Informatics 43

LECTURE 3

“HOW DO WE KNOW WHAT TO BUILD?”

Last lecture (I)

- Software engineering can be looked at from different perspectives
  - Business
  - Engineering
  - User
  - Others
- Essential ingredients of software engineering
  - People + Processes + Tools
- Inf 43 principles of software engineering
  - Rigor/formality, separation of concerns (modularity, divide and conquer, abstraction), anticipation of change, generality, incrementality

Last lecture (II)

- No Silver Bullet
  - Software engineering is hard because of the essential difficulties of software
    - Complexity, conformity, changeability, invisibility
  - "Potential silver bullets"
    - Buy vs. build
    - Requirements refinement/rapid prototyping
    - Incremental development
    - Great designers

Today's Lecture – **How do we know what to build?**

- Software failures
- Why requirements?
- Requirements engineering
  - Requirements phase
  - Requirements analysis
  - Requirements specification (documentation)

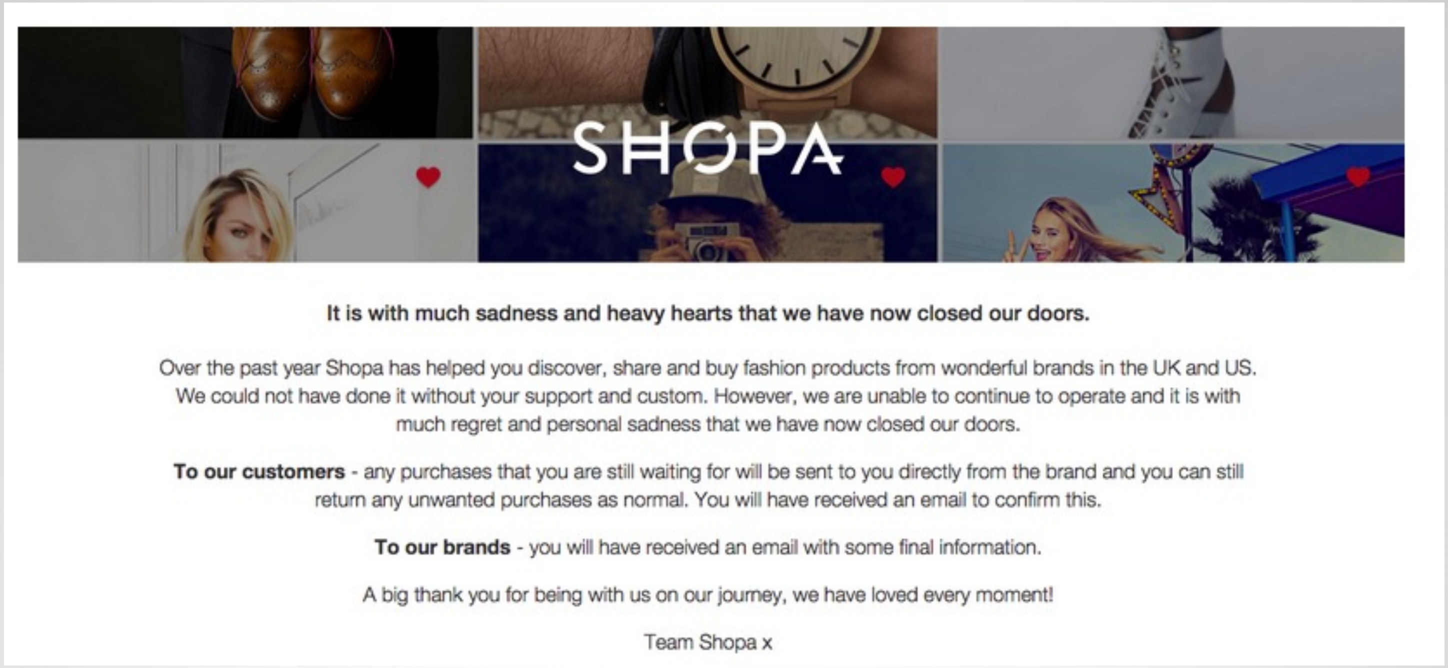
Today's Lecture – **How do we know what to build?**

- Software failures
- Why requirements?
- Requirements engineering
  - Requirements phase
  - Requirements analysis
  - Requirements specification (documentation)

Failure Readings: What is the real problem?

- Complexity
- Conformity
- Progress is hard to measure
- “Legacy” systems
- “Hubble Psychology”

Shopa Failure



It is with much sadness and heavy hearts that we have now closed our doors.

Over the past year Shopa has helped you discover, share and buy fashion products from wonderful brands in the UK and US. We could not have done it without your support and custom. However, we are unable to continue to operate and it is with much regret and personal sadness that we have now closed our doors.

To our customers - any purchases that you are still waiting for will be sent to you directly from the brand and you can still return any unwanted purchases as normal. You will have received an email to confirm this.

To our brands - you will have received an email with some final information.

A big thank you for being with us on our journey, we have loved every moment!

Team Shopa x

"The company, in a lot of ways, is still trying to figure out what product to build. This leads to massive amount of anxiety and ambiguity as no one knows inside the company what they are building. - There is a lot of unspoken strife between the technical implementation and sales teams - Engineers and Sales team running around as headless chickens."

Source: http://www.businessinsider.com/shopa-british-startup-winding-down-2015-8?r=UK&IR=T

Some stats…

"We see from our data that five out of 10 projects deliver on budget and on time. But we also see that on average, cost overruns double their cost and about 34 percent of projects are delivered late. That really begs the question "What is the problem in IT?" because every second IT project is just doing fine. "

-Jurgen Laartz (https://www.computer.org/csdl/mags/so/2016/04/index.html)

More stats…

- From 2015 CHAOS reports
  - Only 29% of projects considered successful
  - 52% “challenged”
  - 19% failed
  - Billions wasted per year on cancelled, unused or unusable projects

Attendance Quiz

Today's Lecture – **How do we know what to build?**

- Software failures
- Why requirements?
- Requirements engineering
  - Requirements phase
  - Requirements analysis
  - Requirements specification (documentation)

Build me a todo list app…

Build me a todo list app…

- Users shall be able to add to do list items with a single action.
- To do list items must consist of text and a binary completed state
- Users must be able to edit the to-do list item text
- Users must be able to toggle the completed state
- Users must be able to delete to-do list items
- All edits to to-do list item state must save without user intervention

Build me a todo list app…

- Users shall be able to add to do list items with a single action.
- To do list items must consist of text and a binary completed state
- Users must be able to edit the to-do list item text
- Users must be able to toggle the completed state
- Users must be able to delete to-do list items
- All edits to to-do list item state must save without user intervention

Complete? Unambiguous? Non-conflicting? Verifiable?

Build me a todo list app…

- Users shall be able to order a list however they wish
  - By default, a new list item is added at the top of the list.
- Users shall be able to add to do list items with a single action.
- To do list items shall consist of a textual name (required), a binary completed state (default not completed), and an optional longer description.
  - Name max 50 characters and description max 500 characters.
- Users shall be able to edit to do list item text.
- Users shall be able to toggle the completed state.
  - Possible item states are "completed" or "not completed."
- Users shall be able to delete to do list items.
  - Delete shall be undoable during a single use of the app
- All edits to to do list item state shall save without user intervention.
- The average user shall be able to create a list with 3-5 items, edit one item, and mark one completed, all within 5 minutes, without any training.
- Users shall be able to set deadlines for items, and optionally add these deadline to Google Calendar or Outlook
- Users shall be able to share their achievements with a single action, which will post the item and the fact that they completed it, to Facebook.

Complete? Unambiguous? Non-conflicting? Verifiable?

Top Software Failure Causes

- Lack of user input/involvement
- Incomplete requirements and specifications
- Changing requirements and specifications
- Lack of discipline in development processes
- Lack of methodical usage of metrics
- Lack of resources

Reminder: Top Software Failure Causes

- Lack of user input/involvement
- Incomplete requirements and specifications
- Changing requirements and specifications
- Lack of discipline in development processes
- Lack of methodical usage of metrics
- Lack of resources

Lack of rigor/formality

Reminder: Top Software Failure Causes

- Lack of user input/involvement
- Incomplete requirements and specifications
- Changing requirements and specifications
- Lack of discipline in development processes
- Lack of methodical usage of metrics
- Lack of resources

Requirements issues

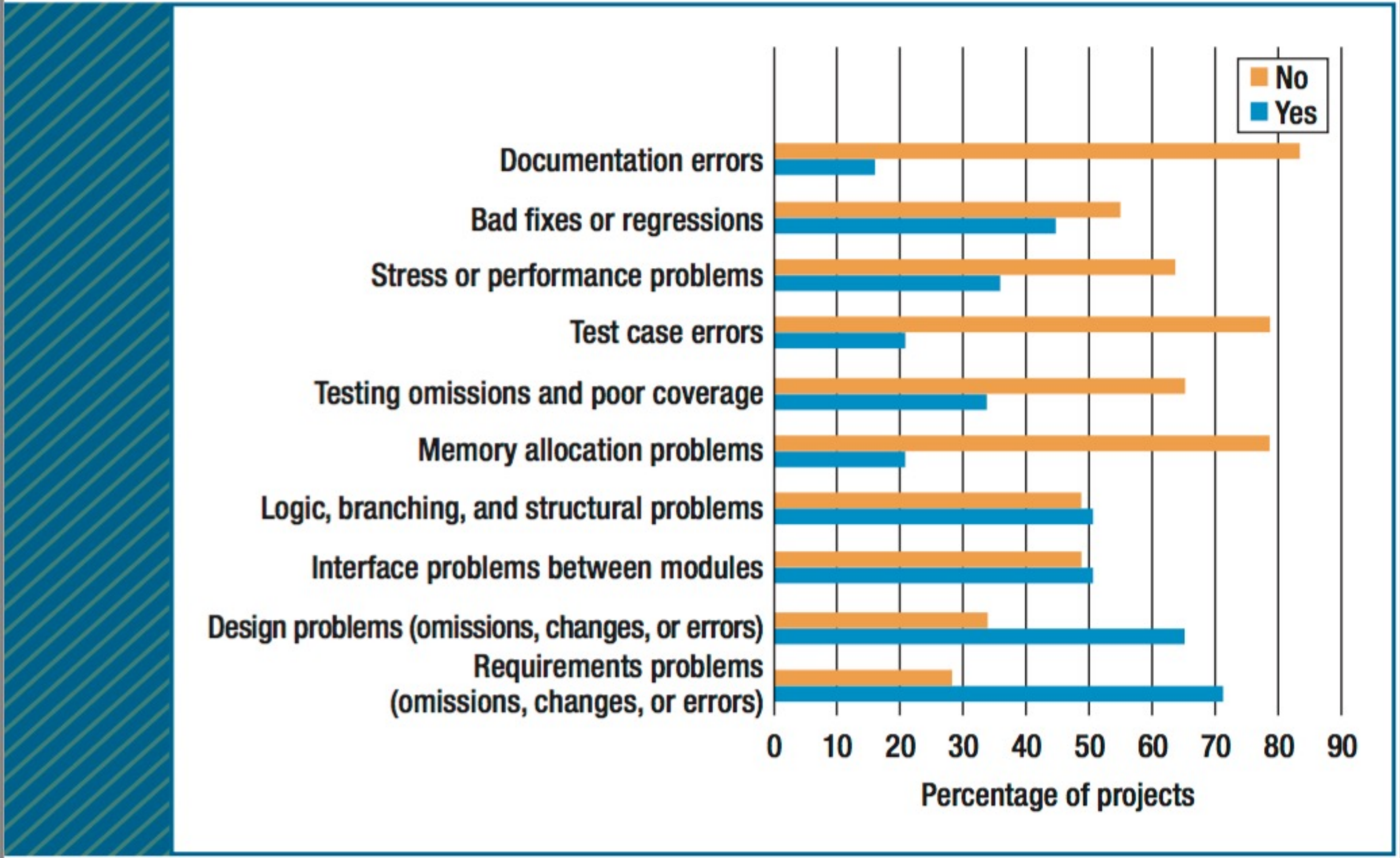


FIGURE 7. Participant responses regarding the causes of discovered defects (98 responses). The most reported cause was requirements problems.

Source: Kassab, DeFranco, LaPlante: "Software Testing: The State of the Practice," IEEE SW, 2017

Definition

Requirements =
what the software should do (without saying **how** it should do it)

Note: requirements address what a customer <u>needs</u>, not what a customer wants

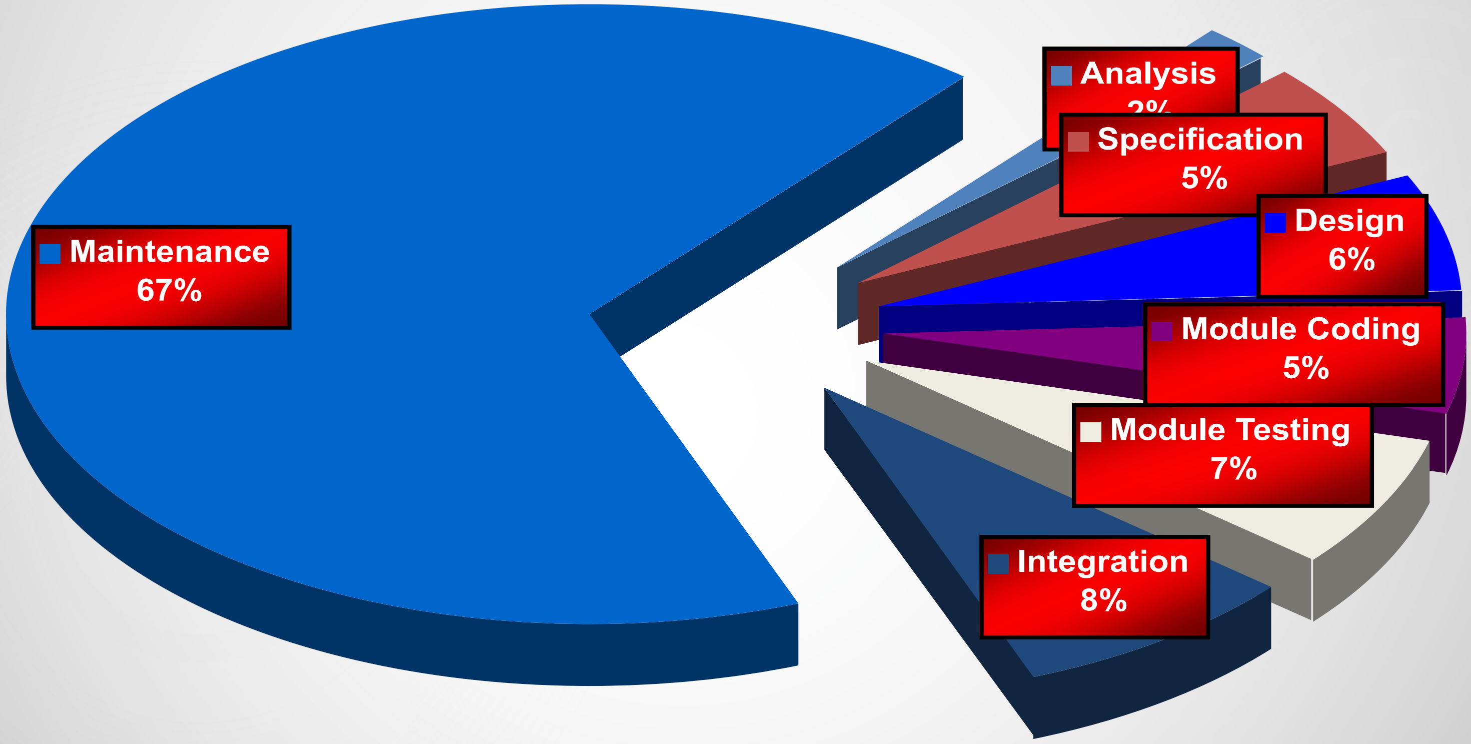
Why Requirements?

- "[We] have grown to care about requirements because we have seen more projects stumble or fail as a result of poor requirements than for any other reason"
  - (Kulak and Guiney, in "Use Cases: Requirements in Context")
- Studies show that many of the key contributors to project failures originate or relate to requirements
  - (The Standish Group CHAOS reports)

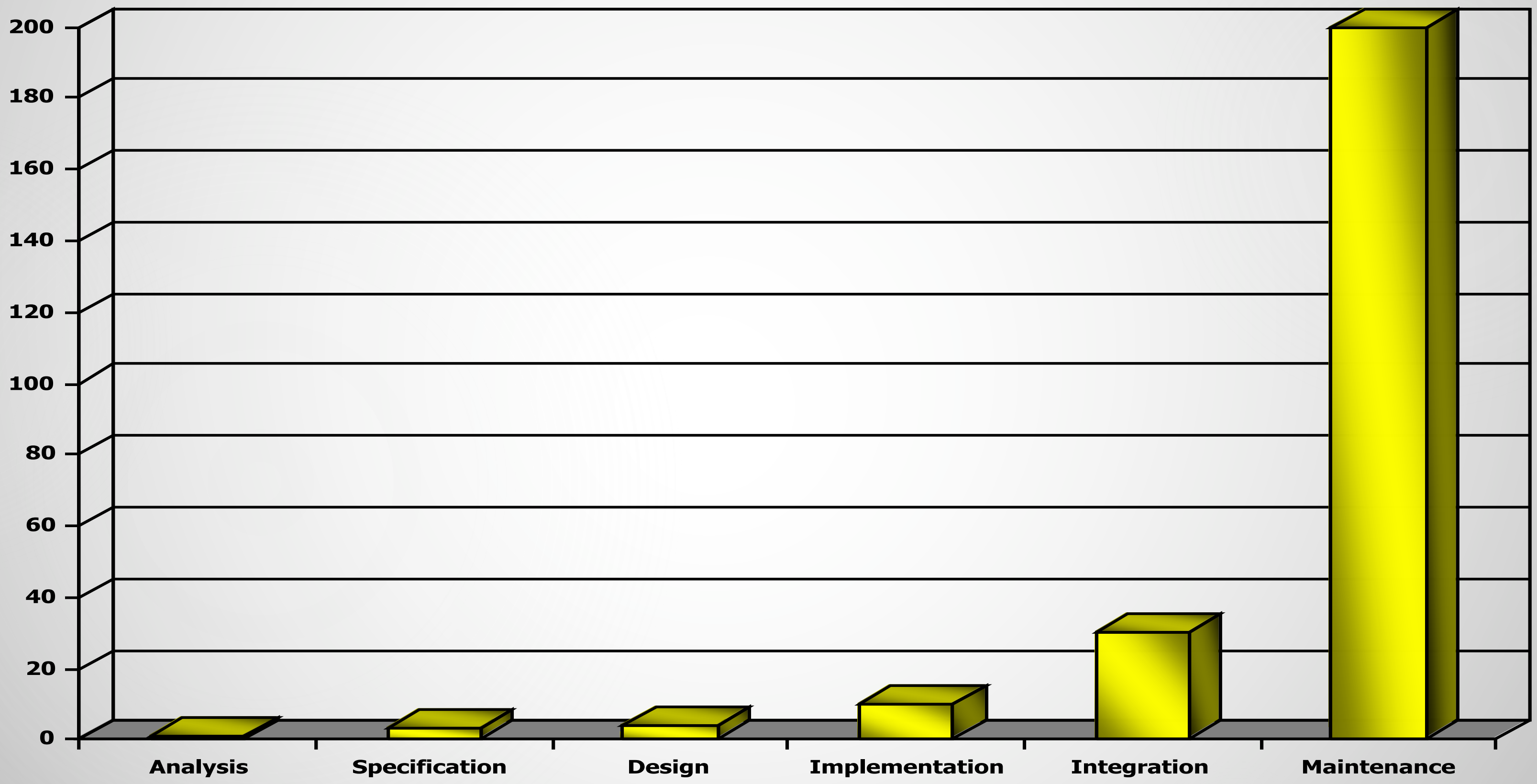
More stats…

- From CHAOS reports: Requirements defects are expensive
  - They represent more than 70% of rework costs
  - Rework consumes about 30-50% of total project budget
  - Lack of user input/user involvement listed as most frequent problem

More Stats: Software Life Cycle Costs



More Stats: Cost of Change Progressively Higher



Today's Lecture – **How do we know what to build?**

- Software failures
- Why requirements?
- Requirements engineering
  - Requirements phase
  - Requirements analysis
  - Requirements specification (documentation)

Which category employs the most computer programmers in the U. S.?

A. In-house staff writing systems for internal use

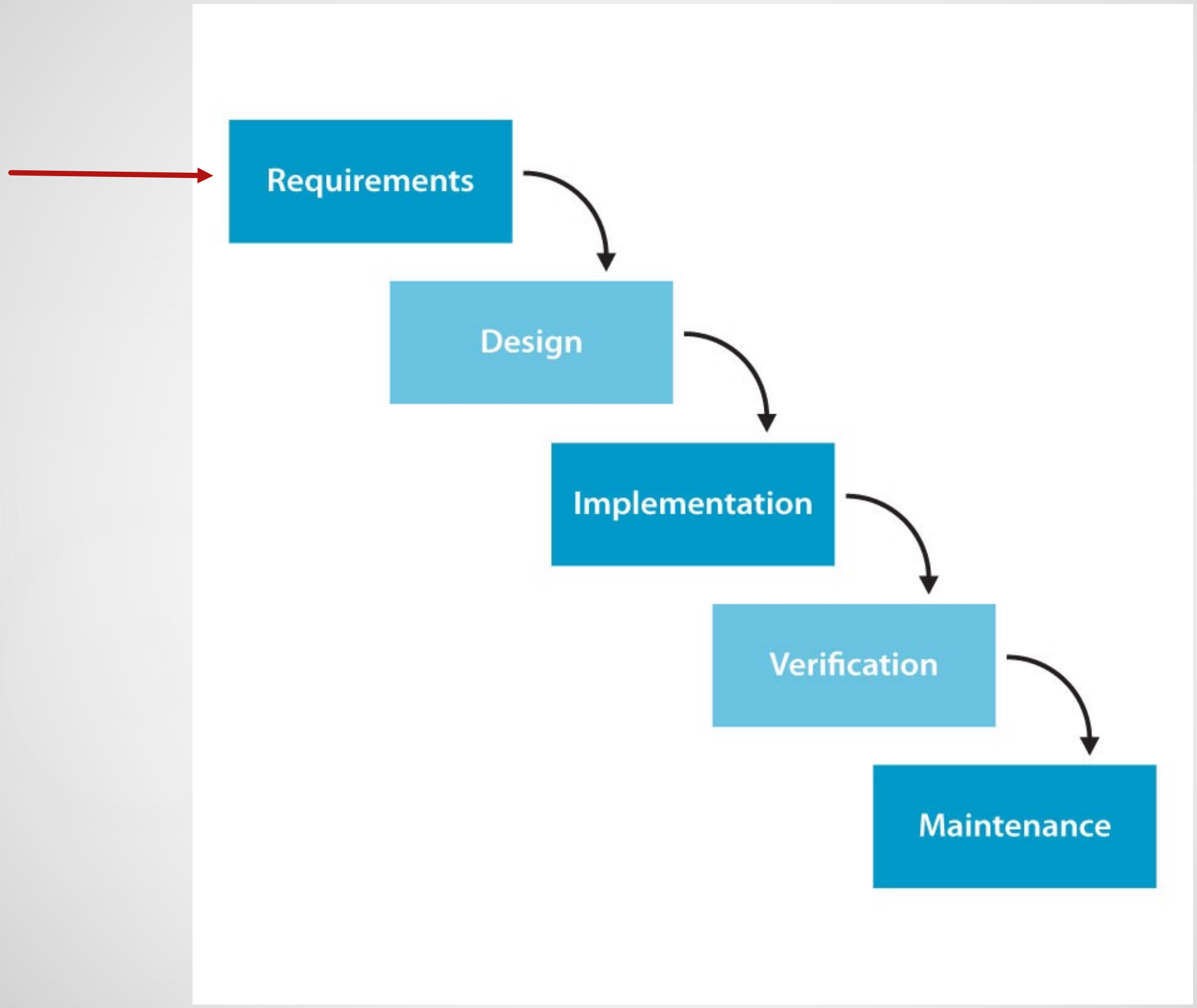
B. Games, apps, productivity software

C. Consulting companies/contract programming

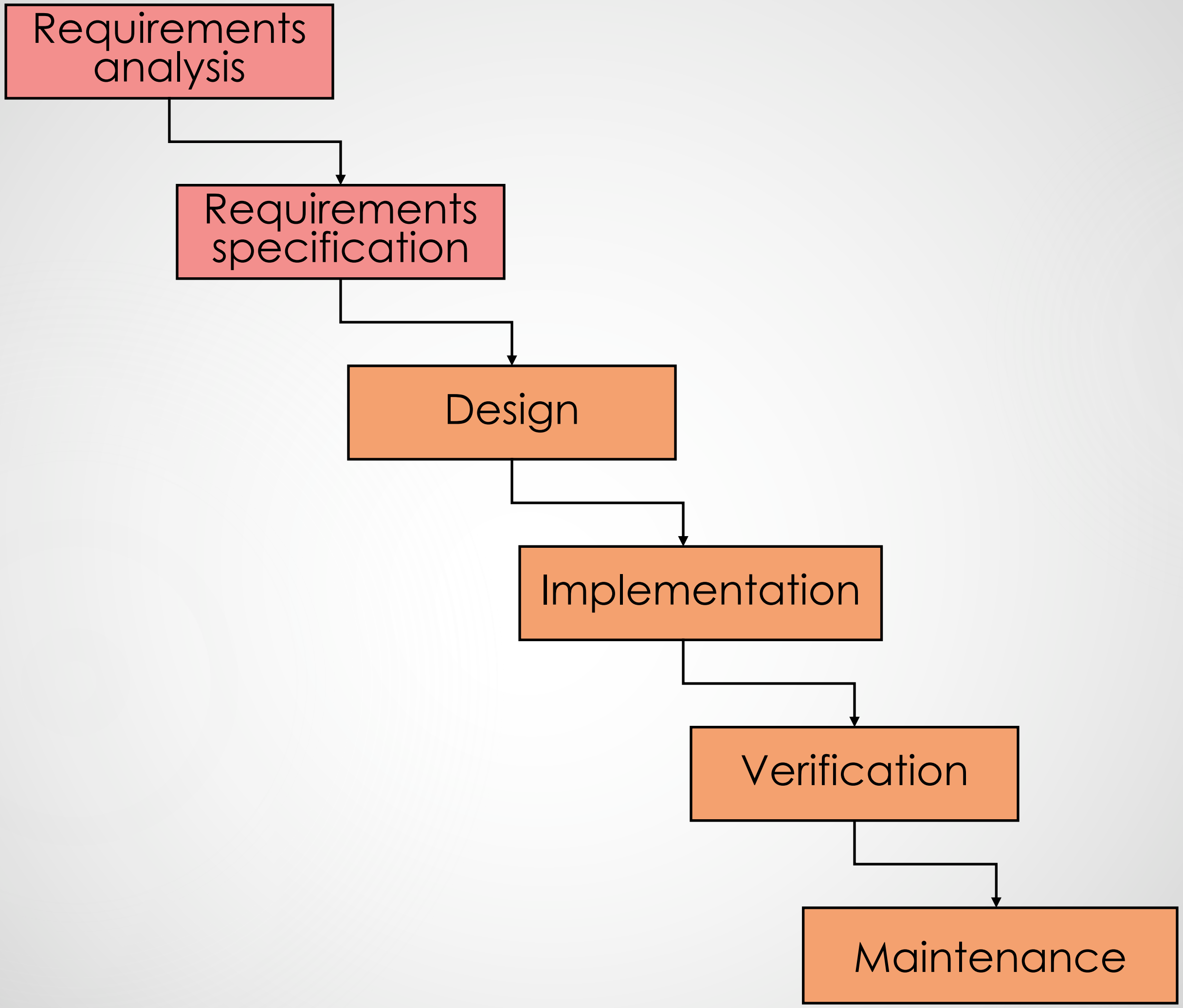
D. Open source projects

E. Creators of viruses and other malware

Waterfall


Requirements
Design
Implementation
Verification
Maintenance

Waterfall


Requirements analysis
Requirements specification
Design
Implementation
Verification
Maintenance

Requirements Phase - Terminology

- Requirements analysis
  - Activity of discovering/observing/gathering customer's needs
- Requirements specification
  - Activity of describing/documenting customer's needs
  - Can also refer to the requirements document

Today's Lecture – **How do we know what to build?**

- Software failures
- Why requirements?
- Requirements engineering
  - Requirements phase
  - Requirements analysis
  - Requirements specification (documentation)

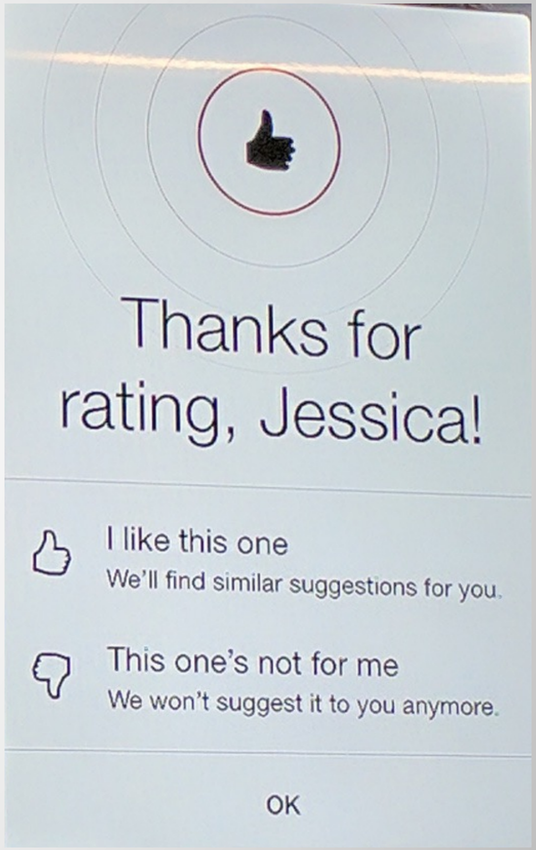
Techniques for Requirements Analysis

- Interview customer
- Observe customer (CNBC article)
- Create use cases/scenarios/user stories
- Prototype solutions
- Identify important objects/roles/functions/goals
- Perform research
- Construct models
- …
- Data (see next slide)

https://www.youtube.com/watch?v=I_GTTyE9i9Y

Data-driven requirements analysis

- Usage data is analyzed and business metrics are recorded to discover the value of new (potential) features
  - Techniques
    - Data analytics
    - A/B testing
    - Prototyping
    - Market research
  - This information is used to create/prioritize/analyze/evaluate requirements



Thanks for rating, Jessica!

I like this one
We'll find similar suggestions for you.

This one's not for me
We won't suggest it to you anymore.

Today’s Lecture – **How do we know what to build?**

- Software failures
- Why requirements?
- Requirements engineering
  - Requirements phase
  - Requirements analysis
  - Requirements specification (documentation)

Summary

- Software failures can be disastrous
- Proper requirements engineering (analysis + specification) is **essential**
- Requirements analysis is learning **what** the system should do (without worrying yet about *how* it should do it)

Next Time

- Requirements Specification (i.e., taking what we learned during the requirements *analysis* phase, and then documenting it)
- Use Cases