Waterfall Applications

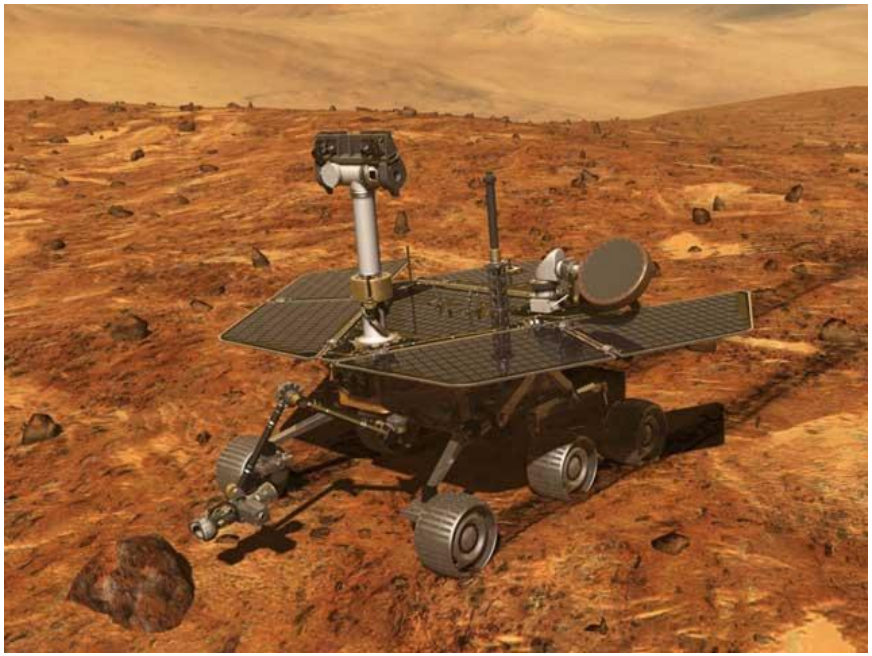
- Works best with stable requirements and technologies
- Not a bad choice for routine IT-like projects
- Arguably useful in any project benefitting from up-front plans

Waterfall Applications

- Arguably the best choice for contractual development
  - Suited executives and their attorneys gather in a conference room
  - Contracts are signed, specifying
    - what will be built,
    - when it will be completed,
    - how much it will cost, and
    - penalties for changes

Waterfall Applications

- Arguably the best choice for contractual development
  - Suited executives and their attorneys gather in a conference room
  - Contracts are signed, specifying
    - what will be built,
    - when it will be completed,
    - how much it will cost, and
    - penalties for changes
  - Widely used in
    - government,
    - aerospace and some
    - enterprise IT



Waterfall Disadvantages and Practical Issues

- Changes waste the painfully created up-front planning
- Inflexible partitioning of development into gated stages
  - may idle resources (e.g., next stage cannot start before current one)
- And, if not used, attempts to achieve the benefits of an agile process without the activities required to be agile

Waterfall Disadvantages and Practical Issues (contd.)

- Big-Bang Integration, if actually used, leads to chaos (Imagine… if we each wrote one *War and Peace* chapter independently, then merged, and sent them to our publisher!)

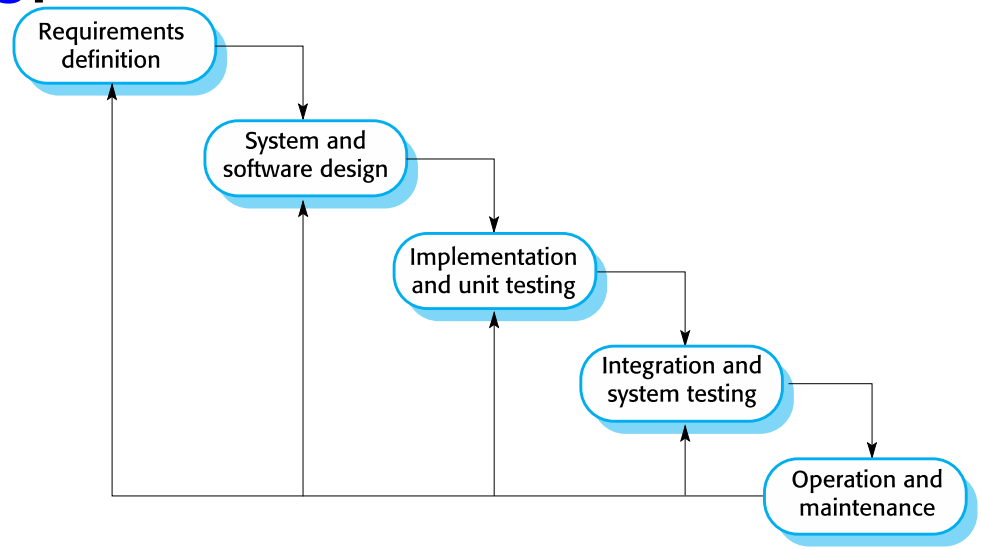
- Invisible problems (e.g., we built the wrong product) may lead to complete failure

Waterfall still exists…

- still a standard
- software engineering textbooks still based on it
- many managers still adhere to it
- rarely followed by the programmers

Waterfall Summary

- Strongly emphasizes up-front planning
- Some phases produce only documents (e.g., requirements)
- Typical sequential phases:
  - Requirements
  - Design
  - Implementation
  - Testing
  - Maintenance



Waterfall Summary

- Each phase built on the planning of a previous phase
- May not produce any code until everything is planned in detail
- Builds the product with "Big bang" integration of code modules

How Would a Team Write *War and Peace*?

How Would a Team Write *War and Peace*?

- English translations contain ≈ 560,000 words
- Tolstoy had 10 years to write *War and Peace* solo

How Would a Team Write *War and Peace*?

- English translations contain ≈ 560,000 words
- Tolstoy had 10 years to write *War and Peace* solo

- 13 Software Engineers developed
  - 115 KLOC (≈ 575,000 "words") in 14 months
  - maybe… perhaps… 575,000 "words" ≈ *War and Peace*

How Would a Team Write *War and Peace*?

- English translations contain ≈ 560,000 words
- Tolstoy had 10 years to write *War and Peace* solo

- 13 Software Engineers developed
  - 115 KLOC (≈ 575,000 "words") in 14 months
  - maybe… perhaps… 575,000 "words" ≈ *War and Peace*

- Imagine writing *War and Peace* using 13 authors in 14 months!

- That's routine for a software engineering team!!!

How Would a Team Write *War and Peace*?

- 13 Software Engineers developed 115 KLOC in 14 months, maybe… perhaps… 575,000 "words" ≈ *War and Peace*

- How many LOC will an engineer write on average per workday? (Solution: whiteboard only)

Group Exercise: Write *War and Peace* in One Year

- Form teams of ~8 people each
- It's 1865 and your publisher has good news and bad news: while agreeing to publish your novel, *War and Peace*, they need the first draft in one year rather than ten

- No problem… you'll simply use as many authors as you need and will be done in a mere fraction of the time!
- Decide how your team will divide the work, coordinate the storyline, and deliver a proofread draft on-time
- Also… your publisher wants French and English drafts in addition to Russian