

Software Engineering Project Workshop (SENG202)

Matthias Galster

Reflection on Phase 1

August 17, 2020

Shared space for this session

- Google Docs
 - <https://docs.google.com/presentation/d/1PX2rvmsh184ugiB48qiMP6yfKW09R8fpkODjsaH5B7k/edit?usp=sharing>
- Link also on Learn
 - COVID-19 section under “Schedule changes” for 17 August
- Everybody can edit
 - No need to log in

Reminders (1)

- Time left for Phase 2

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
August 17	August 18	August 19	August 20	August 21	August 22	August 23
August 24	August 25	August 26	August 27	August 28	August 29	August 30
August 31	September 1	September 2	September 3	September 4	September 5	September 6
September 7	September 8	September 9	September 10	September 11	September 12	September 13
September 14	September 15	September 16	September 17	September 18	September 19	September 20
September 21						

Reminders (2)

- Submit your weekly individual reflection (Mondays, 5:00pm)
 - Updated form
- Keep logging as you go
 - Follow instructions
- Labs
 - Online
 - Same allocation of teams as last week
 - Tutorial session and quiz: CI; stand-ups and feedback sessions
- COVID-19 updates shared via Learn page

Exercise to improve communication in meetings

Task 1: What do I know about communication in meetings?

Task 2: How do I perceive examples + tutorials on meeting communication?

Task 3: How do others perceive videos? Can I learn from them?

Task 4: How does my team communicate in our meetings?

Delay due to
COVID-19

Task 5: What do I think about our meetings as an “outsider”?

Task 6: What do my team mates think about our meetings?

Task 7: Did my understanding of communication in meetings change?

Exercise over the next 5 weeks¹ – updated

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
July 20	July 21	July 22	July 23	July 24	July 25	July 26
[Task 1] Assess initial knowledge (questionnaire)²						
[Task 2] Watch and comment on brief tutorial and example videos on AVW platform (reflection 1)²						
July 27	July 28	July 29	July 30	July 31	August 1	August 2
[Task 3] Review and rate comments on videos made by your class mates on AVW platform (reflection 2)²						
August 3	August 4	August 5	August 6	August 7	August 8	August 9
				[Task 4] Video-record a team meeting		
August 10	August 11	August 12	August 13	August 14	August 15	August 16
[Task 4] Video-record a team meeting (all team members; teaching staff will provide support)						
August 17	August 18	August 19	August 20	August 21	August 22	August 23
[Task 4] Video-record a team meeting (all team members; teaching staff will provide support)²						
[Task 5] Watch and comment on team meeting recording on AVW platform (for own team only)² – by August 19, 5:00pm						
[Task 6] Review and rate comments on meeting recording made by team mates on AVW platform (reflection 3)² – August 20-21						
[Task 7] Assess knowledge (questionnaire)² – August 20-21						

¹ Course exercise, but with your consent we will use anonymized data to study learning experience (consent form will be available at the beginning of the exercise); if no consent: still same activities, but data won't be used to study learning experience

² Individual activity in your own time, but log as #class activity

Agenda

1. Presentations – style
2. Presentations – content
3. Peer feedback and logging
4. Some stats
5. Retrospective

Agenda

1. Presentations – style
2. Presentations – content
3. Peer feedback and logging
4. Some stats
5. Retrospective

“I” versus “we”

- Team owns project, not individuals
 - What individuals did is irrelevant – no individual reports
 - Avoid overlaps and redundancies, coordinate
- Check that materials are all at roughly the same level of detail
 - Avoid deep dives by one team member into very specific areas
 - Avoid impression of disparate individuals, bound together by circumstance
- Stay involved and engaged (when not speaking)

*When 'i' is
replaced By 'we'*

*Even
'illness'
Becomes
'Wellness'*

Flow

- Ensure consistency between team members
 - Logical flow versus who did what?
 - Avoid impression that team members have prepared different parts
- Logical flow, e.g.,
 - Requirements before UC?
 - “based on UCs/ATs we created UI prototypes”?
- May need to adapt
 - Timing issues
 - Off-script sidetracks, presenter stuck or lost
- Slide numbers help audience



Visual aids

- Whenever possible: visuals rather than text
 - Use other visual aids as needed, e.g., whiteboard
- When using slides
 - Don't overload slides, use readable colors
 - Ensure readability of slides, figures, check for typos
- Relate slides to what is currently said
- Don't block screen



Timing

- Evenly distribute speaking time
- Remember
 - If one team member goes overtime: impacts others
- When running out of time
 - Don't stick to original plan but improvise
- Run your own timer
 - E.g., on a tablet put in front of the team



Preparing and rehearsing

- Prepare and rehearse, but don't over-rehearse
 - Should not sound like being read, but rather as a “conversation”
- Avoid content free statements
 - Don't talk about the structure but structure the talk
 - Avoid “ammmmmm”, “... and stuff...”, etc.
- Test setup, slides, etc. before the talk



Speaking

- Interaction among team members could add excitement
 - Team standing in a line, talking in turn, can be quite static and dry
 - Consider changing speakers more than once (but avoid “traffic”)
- Notes (on paper, device) are good, but don’t read too much
 - At this point you should be the experts on your projects
- Avoid talking to the wall
- Avoid talking too fast or too slow
- Speak up



Answering questions

- Involve all team members
- Avoid vague answers
 - Lack of shared understanding?
- Ensure coherent answers
 - Disagreements or no shared understanding?



Agenda

1. Presentations – style
2. **Presentations – content**
3. Peer feedback and logging
4. Some stats
5. Retrospective

General

- Most points apply to design activities and design document
- Consider target audience (what they know and want to know)
- No need to explain basics
 - Structure of templates, technical constraints (e.g., app will be in Java)
 - What are patterns
 - Why do we prioritize
 - Why do we analyse risks, why is there a project plan
 - Etc.
- No need to present (technical) details that are likely to change

Context

- Scope
 - Unclear target audience
 - Too many features – avoid “feature creep”
 - May fail to implement them all
 - May compromise quality: usability (and other qualities) if too many features
- Be realistic rather than “hypothetical”
 - What will be done, not what could be done
 - Avoid “imaginary” features

Stakeholders and requirements

- Stakeholders
 - Remember what stakeholders are (users only one stakeholder category)
 - Often quite diverse stakeholders, possible users; can all be satisfied?
- Requirements
 - Avoid vague requirements

Quality requirements and key drivers

- Quality requirements
 - Quality requirements are about product (rather than process/project)
 - “Product is ready”, “analytics”, “coding standards” etc. are not QR
 - “no unknown errors”, “app must be stable and not crash”, etc. unhelpful
 - How to measure quality requirements
 - Scoring too fine-grained (1-10)
- Key drivers
 - What do they mean
 - How do weights in key driver analysis map to stakeholder priorities
 - How are key drivers related to quality requirements
 - How to measure key drivers

Use cases, GUI, deployment

- Use cases
 - Relationships between use cases, to context, requirements unclear
 - Some use cases were too low level and technical (e.g., “user can filter”)
- GUI
 - Some explained in too much detail (buttons, colors of labels, etc.)
 - Not about demonstrating the final product
 - Might be tempting to replicate prototype in SceneBuilder
 - Would rather see basic features with robust AT and UT environment
 - GUI elements added as needed – focus on quality-assured functionality
- Deployment models
 - Missing SW/HW artefacts

Acceptance tests

- No need to walk through all acceptance tests in detail
- No need to repeat details about functionality
 - Instead: how is it tested?
 - But: Missing details for individual tests, how would one test them?
- Responsibilities: roles rather than developers
- Future issues (heads-up)
 - Issue 1: AT in template form hard to develop further into automated AT
 - Don't want to transform, e.g., to get user feedback
 - Can't transform, e.g., due to GUI elements entangled with scenario steps
 - Issue 2: What is truly an AT (Cucumber), what UT (JUnit)

Class diagrams and design

- Too many details, e.g., what a method will do, visibility
- Patterns other than MVC?
- Why include (many) view classes?
- How fragile is the design when adding features
 - Will impact on GUI, classes, test structures, etc. be minor or major?

Risks and project plans

- Many of the risks were actually about developers
 - System and/or its environment often ignored
 - Has your group experienced any unforeseen risks already?
- Assigning and using priorities, weights
 - How are we to know that these are realistic values and not “magic”?
- Some project plans seem unrealistic

Agenda

1. Presentations – style
2. Presentations – content
3. Peer feedback and logging
4. Some stats
5. Retrospective

Comments on peer feedback

- Peer feedback tends to be “optimistic”
 - Peers do not have full picture of project
- Keep it professional
 - Avoid aggressive language
- Good: some feedback offered personal advice
- If doubts about how to act on or how to give feedback, ask us

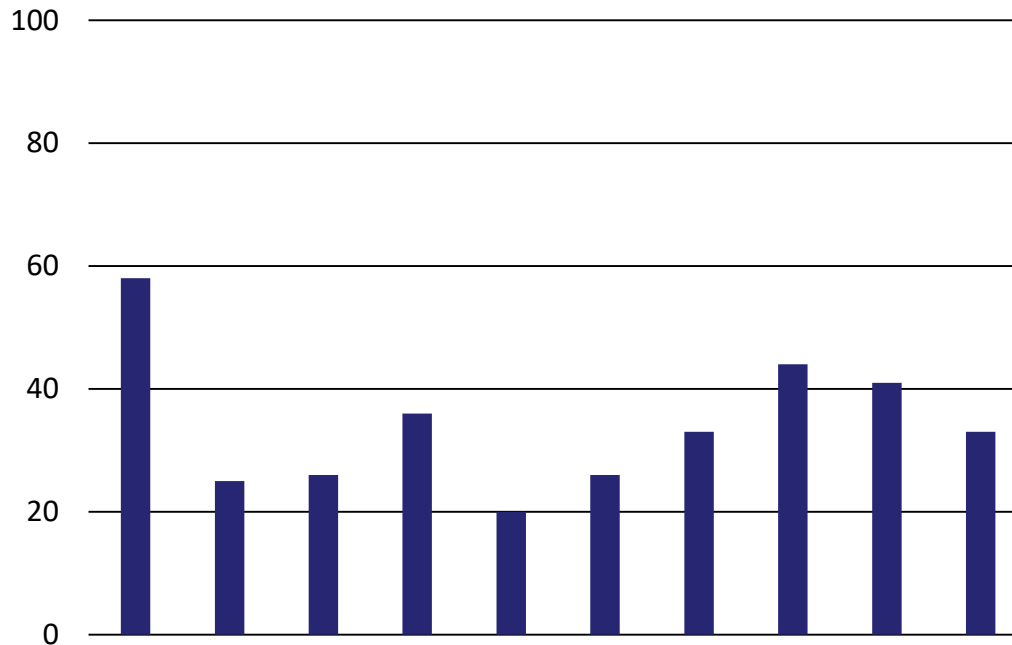
Comments on logging

- Log entries need to have meaningful descriptions
- Your team is your project
- Log as you go and in small sessions
- Each description must include one (and only one) tag
 - Cannot use your own tags
 - 41 hours untagged
 - 35 logs untagged
 - 95 logs multiple tags

Agenda

1. Presentations – style
2. Presentations – content
3. Peer feedback and logging
4. Some stats
5. Retrospective

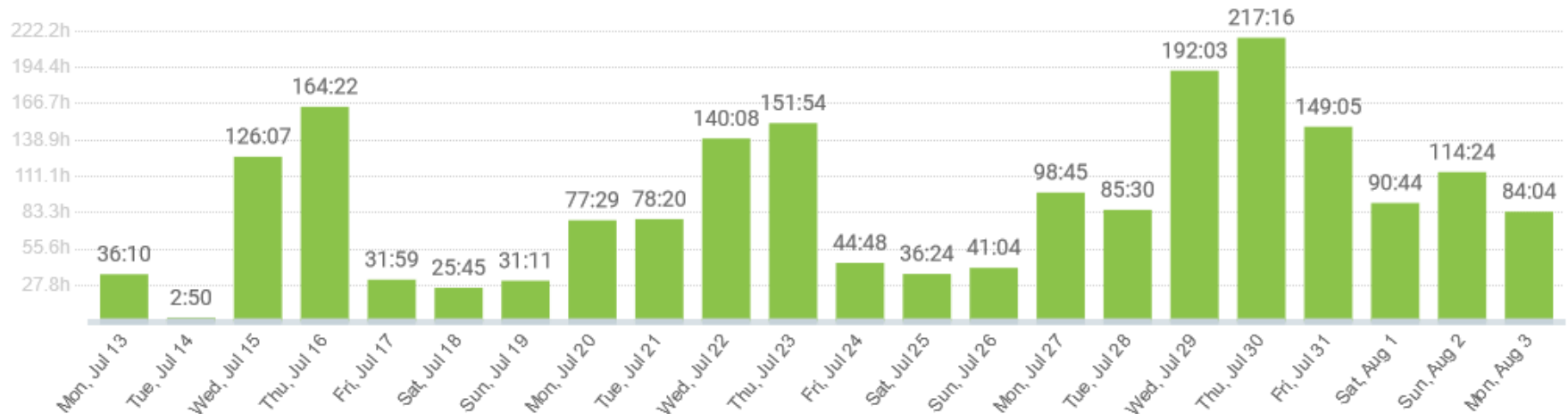
Design document (number of pages)



- Statistics
 - Max: 58
 - Min: 20
 - Average: 34
 - Stdev: 11

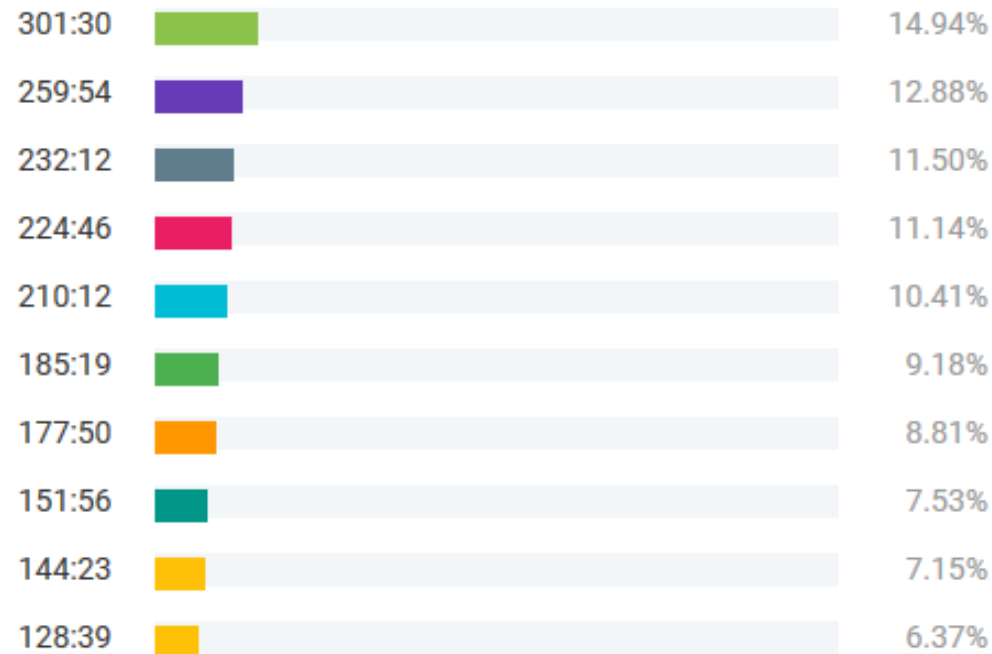
Not necessarily
correlated with quality

When was time spent



Same pattern for your own team?

Hours – teams (total)



Not necessarily
correlated with quality

Hours – individuals



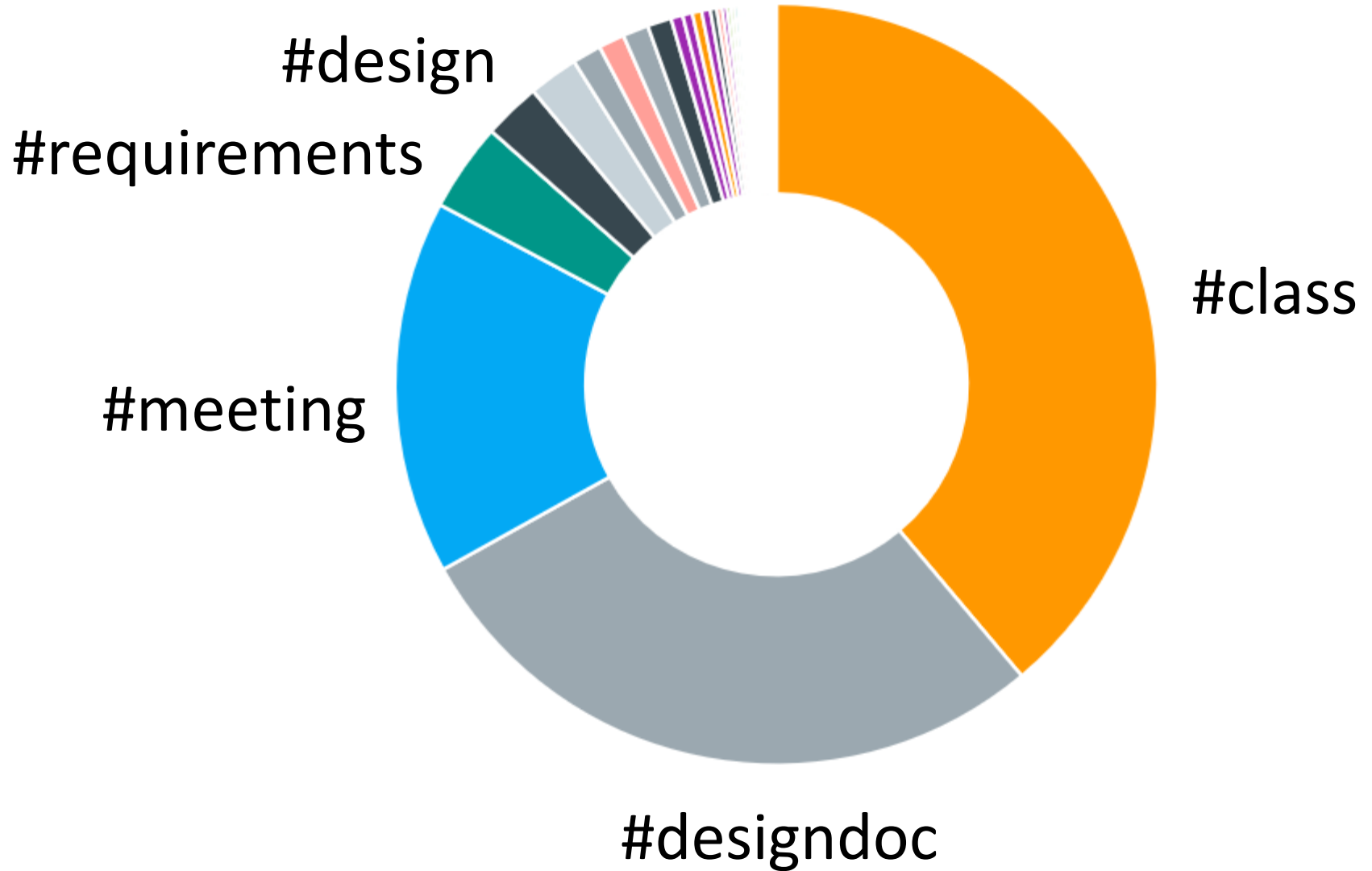
- Statistics
 - Max: ~74
 - Min: ~18
 - Average: 33
 - “Expected” (?)

Hours – top-10 versus bottom-10

73:42		17:49
60:45		22:02
58:50		22:39
54:58	Top-10	22:55
54:54		23:37
47:30	Bottom-10	24:00
45:50		25:29
45:19		25:30
44:55		25:35
44:05		25:51

Includes #class

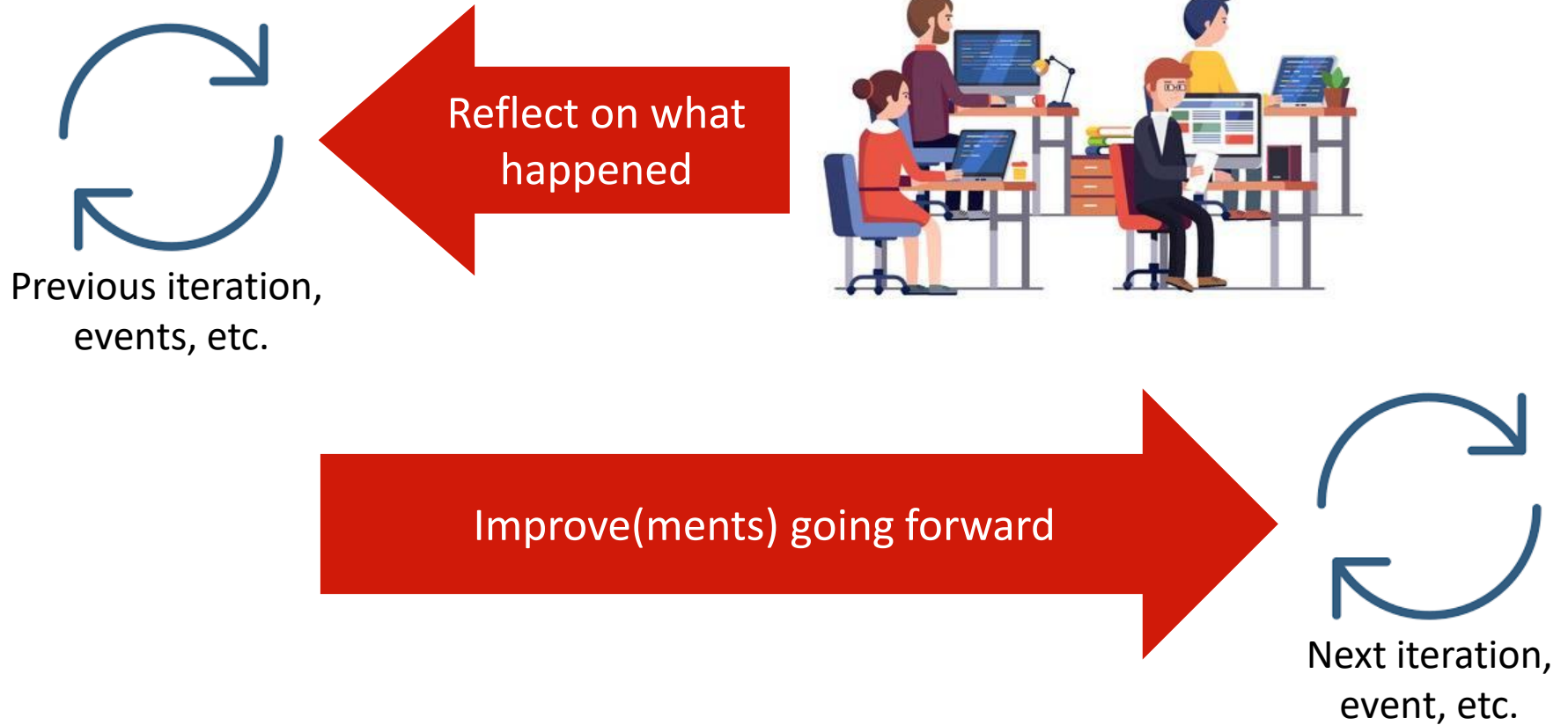
Where did time go



Agenda

1. Presentations – style
2. Presentations – content
3. Peer feedback and logging
4. Some stats
5. Retrospective

Retrospective



Liked – Learned – Lacked – Longed for

Liked

Things you liked, went well, to be repeated



Learned

Things you have learned, new ideas to try



Longed for

Something you desired or wished for







Lacked

Things you disliked, to be changed*

Reflect not only on team, but also on yourself

- How do you feel about
 - Stand-ups, presentation, reflection, feedback
 - Working with others, sticking to commitments
 - Making deadlines, showing up on time, putting in expected effort
 - Tools, fixing technical issues
 - Doing own research to find solution to problems
 - Coping with a vague problem and project description
 - Etc.
- Impact
 - On you, your team, other courses, commitments, life
 - Consider that every project, team, etc. is different
 - Is SENG for you?

Agenda

1. Presentations – style 
2. Presentations – content 
3. Peer feedback and logging 
4. Some stats 
5. Retrospective 