One size fits all for software engineering?

Entrepreneur

We are still seeking out a market, so you might choose to change the target of the software quickly

Corporate software

The target market is generally well-known as are the key requirements

Web software

The market reach is broad and you can have trouble identifying everyone using it. Quick feedback and quick turnaround on changes to the market.

Hobbyist

The focus is on the challenging part and the beautiful part

Consulting

You are hired by a specific client for a specific task. The task could change or adapt as the client learns more

Real-time systems

The task is well-defined, but failure is not an option.

Styles of processes

Core systems development 30 years ago

- **Requirements can be mostly identified up-front**
- **The window of opportunity is relatively large or closes far off**
- **…We have the time to do lots of planning in advance**
- **…We're typically developing everything in-house**

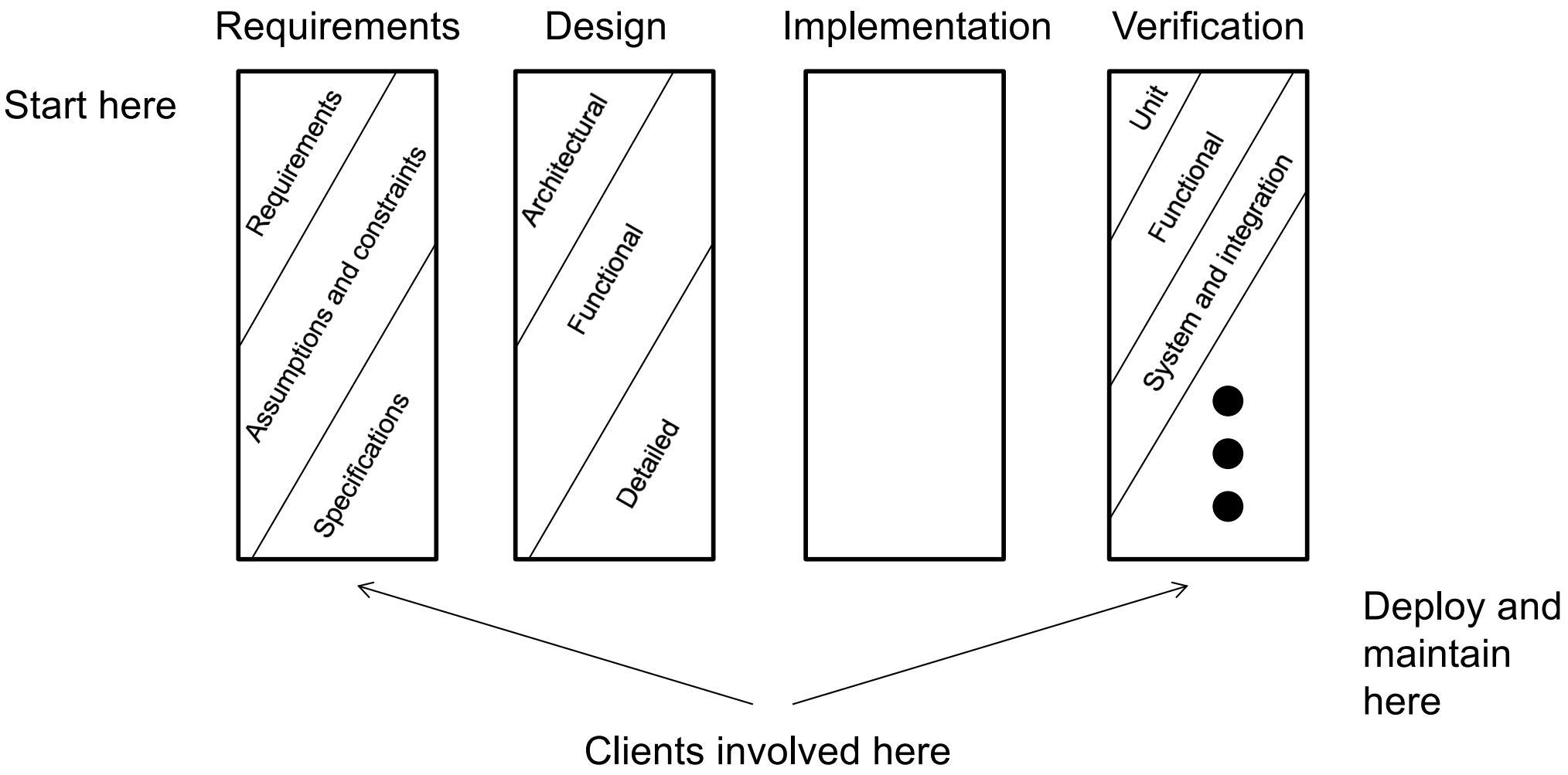
Web development today

- **Functionality changes based on users' reactions**
- **Windows of opportunity open and close quickly**
- **Quick audience attention and attrition**
- **…We release bits incrementally**
- **…We're developing for an existing environment and re-using as much as we can.**

SE Methodology

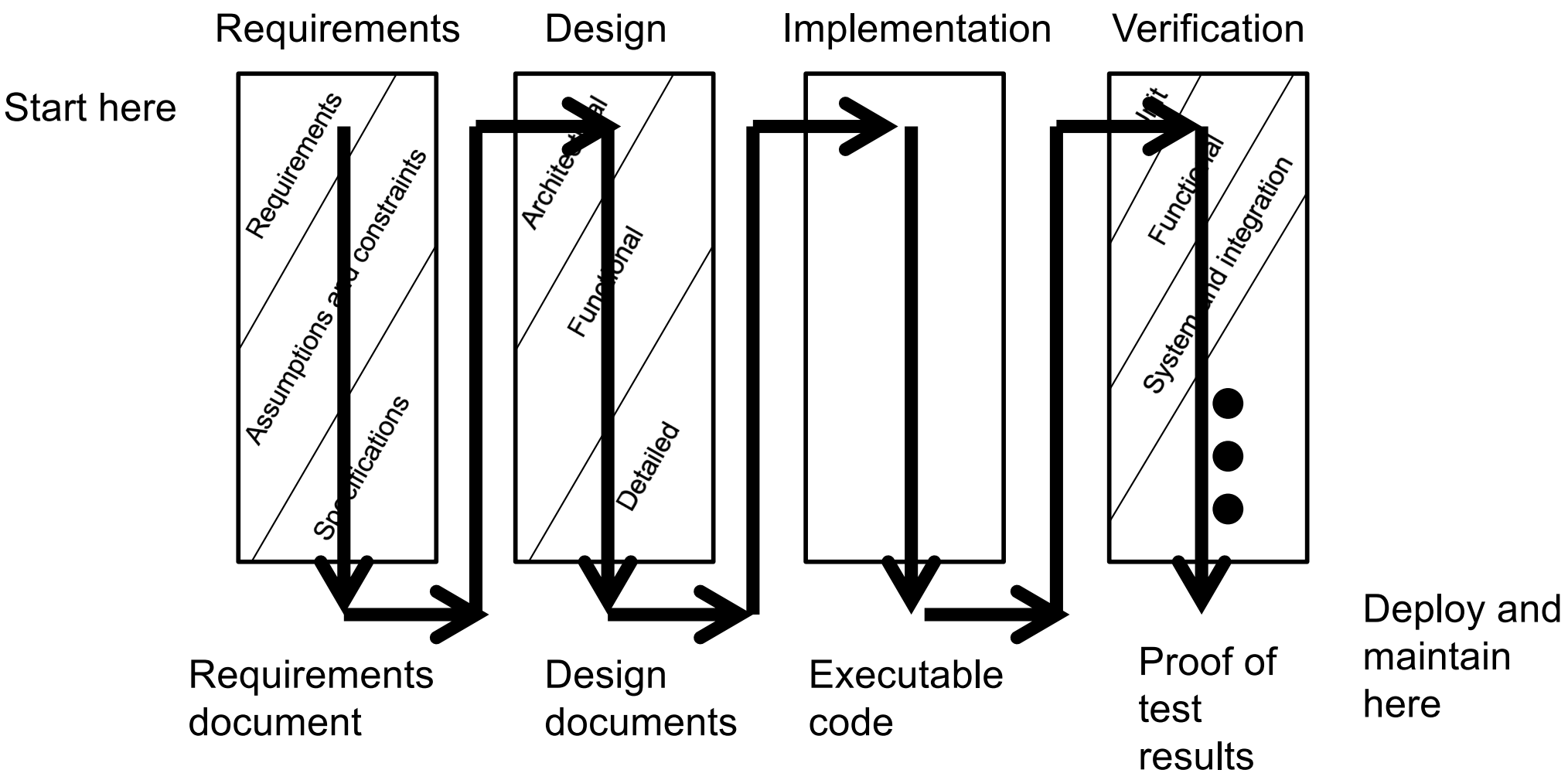
- SE focuses mostly on processes for achieving the goals
- SE separates the process of developing the software from the product that is developed (the software itself)
- Quality and productivity is governed by the people, processes, and technology.

How to you traverse the elements of software development?



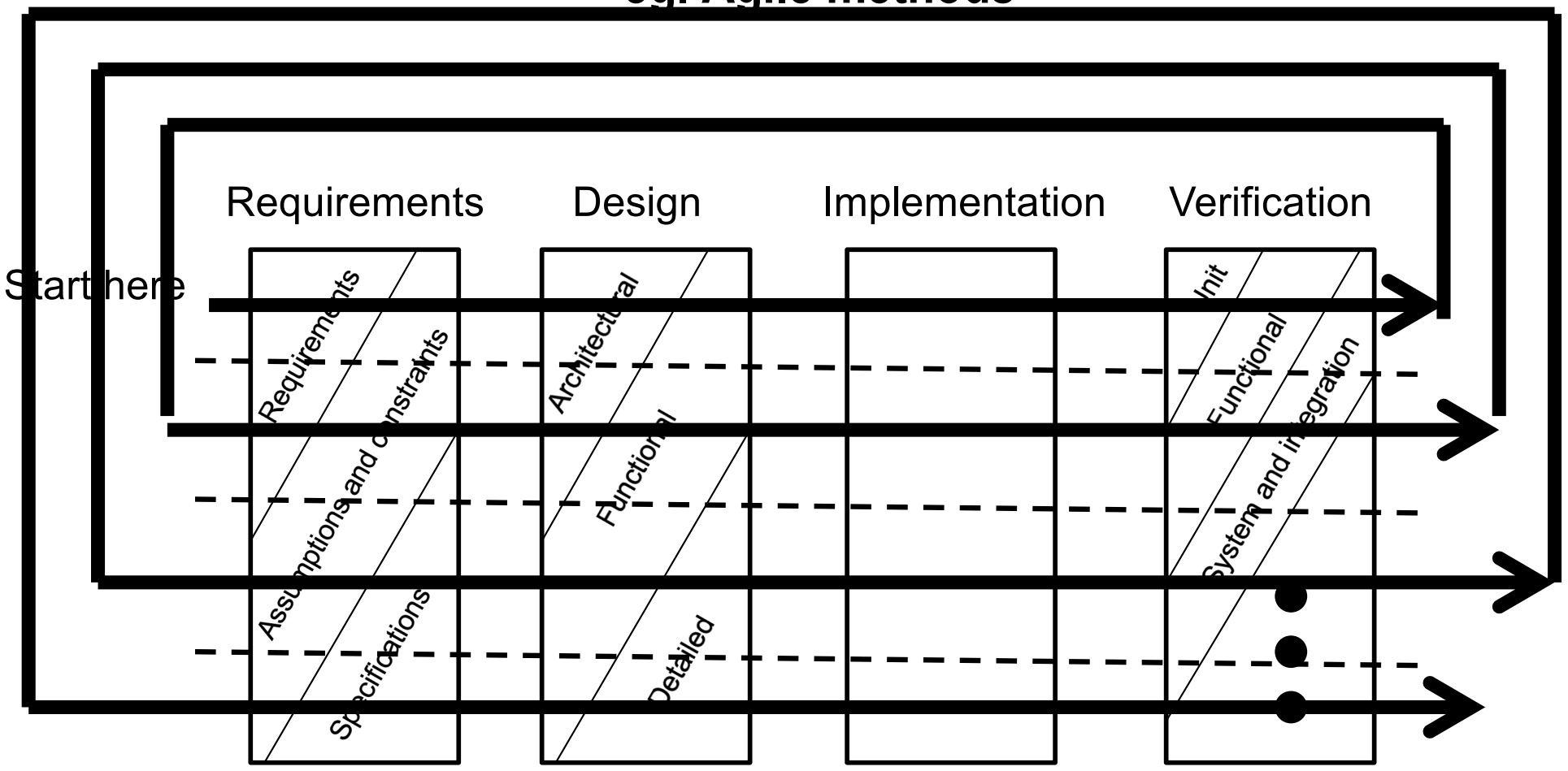
Plan-based development

eg. Waterfall method



Iterative development

eg. Agile methods



Each row is an iteration. Working (but limited) code at the end of each iteration.

Clients involved here

Deploy and maintain here

Tradeoffs

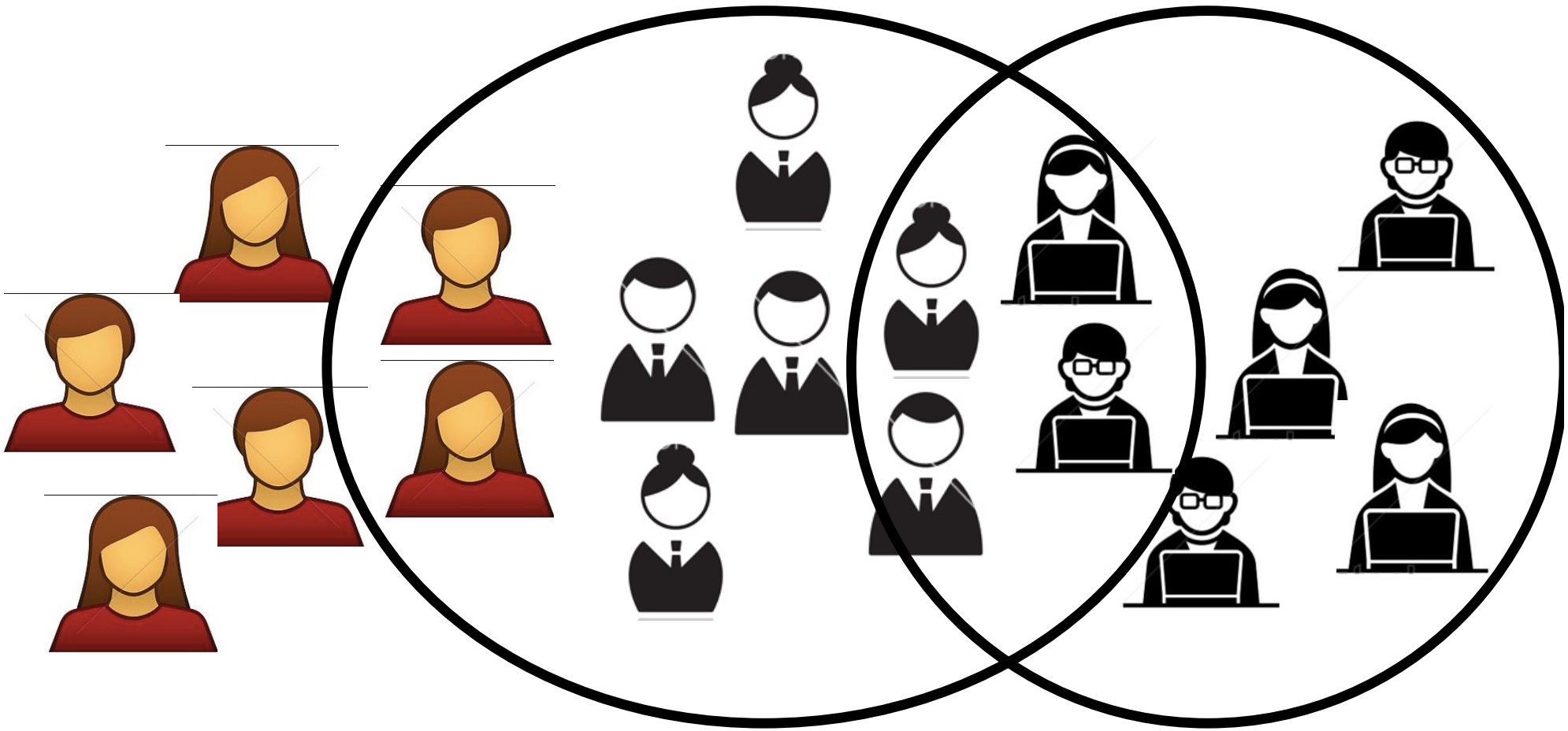
- Plan-based
  - + You get a full picture, so you optimize the design and implementation
  - - Errors detected late in the process are costly to undo
  - - The client sees little until everything is built
- Iterative
  - + Continual client feedback to adjust the direction
  - + Not as much backtracking when requirement or design errors are detected
  - - Feature creep
  - - Rewriting / reorganizing code (called refactoring) is more commonly needed

Agile development

Characteristics of agile development

- Not any one specific method
- Emphasizes teamwork (self-organizing, independent smaller groups) over management
- Values individual creativity and motivation
- Anticipates and welcomes change
- Planning minimized through continual client engagement, short goal-focused iterations leading to releases of working software
- High internal standards of quality
- Devoted to simplicity (in design and process)

Agile interaction



Business value proposition development team

Agile development team

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

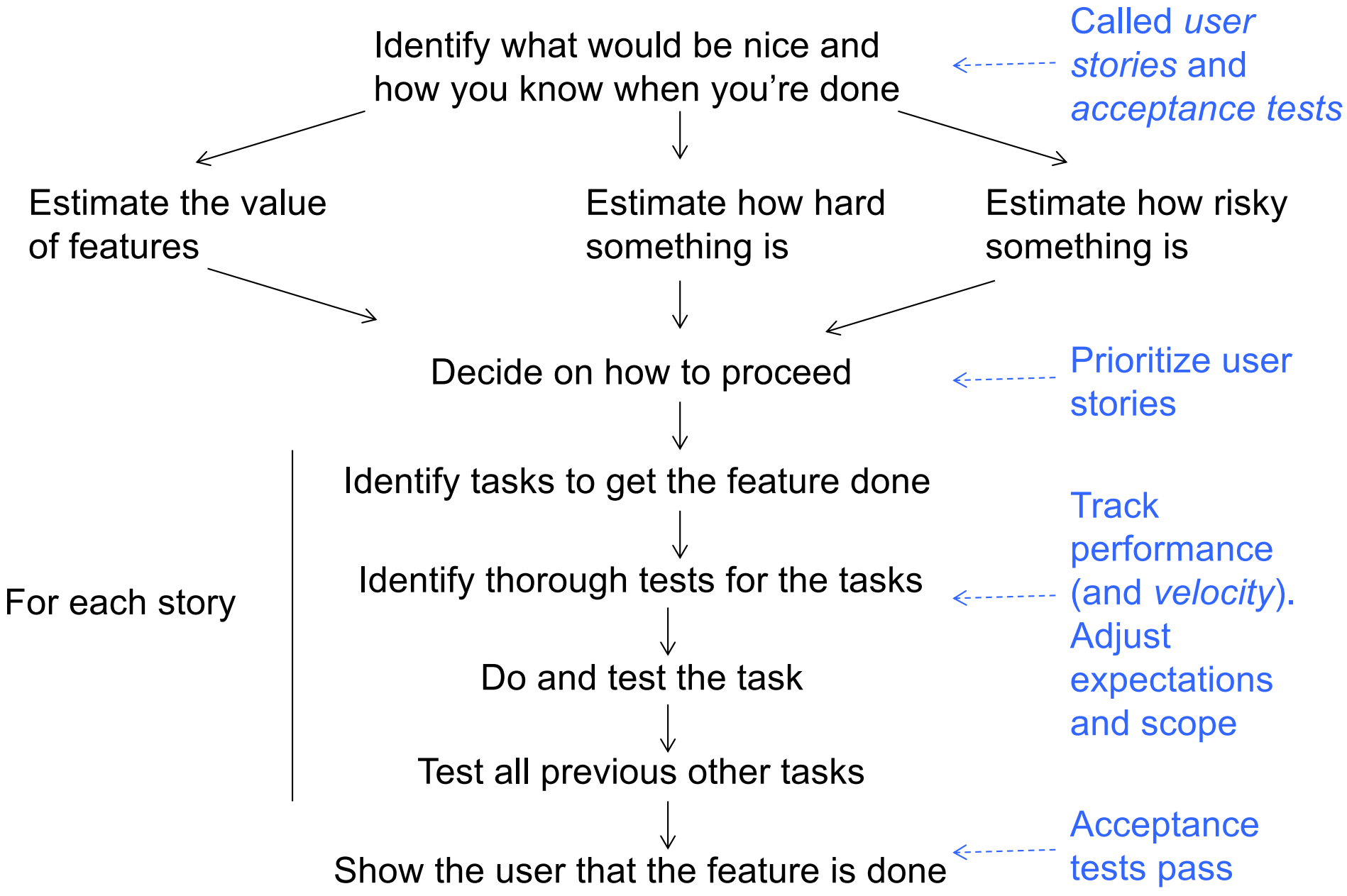
Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

http://agilemanifesto.org

Agile manifesto

- **What the Agile manifesto does not assert:**
  - Don't bother with any process
  - Write all your code from scratch
  - The code is all the documentation you ever need
  - Whatever the customer says to do, whenever they ask it, just do it
  - You don't need to have a plan
  - "If I'm ahead of schedule then I should just add more features to justify not having to do documentation and planning because more (working) code is better."

Agile Iteration Overview



Estimation in Agile projects

- Incremental releases do not avoid cost overruns, but cost issues (scheduling issues, scope issues) become clear early in the project
- An emphasis on complexity and customer engagement should translate into more value for the customer
- Estimation is embodied in the concept of *velocity*
  - Teams continually assess and revise their velocity measure
  - Iteration plans permit fine-grained estimation and feedback
  - Selection of features for an iteration plays a key role in estimation and communicating costs with the customer

How do you do estimation?

- Brainstorming and gathering techniques
- Have all voices heard
  - Sometimes one person may see complexity or simplicity that others don't see

How do you do estimation?

- As a group, order the stories by difficulty and then assign complexity given the ordering
- Individual estimates of the complexity
  - Rank 1-6; high, medium, low; or some other way
  - Compare results; talk about the complexity in extreme situations
- Play the dots game with them
  - Post all of the user stories
  - Give everyone a fixed number of red and green sticky dots
  - Have everyone place dots on cards for easy (green) or hard (red) tasks; unmarked are medium
  - Rank the results by the number of dots

Monitor Your Estimates

- **Monitor the progress with which you complete tasks**
  - Determine the consistency of your estimates
  - Determine a baseline for what your team can deliver

- **Measure your rate of completion to determine a team velocity**

DALHOUSIE UNIVERSITY
Inspiring Minds