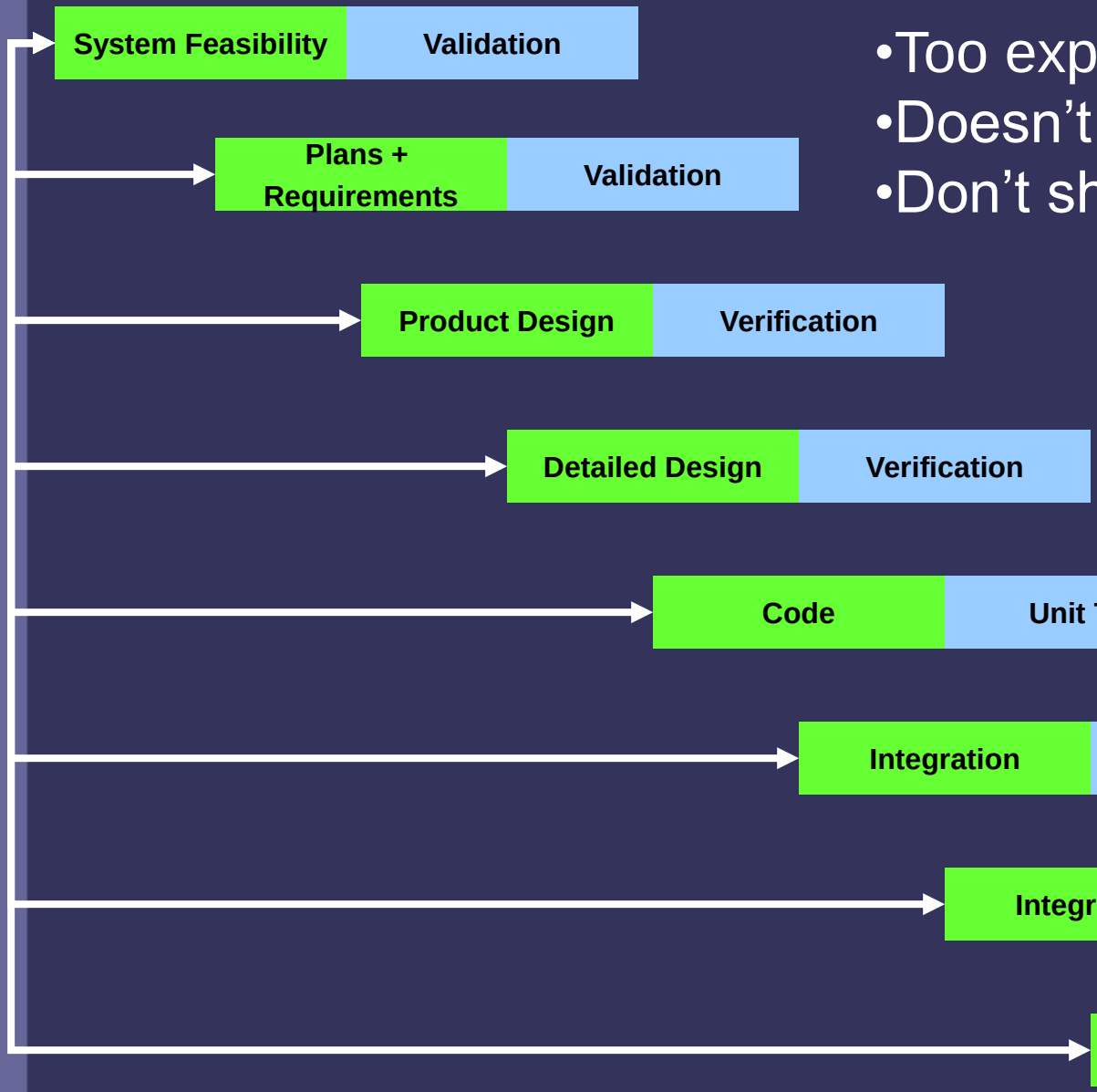
- **Usually requirements change, are incomplete or even not known**
- **Communication ! (…see Mars Orbiter…)**

Result: ‘That’s not what I meant !’ (go back to last step)

WF-Model reacts very statically:

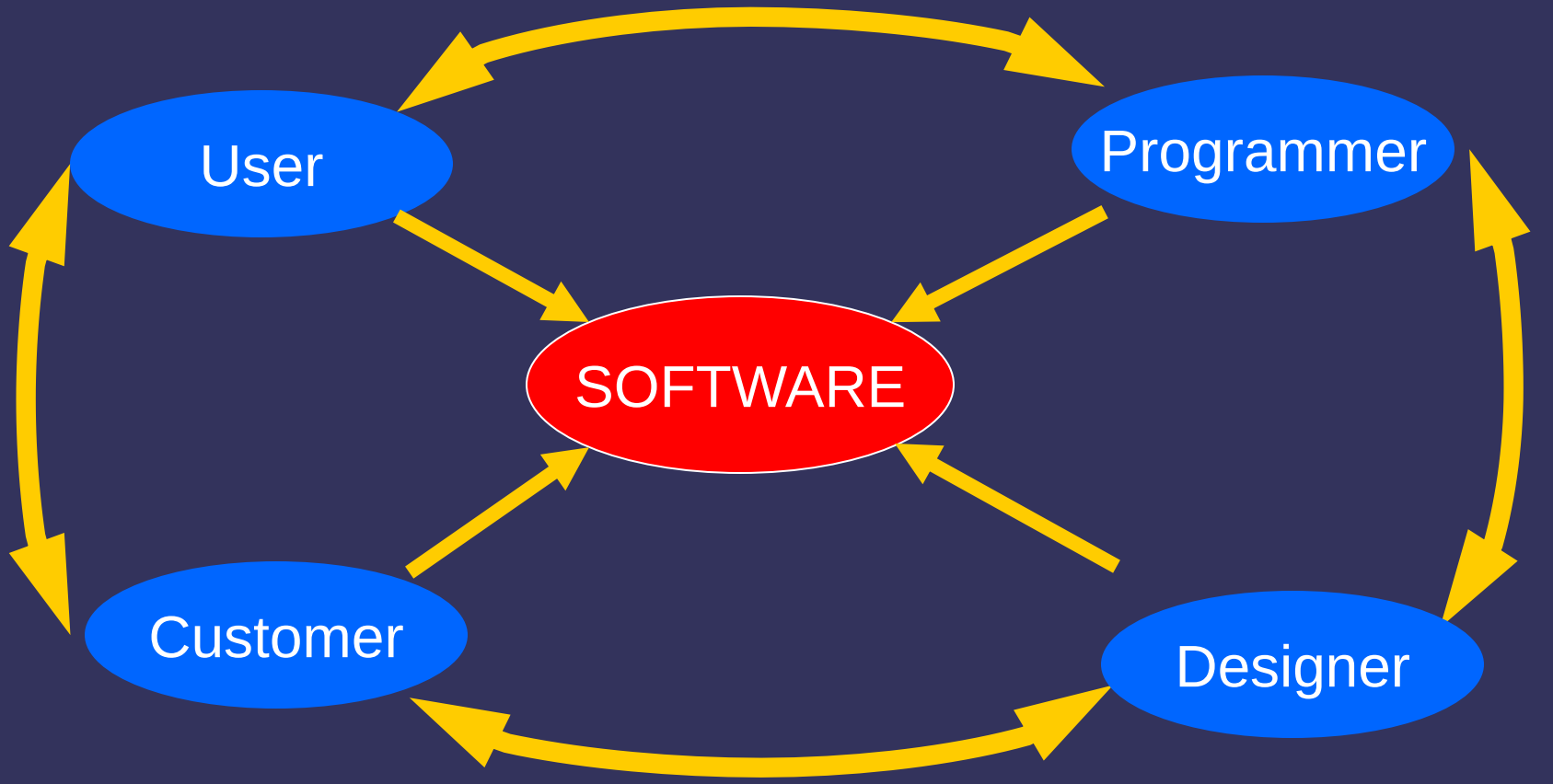
- **Each stage must be completed before next one starts**

Total Feedback



- Too expensive
- Doesn't force to discipline
- Don't show this to your boss !

The Human Factor



Thanks to all