

Teamwork Guidelines

17-423/723 Designing Large-Scale Software Systems

Recitation 2
Jan 24, 2025

Credit

Some slides adapted from [materials](#) by Christian Kaestner for 11-695

Teams are Inevitable

Projects too large to build for a single person (division of work)

Projects too large to fully comprehend by a single person (divide and conquer)

Projects need too many skills for a single person to master (division of expertise)

Importance of Teamwork Skills

Virtually all software projects are done in teams

Good teams make it fun to work together

Learn from each other

Limited influence in selecting/firing team members in most organizations

Peer performance evaluations common in industry (e.g. [Google's Process](#))

Who has had bad experiences in teams? Student teams?
Teams in industry?



Example Complaints

"M. was very pleasant and would contribute while in meetings. Outside of them, he did not complete the work he said he would and did not reach out to provide an update that he was unable to. When asked, on the night the assignment was due, he completed a portion of the task he said he would after I had completed the rest of it."

"Procrastinated with the work till the last minute - otherwise ok."

"He is not doing his work on time. And didnt check his own responsibilities. Left work undone for the next time."

"D. failed to catch the latest 2 meetings. Along the commit history, he merely committed 4 and the 3 earliest commits are some setups. And the latest one commits is to add his name on the meeting log, for which we almost finished when he joined."

"Unprepared with his deliverables, very unresponsive on WhatsApp recently, and just overall being a bad team player."

"Consistently failed to meet deadlines. Communication improved over the course of the milestone but needed repeated prompts to get things done. Did not ask for help despite multiple offers."

Common Frustrations

No visible progress until last minute

Late work

Incomplete or low quality solutions at integration

Unresponsive team members

Passive, uninterested team members without initiative

Needs lots of reminding and help

Sources of Problems

Priority differences ("10-601 is killing me, I need to work on that first", "I have dance class tonight")

Ambition differences ("a B- is enough for graduating")

Ability differences ("incompetent" students on teams)

Working style differences (deadline driven vs planner)

Communication preferences differences (avoid distraction vs always on)

In-team competition around grades (outdoing each other, adversarial peer grading)

Discussion: How would you handle/prevent?

Scenario 1: One team member has very little technical experience and is struggling with basic Python scripts and the Unix shell. It is faster for other team members to take over the task rather than helping them.

Scenario 2: You divide the work but when you try to integrate on the evening before the deadline you learn that some team members have failed to complete their part. You tried the day before, but got stuck with a dependency problem.

Scenario 3: This homework is low priority for one team member. They rarely contribute beyond the bare minimum at the last minute. (The rest of the team grudgingly compensates and achieves full points for the assignment. You do not feel comfortable criticizing the student as it may negatively affect their grade.)

Teamwork Policy in this Course

Teams can set their own priorities and policies – do what works for you & experiment

Not everybody will contribute equally to every assignment – that's okay!

Team members have different strength and weaknesses – that's good!

We will intervene in team citizenship issues

Golden rule: Try to do what you agreed to do by the time you agreed to. If you cannot, seek help and communicate clearly and early.

Team Citizenship

- Be responsive and responsible
- Come to meetings on time, participate actively
- Stick to commitments, work on assigned tasks
- When problems arise, reach out, replan, communicate early, be proactive
- (Replanning and dealing with mistakes is normal)

We will adjust grades based on complains about:

- Lack of communication
- Disrespectful or dismissive communication
- Not attending team meetings (without excuse)
- Blowing internal deadlines without communication
- Failing to complete agreed tasks without timely communication

Teamwork Tips

Establish Communication and Meeting Patterns

Agree on how to communicate in the team: Email? Slack? Whatsapp?

Agree on communication expectation. Different people have different habits and expectations. Be explicit!

- Read emails daily? On weekends?
- Respond to urgent chat messages within 3h? Read old chat messages?
- Be available for chat during certain hours?

Find meeting times. Plan ahead or meeting as needed?

Set intermediate internal deadline for integration

Set realistic expectations: All have other classes and distractions; communicate availability openly

Communication Tips

Focus full-group meetings on planning and reflection, meet in smaller groups for focused work

Use Slack/chat deliberately

- Consider chat ephemeral, don't expect everybody to catch up on all old messages
- Separate social communication from work comm., urgent from not urgent
- Explicitly tag people if you need their input, enable notifications during "working hours"
- Discuss non-urgent, long-term things outside of chat associated with topic (issue tracker, Google doc, ...)

Reserve time for socializing and celebrating success

Share the Work!

Team members have different strength and weaknesses – that's good

Make use of individual strength of team members (split, pair up, help, ...)

Usually somebody will take responsibility for team management tasks (e.g., schedule meetings, moderate, meeting notes, track work, reminders, check submission) or reporting

- Team management is work too!
- Consider rotating

Maintain Accountability

Write down explicit deliverables: Who does what by when

- Be explicit about expected results, should be verifiable
- Track completion, check off when done
- GitHub issues, Jira, Trello board, Miro, Google docs, Slack, ... – single source of truth, with history tracking (see next recitation)

Complete deliverable list during meeting: Everybody writes their own deliverables, others read all deliverables to check understanding

- If not completed during meeting or team member not at meeting, email assignment after meeting to everybody; no objection within 24h counts as agreement with task assignment

Recall: Common Sources of Conflict

Different team members have different working patterns and communication preferences

- e.g., start early vs close to deadline; plan ahead vs try and error
- discuss and set explicit expectations; talk about conflicts

Different abilities, unexpected difficulties

- work in pairs, plan time for rework and integration
- replan, contribute to teams in different ways
- work around it, it's the team's responsibility

Unreliable team members, poor team citizenship

- e.g., not starting the work in agreed time, not responding, not attending
- have written clear deliverables with deadlines

Best Practices For Team-Based Software Development

Development with Git

Each team will be assigned a Github repository to store & develop code over the project milestones.

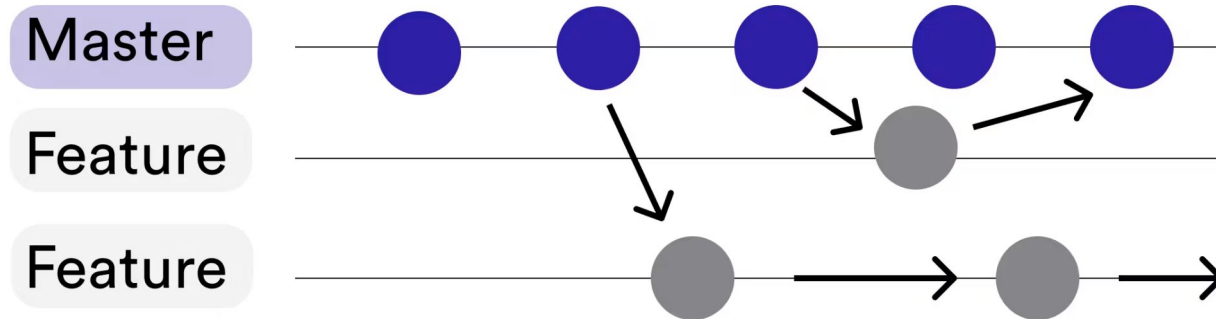
For reviewing the basics on Git: [Pro Git book](#), first 3 chapters

Use Feature Branches in Git!

Context: **Small & coherent commits** make it easier to organize code changes.

Problem: When every team member pushes their commits to the same branch **incomplete features** and frequent **merge conflicts** can create confusion.

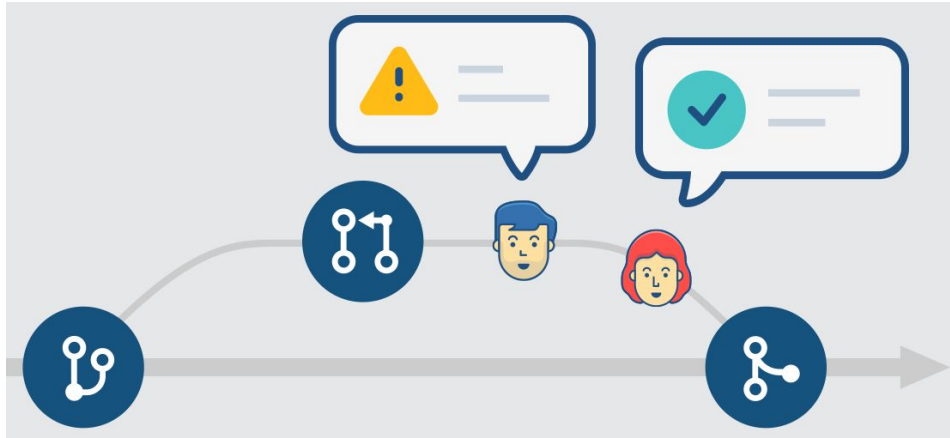
Solution: Every feature gets its own branch that gets merged into the main branch when it adds a stable increment to the project.



Practice Code Reviews via Pull Requests!

Problem: Code only seen by one developer tends to (1) have **lower code quality**, (2) be **harder to change** since only one developer understands it

Solution: Before merging pull requests, add one or more team members as reviewers to let them check whether your changes are **understandable**, **bugs-free**, and are **consistent** with other parts of the code



Bad Habits You Should Avoid In Team Projects

Never force-push (`git push -f`)

- Someone might have pushed changes in the meantime
- Always merge changes first



Bad Habits You Should Avoid In Team Projects

Avoid commit Messages such as “**small changes**”, “**bugfix**”, or “**coding**”

- Commit messages are intended to help yourself and your team members understand what and why changes were made

	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Bad Habits You Should Avoid In Team Projects


Avoid huge pull request


- Nobody wants to go through a large list of changes that are partially un-related
- Better: Keep PRs small and simple.

“There are two kinds of PRs: The ones that obviously don’t introduce any bugs, and the ones that don’t introduce any obvious bug”

Comparing changes

Choose two branches to see what’s changed or to start a new pull request. If you need to, you can also [compare across for](#)

 base: master ◯ ← compare: develop ◯ ✓ Able to merge. These branches can be automatically merged.

 **Create pull request** Discuss and review the changes in this comparison with others.

◯ 19 commits **139 files changed** ◯ 0 commit comments



Summary

Teamwork is unavoidable, teams rarely fully self-selected, good teams are fun

Teamwork is hard, skills to be learned

Set explicit expectations for communication and work allocation

Talk to the course staff if you have any questions or concerns!

Further Reading

- Mantle, Mickey W., and Ron Lichty. Managing the unmanageable: Rules, tools, and insights for managing software people and teams. Addison-Wesley Professional, 2019.
- DeMarco, Tom, and Tim Lister. Peopleware: Productive projects and teams. Addison-Wesley, 2013.
- Brooks Jr, Frederick P. The mythical man-month: Essays on software engineering. Pearson Education, 1995.
- Classic work on team dysfunctions: Lencioni, Patrick. "The five dysfunctions of a team: A Leadership Fable." Jossey-Bass (2002).
- Oakley, Barbara, Richard M. Felder, Rebecca Brent, and Imad Elhajj. "[Turning student groups into effective teams](#)." Journal of student centered learning 2, no. 1 (2004): 9-34.