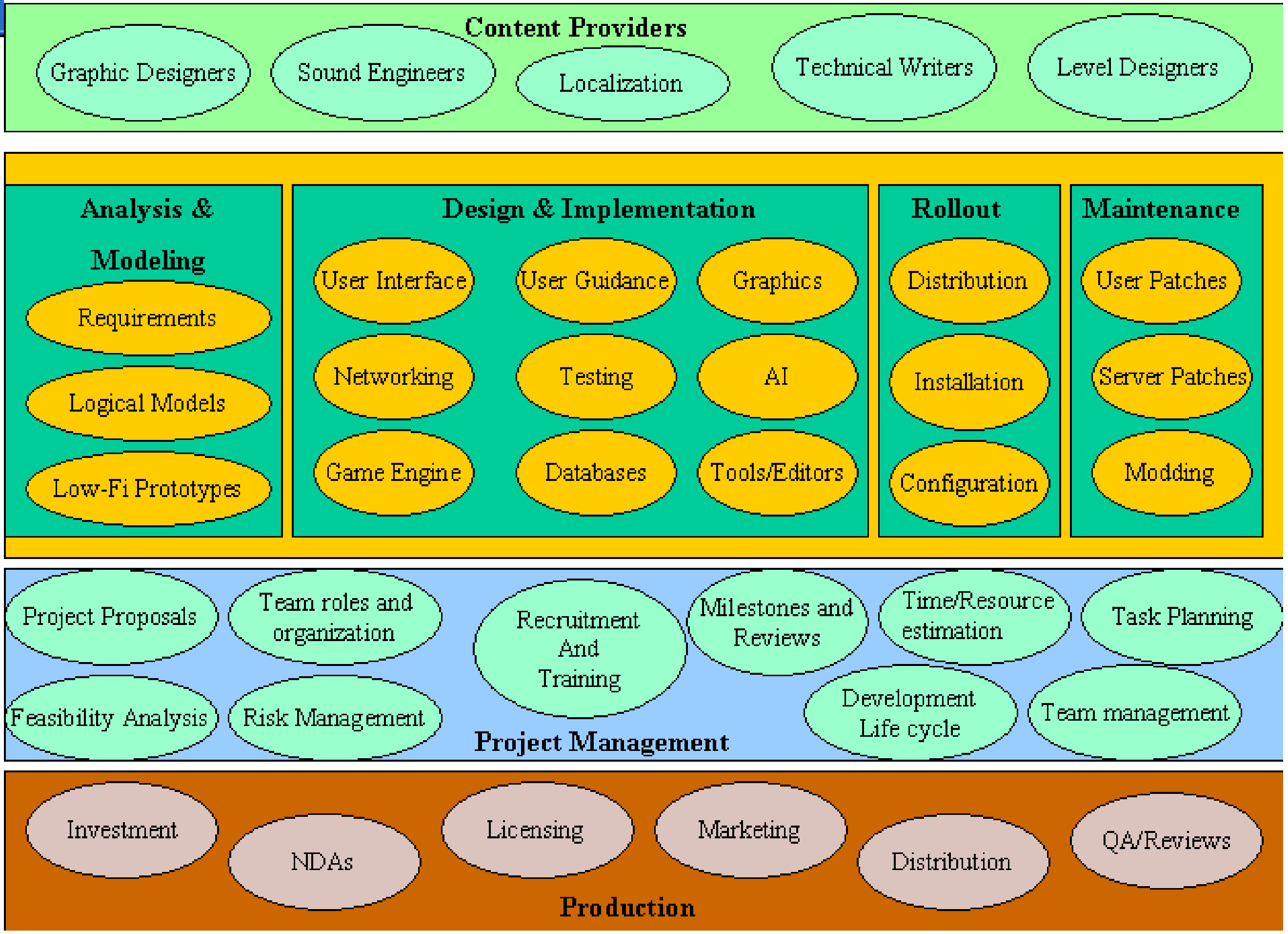
Project challenges

- In first year CS courses, programming exercises are pretty small scale: challenges come from limited experience and time frames
- Actual objectives and techniques to use are (supposed to be) pretty clear
- Rarely required to go back and use your old code as basis for something newer/bigger (so rarely have to clean up old messes)

Scaling up

- Suppose we broaden our horizons... we want to develop a big game or application, using a huge team of people, years of development, years of use/maintenance after it’s created
- Raises a huge collection of issues around conceptual development, actual solution design and implementation, project management, communication, organization, finances, skill and knowledge acquisition

Who’s involved?



Team challenges

- Making sure everyone has same understanding of what we’re building
- Making sure everyone knows their part and is ready on time (plus figuring out what the timeline should be!)
- Making sure everyone is on budget (plus figuring out what the budget should be!)
- Actually finding and/or training everyone we need
- Dealing with people management issues

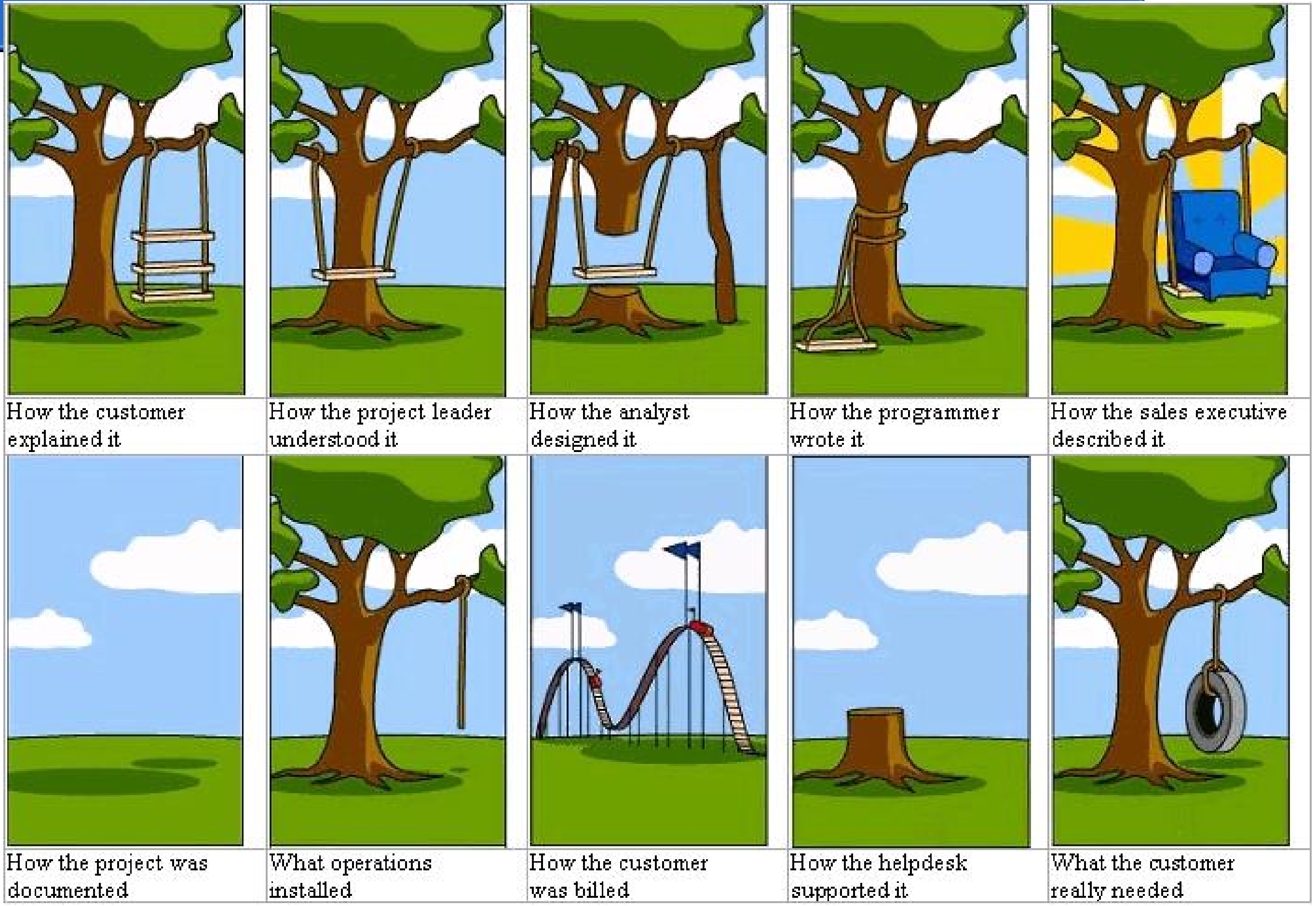
Conceptual challenge

- How do we decide exactly what it is we need to create?
- Need involvement of whoever will be using it (what do they need/want)
- Need agreement of whoever will be paying for it (how much are they willing to pay, when do they want it)
- Need buy-in from whoever will create it (what is/is not possible for the given team, budget, timeline, technology)

Communication challenge

- Need to document exactly what we’re trying to create, in enough detail that every single person involved understands exactly what their part has to be
- The documentation has to be clear, complete, unambiguous, understandable to all these different groups of people
- The document needs to be reviewed by the different groups to make sure it’s actually correct
- The document (and people involved) needs to be kept up to date when things change as the project rolls along

Perspective is everything...


How the customer explained it
How the project leader understood it
How the analyst designed it
How the programmer wrote it
How the sales executive described it
How the project was documented
What operations installed
How the customer was billed
How the helpdesk supported it
What the customer really needed

Technical challenges

- Suppose we support multiple platforms, multiple languages, multiple currencies, sites around the world, multiple versions of the software (free, personal, enterprise, etc) and there are hundreds or thousands of different files to track
- How do we track the right versions of the right file in the right language for the right platform?
- How do we develop and test the parts, and eventually the final product, to make sure we actually got it right?
- How do we track and fix bugs, and add features/updates over time?

This course begins to address the challenges...

- Writing software specifications
- Following common software development life cycles
- Effective use of scripting to automate common tasks
- Effective use of debuggers and code profilers
- Developing and applying test plans, with automation
- Following sound design and coding principles
- Creating/enforcing code/document standards
- Creating effective user interfaces
- Examining code maintenance and refactoring
- Examining localization issues
- Examining deployment and update issues