PURDUE UNIVERSITY®

CS 307: Software Engineering

Lecture 3: Software Life Cycles and Scrum

Prof. Jeff Turkstra

Software Development Cycle

- Begins with decision to develop software product
- Ends when the software is delivered
- Typically includes
  - Requirements phase
  - Design phase
  - Implementation phase
  - Test phase
  - Installation and checkout phase

Software Life Cycle

- Software Development Cycle
- And
  - Operation and maintenance phase
  - Retirement phase

Process models

- "The primary functions of a software process model are to determine the order of the stages involved in software development and evolution and to establish the transition criteria for progressing from one stage to the next." – Boehm

Process model questions

- What should we do next?
- When should we start doing it?
- How long should we continue to do it?
- How will we know when we are done?
- How well does it handle change?
- How are decisions made?

Why?

- To define the project activities
- For consistency
- To set up meaningful management checkpoints

Observations

- The life cycle is there to help (guide) the project team with their project activities
- The life cycle is a roadmap showing how to get successfully from the initial idea to the final retirement of the product
- People must still manage the project and carry out the tasks

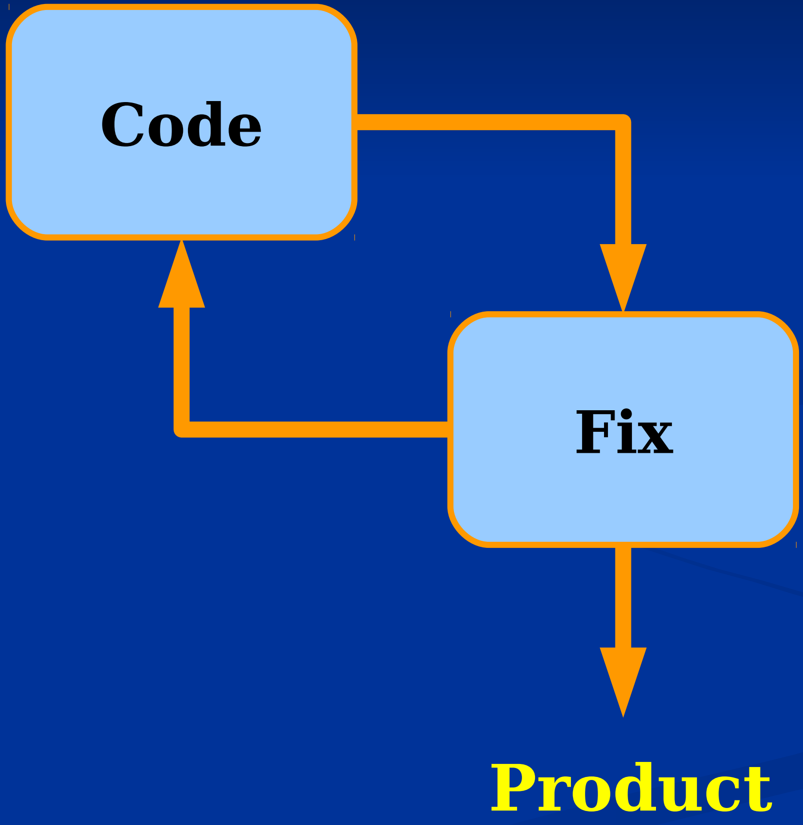
Observations

- Software methodologies focus on how to navigate through the various phases of a life-cycle model
- They also deal with the physical representation of the various phase products
- Software process models provide the guidance on the order in which the major events in the life-cycle are carried out

Models

- Code and Fix
- Stagewise and Waterfall
- Prototyping
- Evolutionary
- Spiral
- Agile/Scrum

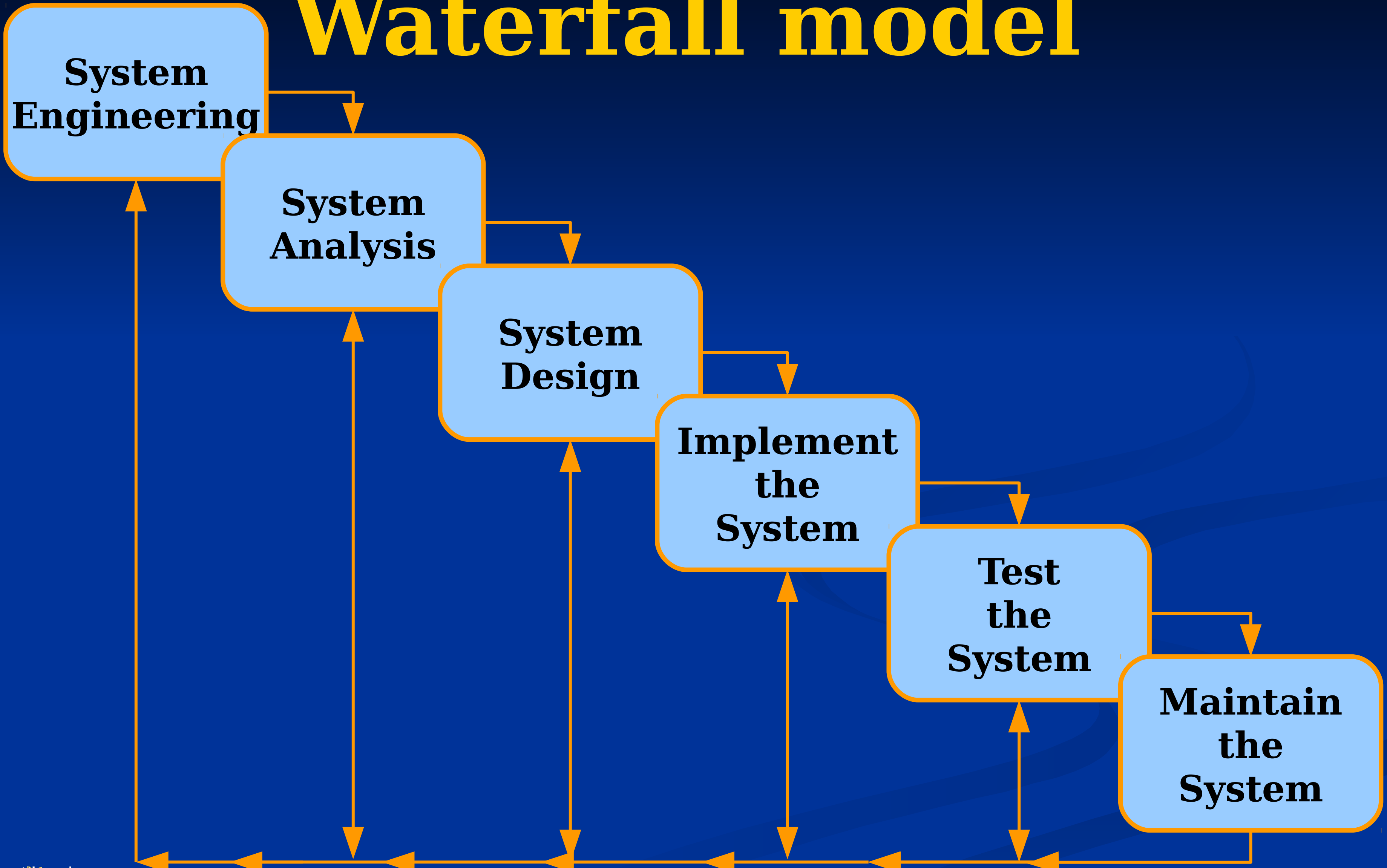
Code and fix



Code and fix model

- Code a little
- Fix a lot
- Repeat until your time, money, people, and customers run out
- Good luck
- It’s time to update your resume

Waterfall model



System engineering activities

- Business considerations
- Technical considerations
- Manufacturing considerations
- People
- Environmental considerations
- Legal considerations
- Develop an installation plan

System analysis

- This is the what part
- Identify the real system requirements
- Hopefully work with and involve the user
- Develop an acceptance test plan

System design

- This is the How part
- Identify the system specifications
  - Hardware
  - Software
- Develop the system integration test plan

System implementation

- Build the system
- Test the parts
- Carefully complete all system documentation

System test

- Carry out system integration test plan
- Carry out the acceptance test plan
- Carry out system installation plan
- Establish system regression test library

System maintenance

- Manage change
- Fix defects
- Port the system
- Enhance the system
- Retire the system

Weaknesses of classic waterfall

- Difficult to measure real progress and manage change
  - Nothing is really done until the end
  - Serious design flaws are usually found toward the end of the cycle
  - Testing tends to be done last and is often short circuited due to lack of money and time to market

- Tends to push much of the real analysis and design feedback into the maintenance phase
  - We will fix it in the next release
  - We will add it to the next release
  - Let the user test it for us

- Document driven up front
- Form seems to be more important than content
- Feedback paths are not used very much

- Assumes an unrealistic sequential progression through the cycle
  - People tend to be specialists. They tend to lose sight of the big picture.
  - People are very mobile. They are gone when the questions are asked and the flaws discovered.
  - Tends to be bureaucratic. It is a document driven model.

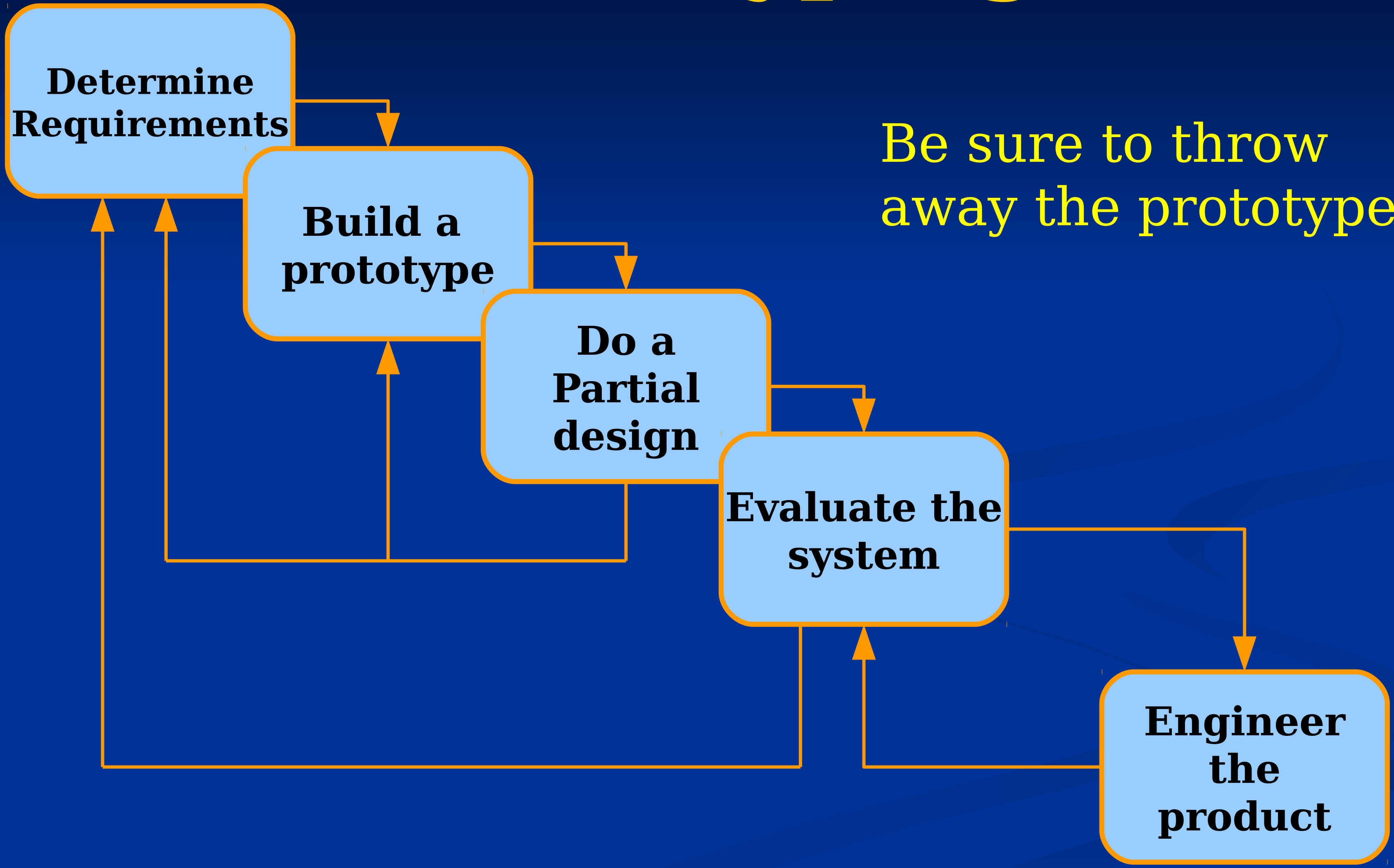
- Top-down approach works as long as the number of levels between top and bottom are small
- Too much focus on justifying work and meeting milestones and not enough focus on doing the work
- This model is not able to respond to problems (change requests) in an efficient and timely manner

When should it be used?

- For systems which are well understood?
- In environments with a good software development process?
- Small start up projects?
- When a high degree of risk and a great amount of accountability is required?

- As a teaching tool to introduce students, in an orderly manner, to the activities required for a successful software project?
- Maybe never?

Prototyping



Be sure to throw away the prototype

Why prototype?

- When the user is not very computer literate
- When the user is unable to completely pre-specify the needed system requirements
- When there is little algorithmic detail
- Whenever you are not sure your design will work

Evolutionary development models

- Can become a modern version of code and fix
- Product evolves over time as the real requirements become known
- Assumes the customers will tell us what they like and dislike about each incremental release
  - Fix and enhance the product in next go around

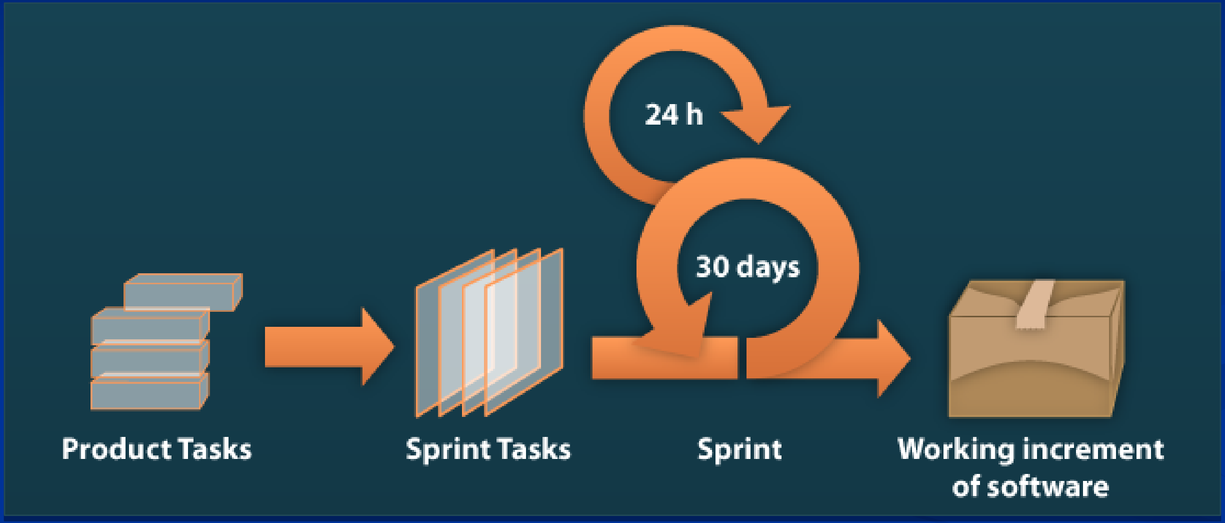
The Right Way (TM)

- There is no single best process model for developing high quality software systems

Slides

- The following slides are based on a presentation created by Kevin Schenk, BS in Computer Science, 2012

Scrum


24 h
30 days
Product Tasks
Sprint Tasks
Sprint
Working increment of software

Roles

- Product owner
- Scrum master
- Development team

Product owner

- Defines the features of the product
- Decides on release date and content
- Prioritizes features
- Adjusts features and priority every iteration, as needed
- Accepts or rejects work results

Scrum “master”

- Represents management to the project
- Responsible for ensuring adherence to scrum practices
- Removes impediments
- Ensures that the team is fully functional
- Shields team from external interference

Development team

- Members are assumed cross-functional
- Not always (usually?) possible
  - Programmers, UI designers, testers, business analysts, etc…
- Only title for members is "developer"
  - May have specialized skills, but accountability belongs to the team
- No sub-teams (testing, etc)
- "Optimal" size 3-9 members

Events

- Sprint
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

Sprint

- "Heart" of Scrum
- Usually one month or less
  - Consistent duration throughout development effort
- "Done," usable, and potentially-releasable product increment is created
- New sprint starts immediately after previous ends

- No changes are made
  - that impact the Goal(s)
- Team composition remains constant
- Quality attributes do not change
- Scope may be clarified/re-negotiated

Sprint planning meeting

- Work to be performed in upcoming sprint is planned
- Suggested eight hours for a one-month sprint
  - Shorter sprint, shorter meeting
- Two parts
  - What will be delivered?
  - How will the work get done?

Deliverables

- Product owner presents a list of items to complete
- Team develops a sprint goal, considering:
  - Product backlog
  - Latest product increment
  - Capacity of the team
  - Past performance

Sprint backlog

- Selected product backlog items, along with The Plan.
- Organized by team
- Estimate time to complete each item
- Assign tasks to individual members

Daily scrum meeting

- “Stand-up meeting”
- Fifteen minutes for team to “synchronize” and plan for the next 24 hours
- Held same time, same place
- Three questions
  - What was accomplished since the last meeting?
  - What will be done before the next meeting?
  - What obstacles are in the way?

Sprint review meeting

- Development team presents accomplishments
- Form of a demo of new features or progress
- Informal
- Entire team participates

Retrospective

- Opportunity to reflect and plan for improvements during the next sprint
- Process
  - Think about how the last sprint went
    - People, relationships, process, and tools
  - Identify major items that went well and potential improvements
  - Create a plan for implementing improvements

Retrospective questions

- What went well during the last sprint?
- What didn’t go well during the last sprint?
- How should the team improve for the next sprint?

Artifacts

- Project charter
- Product backlog
- Sprint backlog
- Burn down chart

Project charter

- Problem statement: short and succinct (one or two sentences)
- Project objectives: what the project will achieve
- Stakeholders: persons who will be actively involved with the project
  - Project sponsor, types of users, etc
- Deliverables: major results or services that will be produced
  - Specific things the software will do

Product backlog

- Ordered list of everything that might be needed in the product
  - Source of requirements
- Product owner is responsible for the backlog
  - Content, availability, ordering
- Never complete
- Lists all features, functions, requirements, enhancements, and fixes that need to be made

Format

Backlog Item	Estimate
As a guest, I want to make a reservation.	3
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As an admin, I want to change the availability of dates at my hotel.	8
As a developer, I want to improve exception handling.	8

- Backlog items are usually of the form:
  - As a _____, I want to _____ (so that I can _____).
- Product backlog items are sometimes called "user stories"

Sprint backlog

- Team selects items from the product backlog and commit to completing them
- Identifies tasks associated with each item and estimates hours to complete

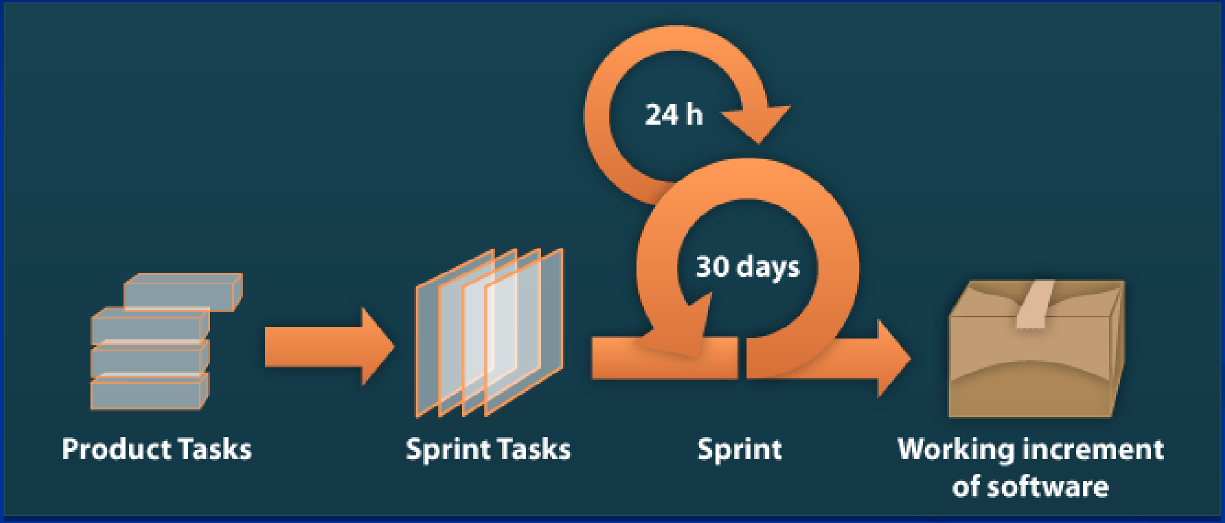
As a travel planner, I would like to see the reviews of each hotel.

- Program the Back-End (8 hours)
- Program the Front-End (4 hours)
- Write Test Cases (4 hours)
- Make Database Changes (2 hours)
- Update Dependent Pages (3 hours)

Sample sprint backlog

Backlog Tasks	Mon	Tue	Wed	Thur	Fri
Program the Back-End	3	4	1		
Program the Front-End			2	2	
Write Test Cases	3		1		
Make Database Changes		2			
Update Dependent Pages					3

Scrum


24 h
30 days
Product Tasks
Sprint Tasks
Sprint
Working increment of software

Criticisms

- Assumes you can estimate task duration
- Many meetings
  - Expensive
  - Disrupt flow
- Interchangeable cogs
- Regulatory environments?
- Difficult to collaborate with outside groups
  - How does marketing know what to advertise?

- Agile assumes development will be ongoing
  - If it doesn’t get done, it doesn’t get done
- Companies that write software to support traditional business models (e.g., making and selling goods) often have too many competing interests

Sprints

- We will have three sprints:
  - Fri, Feb 20 – Fri, Mar 3
  - Fri, Mar 3 – Fri, Apr 7
  - Fri, Apr 7 – Fri, Apr 28
- Sprint planning meeting should take from one to two hours
- “Daily” scrum meeting should be held 2-3 times per week
  - Same time and place

Sprint review meeting

- Held with your project coordinator on or before the following dates:
  - Friday, March 3
  - Friday, April 7
  - Last few days of the semester
    - See course website

Questions?