CSE 403

Software Engineering

Scrum and Teams

Today

- Project proposals
  - Proposal submission
  - Project registration
  - Preference submission
- Scrum
- Working in Teams

Project Proposals

Project proposals: submission

Proposal Submission on Canvas

- **Due Tuesday 04/08**
- https://canvas.uw.edu/courses/1798606/assignments/10278555

Project registration (Google form)

- **Due Tuesday 04/08**
- https://homes.cs.washington.edu/~rjust/courses/CSE403/project/project_registration.html

The intake form adds your project to a shared spreadsheet for others to review and follow up with questions.

Project proposals: review

Project review (Summary spreadsheet)

- **Wednesday — Friday**
- Read proposals, watch pitches, ask questions
- https://homes.cs.washington.edu/~rjust/courses/CSE403/project/project_summary.html

Project proposals: preferences

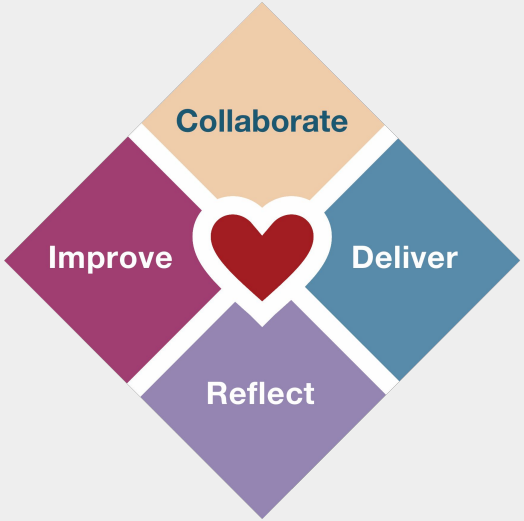
Project review (Summary spreadsheet)

- **Due Friday 04/11 at 2pm**
- Indicate preferences for at least 5 projects
- **Submit individual preferences OR group preferences**

- **Individual preferences**
  https://homes.cs.washington.edu/~rjust/courses/CSE403/project/project_preferences_individual.html

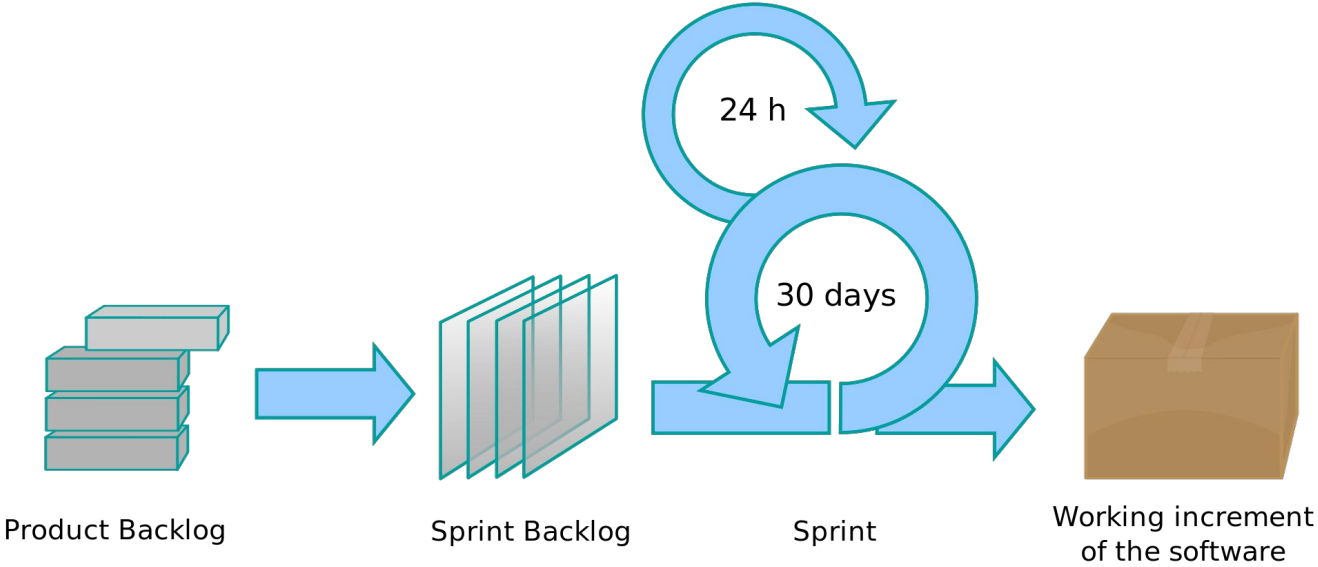
- **Group preferences**
  https://homes.cs.washington.edu/~rjust/courses/CSE403/project/project_preferences_group.html

Scrum



Heart of agile [Cockburn]

Scrum: overview



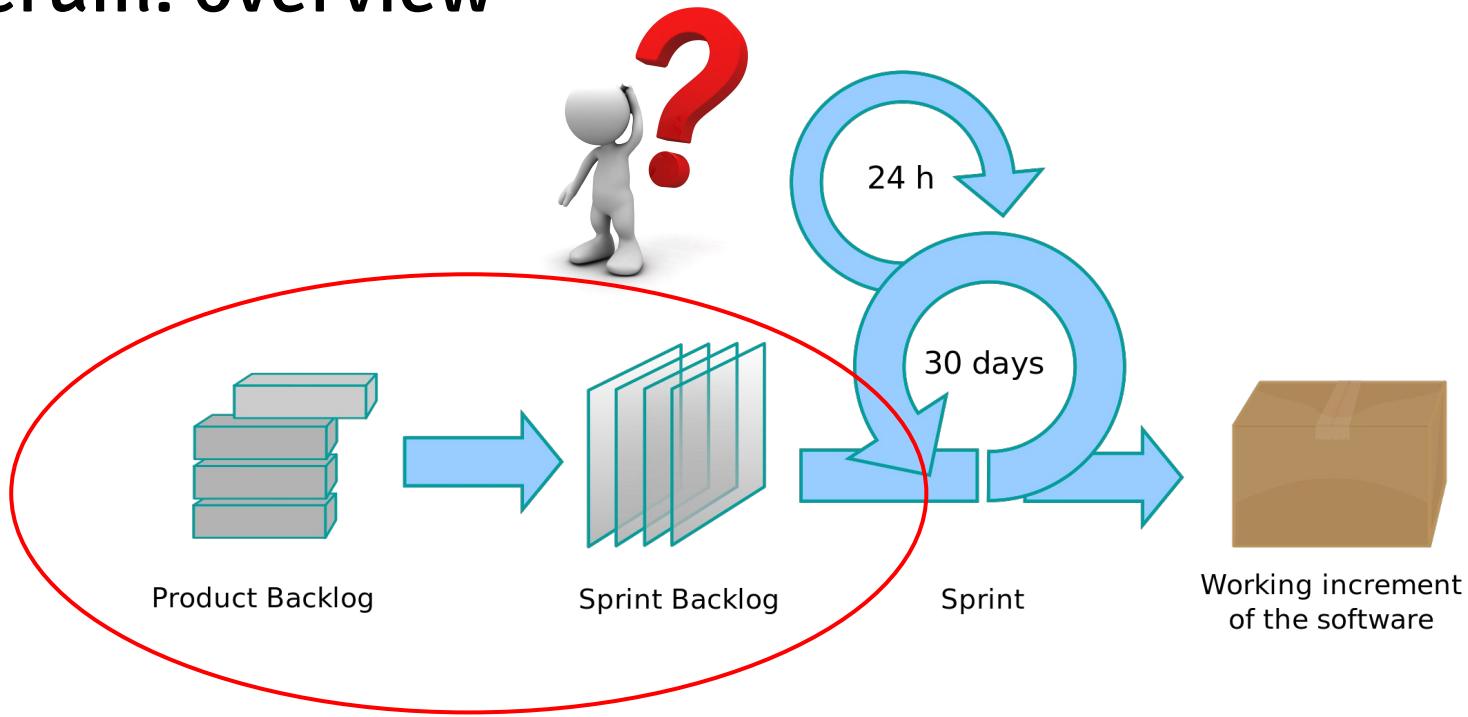
Scrum: overview

Small number of team members: 6 (+/- 2)

A time-boxed model:

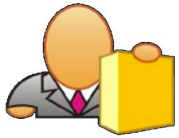
- Each Sprint (time box): max 30 days
- Fixed number of tasks for each Sprint
- Daily Scrum meeting: 15 min max
- Each sprint results in a
  - Sprint review (product demo): 0.5-1 hour
  - Sprint retrospective (post-mortem): 1-3 hours

Scrum: overview



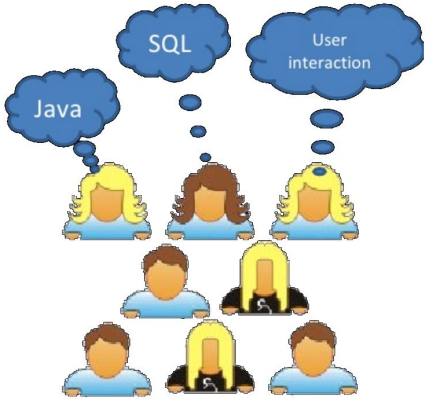
Prioritization: Must have vs. Should have vs. Could have vs. Won’t have

Scrum: roles



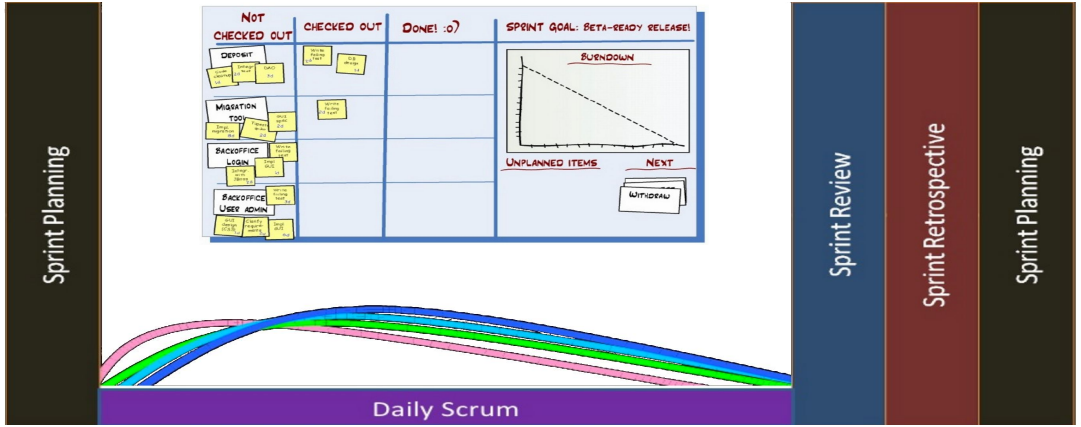
Product owner
(*Customer*)

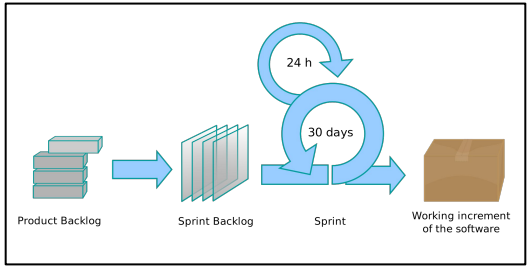
Scrum master
(*Manager/Moderator*)



Scrum team
(*Tech experts*)

Scrum: activities and planning





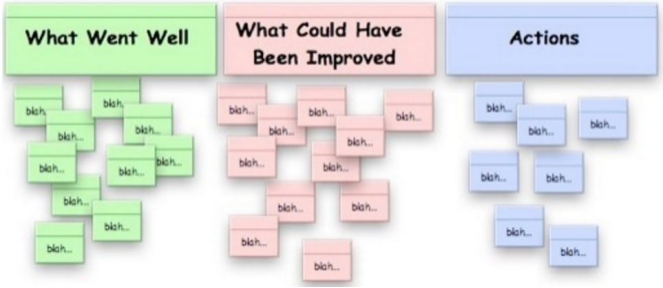
Daily scrum meeting (15min):

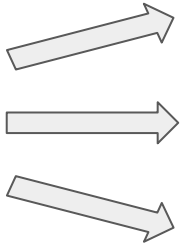
- What did I do since the last meeting?
- Any obstacles or blocking issues?
- What will I do until the next meeting?

Scrum: sprint retrospective

Who and what?

- Product owner, scrum master, and scrum team.
- Reflect, change, improve



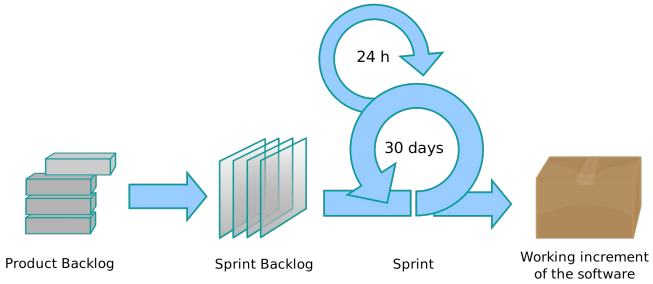


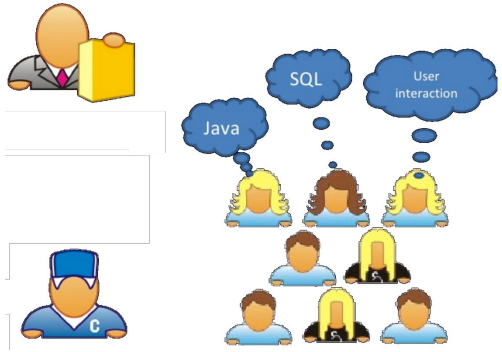
Stop doing

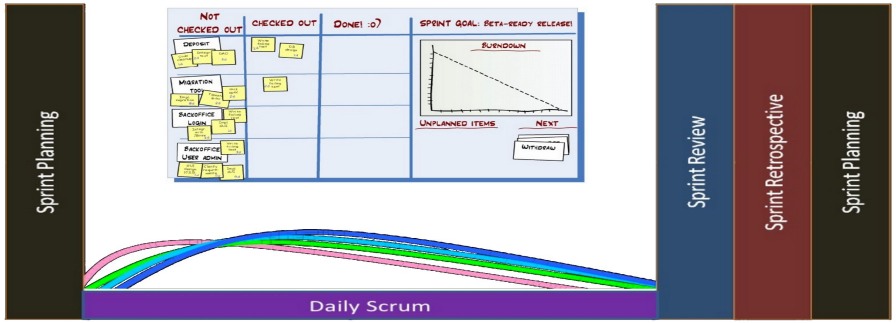
Continue doing

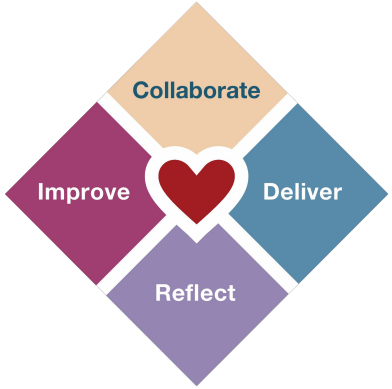
Start doing

Scrum: summary









Working in Teams

Seriously, working in teams can be great!

Benefits

- Attack bigger problems in a short period of time
- Utilize the collective experience of everyone

Risks

- Communication and coordination issues
- Lack of planning, reflection, improvement
- Conflict or mistrust between team members

Big questions

- **Communication:** How will everyone communicate?
- **Decisions:** How will your team make decisions?
- **Structure:** How do you **divide** your team **into subgroups**?

Big questions

- **Communication:** How will everyone communicate?
- **Decisions:** How will your team make decisions?
- **Structure:** How do you **divide** your team **into subgroups**?

Communication: powerful but maybe costly

- **Communication** requirements increase with increasing numbers of people (**everybody to everybody: quadratic cost**)
- Every attempt to communicate is a **chance to miscommunicate**
- **Not communicating will guarantee miscommunication**

Communication: example

"Hey *X*, I was wondering whether you finished the *Y* feature you were assigned?
Since we were late on some features last time, I thought I'd check.
When you have time, can you please tell me when *Y* is done. Thanks, *Z*."

What do you think about this email?

Communication: example

"Hey *X*, I was wondering whether you finished the *Y* feature you were assigned?
Since we were late on some features last time, I thought I'd check.
When you have time, can you please tell me when *Y* is done. Thanks, *Z*."

Be quantitative and specific:

- Use **specific, incremental goals**, not just things must be "done".
- List **specific dates** for when results are expected.
- **State requests** in a communication **explicitly**.
- **State an expected date/time** for a reply to a communication.
- **Remind** about upcoming **deadlines**, meetings, key points.

- **Don't be accusatory**; offer support and gratitude as appropriate.

Communication: example

"Hey *X*, I was wondering whether you finished the *Y* feature you were assigned? Since we were late on some features last time, I thought I'd check. When you have time, can you please tell me when *Y* is done. Thanks, *Z*."

A possibly better email:
"Hey *X*, how is your work on *Y* going?

It's due a week from Friday. Like we talked about at our last meeting, we are hoping to have the first 2 (out of 3) features designed by Sunday so we can review them together.

Please let me know by tomorrow night how much progress you made on *Y*. If you have any questions or need some help along the way, please let me know.

We'll all meet Saturday in person and you can give us another update at that time. Thanks, *Z*."

Big questions

- **Communication:** How will everyone communicate?
- **Decisions:** How will your team make decisions?
- **Structure:** How do you **divide** your team **into subgroups**?

Leadership and high-impact decisions

Who makes important product-wide decisions?

- One person?
- All by unanimous consent?
- Other options?

- Is this an **unspoken or** an **explicit** agreement?

Making decisions

- Document, Plan, Prioritize
  - Know what the real problem is!
- Delegate to subteams when possible
- Let everyone give their input (even if some is off-track)
- Write down pros/cons of alternatives
  - Evaluate cost/benefit/risks.
  - How long will it take? How much to learn? etc.
- Have a clear procedure for resolving disagreement
  - Strive for consensus, but if it cannot be achieved, ...
  - Majority vote and PM decides on a tie, etc.

Big questions

- **Communication:** How will everyone communicate?
- **Decisions:** How will your team make decisions?
- **Structure:** How do you **divide** your team **into subgroups**?

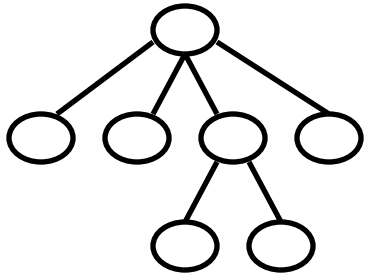
Common SW team responsibilities

The following could be all different team members, or some members could span multiple roles:

- Project management
- Functional management
- Designers/architects
- Developers: programmers, testers, integrators
- Lead developer (“tech lead”)

Key: Identify and stress roles and responsibilities

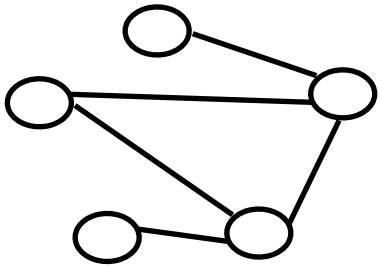
Team structure models: dominion



Dominion model

- Pros:
  - clear chain of responsibility
  - very familiar model
- Cons:
  - single point of failure at the top
  - little or no sense of ownership by everyone

Team structure models: communion



Communion model

- Pros:
  - a community of leaders, each in their own domain
  - inherent sense of ownership
- Cons:
  - miscommunication, competing visions, dropped responsibilities
  - many points of partial failure