

Team Management

CSCI2340: Software Engineering of Large Systems Steven P. Reiss





SMARTBEAR

CSCI2340 - Lecture 6

Continuing Preparation for Programming

Working in teams

- Collaborative software development
- Workflows and actions
- Cost Estimation
- Test-Driven Development



Working in Teams

- Required for large projects
 - Too big to do on ones own
 - Multiple people can work in parallel
- Requires Effort
 - Coordination, organization, management
- Difficult to make productive
 - Mythical Man Month (Fred Brooks)
- Tools can help
 - Software process programming
 - Modern team management tools

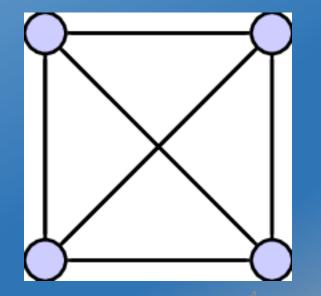
THE PROJECT MANAGEMENT GUIDE FOR REMOTE TEAMS



Team Size Matters

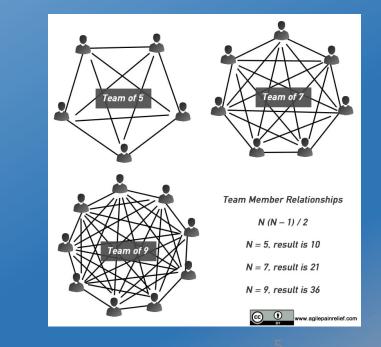
- Small teams (<= 4)
 - Can get everyone together
 - Number of communication paths is small
 - Meetings can be productive, even if not organized
 - Easier to move people around
 - People can learn the whole system
 - Easy to divide project into logical pieces
 - Front end, back end, database, business logic, ...
 - People can work independently





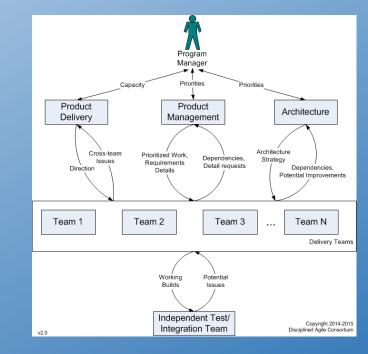
Team Size Matters

- Moderate Teams (~6-12)
 - Assume at least one person will be absent from any meeting
 - Assume at least one person will flake out (get sick, ...)
 - Can't communicate with everyone
 - Meetings take longer & get less done
 - Need better coordination
 - Project needs to be managed
 - More difficult to divide into independent pieces
 - More critical paths (things that can go wrong)
 - Difficult to understand the whole system



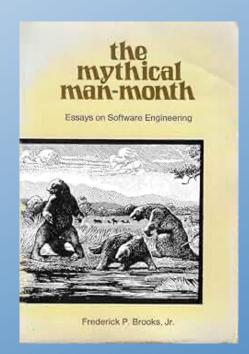
Team Size Matters

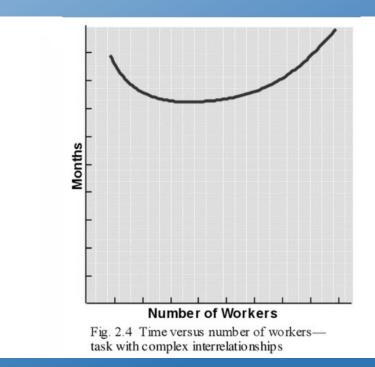
- Large Teams (15+)
 - Require a hierarchy
 - Target system generally quite large
 - No one knows all the details
 - Most members don't know the complete system
 - Concentrate on your piece and how it fits in, not everything else
 - Different strategies are used
 - These are out of favor except for very large systems
 - Divide the system into separable components
 - Build these independently using smaller teams
 - With common libraries or frameworks (separate teams)



Mythical Man-Month

- Adding people to a project can delay it further
 - Can't measure effort in terms of man-months





Team Management

- Software teams need to be managed
 - Ensure equality of workload
 - Ensure consistency of decisions
- Democracy doesn't work
 - Someone must make hard decisions
 - Someone must ensure consistency
 - Someone must make sure everyone is pulling their weight
 - People aren't getting in over their head
 - People work on things they are best at



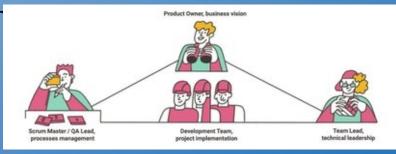
Team Leader

- Need a team leader or manager
 - Break project into tasks
 - Assign people to tasks
 - Keep track of what everyone is doing
 - Coordinate where needed
 - Reassign people as needed
 - Make critical decisions
 - Ensure consistency
 - In charge of presentations



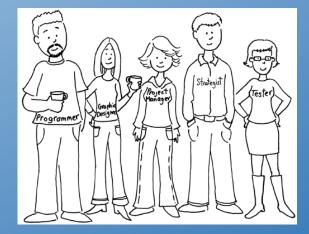
Team Management: Other Roles

- Assistant Manager or Leader
 - Manager not available 24/7, might get ill, might need help
- Product Managers
 - In charge of a portion of the software (e.g., front end or back end)
- Documentation Manager
 - Organize files, documents, requirements, repository, versions & branch
 - This is what I see to grade you on
- Quality Assurance (QA, Testing) Manager
 - Create system test cases, supervise testing, ...
 - Approve code for release
- User Interface Guru
 - Ensure a consistent, easy-to-use, nice-to-look at user interface
 - Designing look and feel, icons, logos, etc.
- Provisionary
 - Handles setting up or provisioning AWS, VMs, Containers, databases, ...



Team Management: Other Roles

- Security, Privacy and Ethics Czar
 - Ensure system is secure & fair
 - Security testing, fairness testing
 - Set privacy policy, enforce privacy policy
- Performance Analyst
 - Determine where and when there are performance problems
 - Performance testing
- Scribe
 - Take notes at meetings; ensure information is current
- Domain Experts & Users
 - Provide essential information about the problem being solved
- Skeptic
 - Question everything. Avoid risks and failure.



PROJECT

- Let's have a short project meeting
 - Discuss roles
 - Tentative role assignments
 - Check status
 - What should be done by whom this week
 - Requirements, specifications and software architecture
 - 10 minutes
- Then we'll get into tools & other topics

Team Communication

- Regular communication is key to a successful team
 - Knowing how your part fits with others
 - Getting things done in a time fashion
 - Negotiating who does what
 - Negotiating interfaces between components
- Physical meetings can be difficult
 - Team members might be distributed around the world
 - People have other commitments
- Tools for communication
 - SLACK
 - Discord
 - JIRA
 - GITHUB (wiki, issues)



Slack Alternatives for Team Communication and Collaboration

GitHub Communication Tools

• GitHub Wikis

- Good for documentation, notes, comments
 - Internal as well as external documentation
- Let others provide feedback

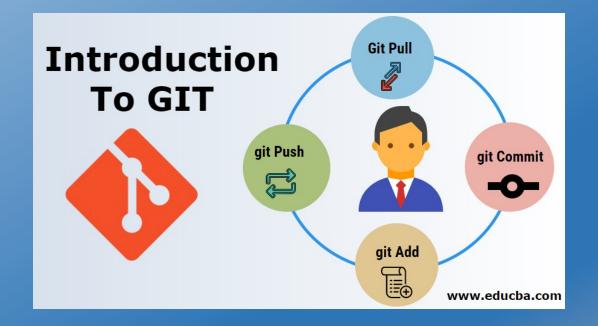
GitHub Issues

- Items in repo to plan, discuss and track work
- Bug tracking
- Combines slack-like messaging with project planning

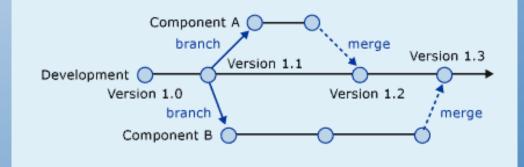
📮 angular / angular	⊙ Watch ▾ 3.2k	☆ Star 70.5k	양 Fork 18.5k
↔ Code ① Issues 2.5k 1 Pull requests 292 ⊙ Actions	Projects 6	🛈 Security 🛛 🗠 Ir	nsights
Filters - Q ississue issopen	C Labels 204	⇔ Milestones 10	New issue
① 2,452 Open ✓ 19,744 Closed			
Author • Label • Projects • Milestones • Assignee •	Sort -		
① Create a "Directives Overview"or similar topic P3 comp: docs #40755 opened 2 hours ago by aikidave 中 Backlog	feature		۲
Make FormArray's inability to be treated as a regular Array mo docs comp: docs comp: docs/api comp: forms #40753 opened 6 hours ago by ychalkin 中 needstriage	ore prominent in th	le	
① bug(cdk-drag-drop): :enter / :leave animations are triggered in comp: animations #40752 opened 9 hours ago by udos86 ♀ needsTriage	ncorrectly		D 1
The structural directive type check docs section is incomplete #40739 opened 2 days ago by ArielGueta reedsTriage	comp: docs		₽ 8
Provide usage examples for APP_INITIALIZER token comprodoes #40730 opened 3 days ago by AndrewKushnir 中 needsTriage	hotlist: community-h	alb	🤹 🖓 2
More precise type for APP_INITIALIZER token comp: core cross flag: breaking change hotilst: community-help #40729 opened 3 days ago by AndrewKushnir ⇔ Backlog	s-cutting: types feature		🍩 🖓 2

Version Management: GIT

- Designed for modern software development
 - Flexible repositories
 - Ease of creating branches
- Designed for agile development
 - Good for all development



Software Branching



- Individuals (or smaller teams) work on an extension or feature
 - Interim system might not be usable
 - Extension might not work or be desirable at the end
 - Multiple such extensions developed at once
- Want to allow these teams to work productively
 - Without interfering with one another
- This can be accomplished with branches
 - Different, independent versions of the software
 - Developers (or small teams) work on their own branch
- But branches aren't necessary
 - Can develop a substitute method or class
 - And make using that class conditioned on a flag or environment variable
 - A bit more work as the status quo has to be maintained

Avoiding Branches

- Branches can create confusion
 - Coding for the past, not the future
- Eventually branches need to be merged
 - With each other
 - With the main system
- Problems arise when there are conflicting changes
 - Ideally these are avoided
 - Selecting features to avoid conflicts
 - In practice, they are common
 - Branching makes these worse
- Need a team strategy for handling merge problems
 - Team manager, negotiation, code ownership, ...



Continuous Integration

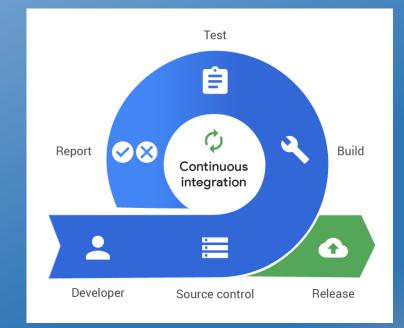
- Always have a working version of the system
 - Minimize branching for everyday development
 - Merge all code changes regularly (daily/weekly/...)
 - Build and test the system on merge
 - Automate this process as much as possible

• Pros

- Simplifies merging
- Helps find bugs faster (continuous testing)
- Improves productivity
- Supports dogfooding

• Cons

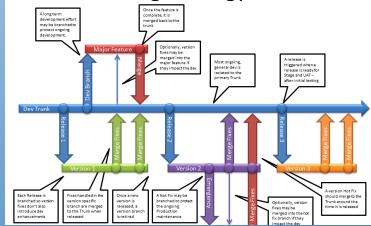
- Requires development that can be merged
- What to do when tests fail



Branching and Maintenance

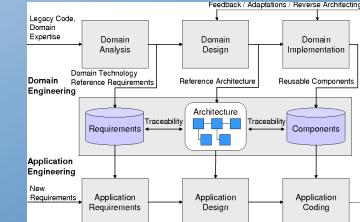
- Continuous integration obviates branching
 - Everything done in the main branch
 - Everyone works in the main branch
 - Generally fewer and less severe merge conflicts
 - Requires more thought while coding
- Branching might still be needed for maintenance
 - Current stable version of the system for users
 - Current development version of the system
 - Previous user versions of the system
- Security patches need to be made in all of these





Families of Software Systems

- Set of applications with common set of features
 - All developed at once as one system
 - System can be configured in multiple ways
- Might be related versions of the same system
 - State & Federal tax programs
 - IntelliJ enterprise versus community
 - Different versions of an operating system
- Might be separate systems with a common framework
- This is another approach that is used
 - But should be considered in architecture & design
 - Generally, for larger systems



Workflows and Automation

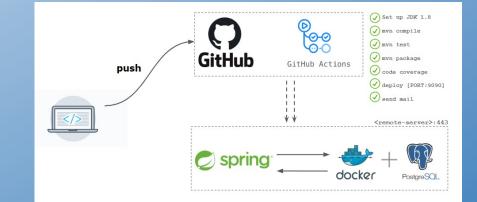
- Continuous Integration requires automation
 - Building, testing, deploying the system
 - Useful in general
- Workflows are a way of doing this automation
 - Triggering event
 - Actions to be run at that point
- Workflows are more general
 - Can be used for lots of different things
 - On check out; on push; on pull; on reviews
 - Supported by tools

What is Continuous Integration



GitHub Actions

- A general approach to automation
 - Implementation of workflows
 - For continuous integration or otherwise
- Actions have a trigger event
 - Push, pull most common
 - Lots of others available
- Actions have one or more jobs
 - Common jobs (e.g., git actions, email, build & test)
 - Scripts
 - Actions can be conditional, run locally or globally, ...



Using GitHub Actions

- Defined in a yaml file in .github/workflows in project directory
 - Useful for team projects
 - But these run in GitHub
- AWS CodePipeline and other alternatives exist
- Git provides a local alternative: hooks
 - Executable scripts in .git/hooks directory
 - pre-commit, prepare-commit-msg, commit-msg, post-commit
 - post-checkout, post-merge (pull), pre-push
- Can also just create your own shell scripts
 - Or ant tasks or make tasks
- You should think about how these could be used in project
 - To simplify your work

Estimating Time and Effort

- Required for Team Organization
 - Divide the project up fairly
 - Get critical pieces working early
 - Especially when others are dependent on them
 - Determine what will be done when
 - Determine who will do what
 - You need to know how long it will take to build the software
- Based on
 - Complexity and size of code involved
 - Complexity of interaction with other components
 - Knowledge & abilities of programmer assigned

COST ESTIMATION

	MEANING		
	Cost estimation is the process of	forecasting the operation	e cost of completion of a project, task or n.
	STEPS IN COST ESTIMAT	ION	COST ESTIMATION METHODS
۲	KNOW YOUR CLIENT : It is imp take all limitations or benefits of clien		TOP DOWN ESTIMATION : Project Manager works from top and breaks
Þ	WORK WITH BUDGET : It must certain ground rules and subject to lin		 down to allocate cost and or hours. ANALOGOUS COST ESTIMATION This method relies on data from similar past projects.
۲	BREAKDOWN APPROACH : An cost estimation must have a break dow		• PARAMETRIC ESTIMATE : It adds a layer of additional relevant information to arrive at the closest estimate.
	HAVE A WRIGGLE ROOM : Sho room for contingencies	uld have a	 THREE POINT ESTIMATION : A three-point estimation is like conducting a scenario analysis.

)/19/24

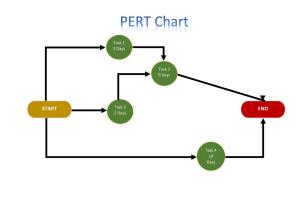
Estimating Time and Effort



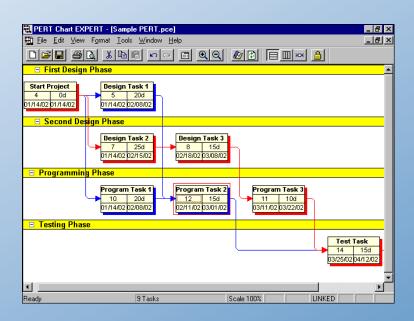
- Known systems take various factors into account
 - COCOMO model is the best known
 - Expert experience is most widely used
- My Approach
 - Expert estimate based on similar code, size, complexity
 - Multiply by 4 (for myself; different factor if others coding)
- Easier to do for smaller pieces of code
 - Rather than the whole system at once
 - Agile sprints only estimate the week's work, not the project

Project Management

- Some pieces of the project are more important than others
 - Required before other pieces can be tested (or even written)
 - Required as a framework for building other pieces
- Project management tries to identify these dependencies
 - Project dependency graph (PERT & GANTT Charts)
- Add time estimates to this graph
 - Identify critical path (what takes the longest)
 - When to start each piece so it gets done in time
 - Who to assign to each piece



PERT and GANTT Charts

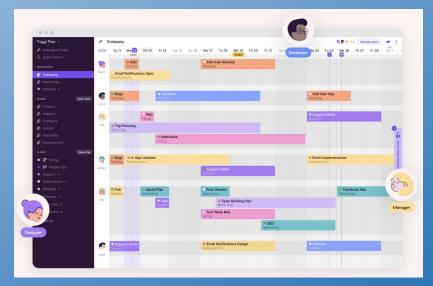


LE	5 - TASI		Vr III ∓ DURCE REPORT PROJECT VIEW DE	VELOPER	FORMAT	DLS	PIPE REPAIR-18.mpp - Project Standard
es	idlines •	Layout	LINERT Insert Column Settings → Column Settings →	Task	Baseline Slippag	je	Gantt Chart Style
~	0	Task Mode ▼	Task Name 🗸	Duration 🚽	Start 🗸	Finish 👻	Jun 2, '19 Jun 9, '19 Jun 16, '19 Jun 23, '19 Jun 30, '19 S. M. T. W. T. F. S. S. M. T. W. T. F. S. S. M. T. W. T. F. S. S. M. T. W. T.
0			4 Pipe Repair & Improve	17 days	6/3/2019	6/25/2019	Pipe Repa <mark>ir & Im</mark> prove
1	C.		Contract Award Date	0 days	6/3/2019	6/3/2019	-+ 6/3
2	4		Start Project	0 days	6/3/2019	6/3/2019	₩ 6/3
3	1		Project Completion Date	0 days	6/25/2019	6/25/2019	€/25
4		-	Demolition Piping	2 days	6/3/2019	6/4/2019	Demolition Piping
5	4		Drain Piping System	1 day	6/3/2019	6/3/2019	Ly_Drain Piping System
6	4		Remove Damaged Piping	1 day	6/4/2019	6/4/2019	Remove Damaged Piping
7			Installation Piping System	12 days	6/5/2019	6/20/2019	Installation Piping System
8			▲ Piping	12 days	6/5/2019	6/20/2019	Piping
9	-		Install Piping & Couplings	3 days	6/5/2019	6/7/2019	Install Piping & Couplings
10	4		Test Piping at Pressure Inspection	1 day	6/10/2019	6/10/2019	Test Piping at Pressure Inspection
11	÷		Insulate Piping	5 days	6/14/2019	6/20/2019	Ansulate Piping
12		->	Thrustblock	8 days	6/11/2019	6/20/2019	Thrustblock
13	4		Set Forms	1 day	6/11/2019	6/11/2019	-Set Forms
14	1		Lay Rebar	1 day	6/12/2019	6/12/2019	Lay Rebar
	4		Pour Concrete	1 day	6/13/2019	6/13/2019	Pour Concrete
16	1		Strike Forms	1 day	6/20/2019	6/20/2019	Strike Forms
17			A Quality Assurance	3 days	6/21/2019	6/25/2019	Quality Assurance
	4		Write Quality Assurance Report	2 days	6/21/2019	6/24/2019	Write Quality Assurance Report
19	4	-	Final Quality Assurance Inspection	1 day	6/25/2019	6/25/2019	-Final Quality Assurance:Inspe

Project Management Software

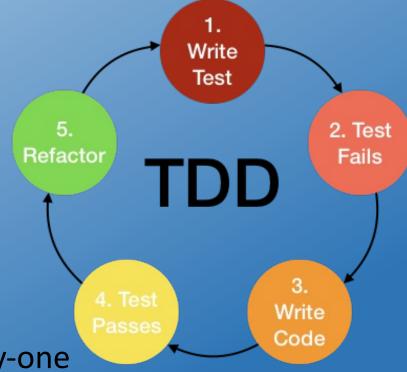
- Microsoft Project Manager
- JIRA
- GitHub issues (gantt chart generator)





Test-Driven Development

- This is emphasized in agile development
 - Used in earlier courses
- Write the test cases first
 - Develop the code to pass the test cases
- Design the code & system so it can be tested
 - This is not as easy as it sounds
 - Especially for UI-based, interactive systems
- Write the user interface first
 - Develop code to handle user interactions one-by-one
 - Simulate the user interactions with test cases
 - Good alternative or addition for UI-heavy applications



Test-Driven Development Pros/Cons

Advantages

- You can see how well you are doing
- Provides a focus for code development
- Better understanding of what needs to be written
- Ensures code works as it is written

• Disadvantages

- Test cases can be difficult to write, especially interactive/graphical
 - Can be as much work as actually writing the code
- Might need to write mocking (dummy) libraries, etc.
- Test cases and testing code can be buggy
- Evolving code means evolving the test cases as well
- Can overfit code to test cases, not look forward
- Files, external systems, etc. get in the way

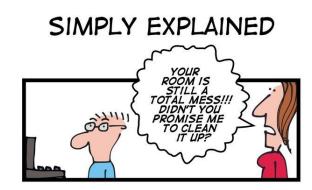


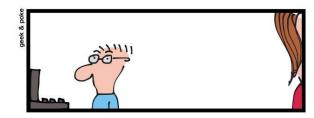
IS TDD THE RIGHT CHOICE FOR

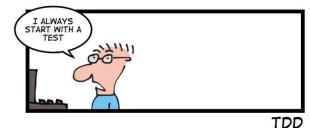


Test Cases

- Are an essential part of the system
 - Whether written before, during or after coding
 - Needed to avoid regression errors
 - Needed to approve code before general use
- Can clarify design before coding
- Can need as much effort as the code
 - Typically, with more bugs
 - Often not maintained as well
- Alternatives
 - Dogfooding
 - Automated bug reporting
 - Alpha and beta testing







Research in Programming Teams

- Continuous Integration and workflows
- User interface testing
- Design for software families

PROJECT HOMEWORK

- Check out various communication tools
 - SLACK, GitHub Issues, ...
 - Choose one or more for your team
 - Set it up, start using it
- Finalize project roles
 - Ensure absent people agree to their role
 - Ensure everyone is "happy"
- Set up an initial PERT/GANTT chart for your project
 - Possibly using GitHub issues
 - Little to put in there now since we haven't done design
- In addition to deciding on an overall software architecture
 - Based on requirements and specifications
 - Software architecture hand-in due Thursday (one per team)

HOMEWORK / Further Reading

- Code up the bouncing balls assignment
- Make your programming assignment code match the coding style for your project
 - With a personal package name or equivalent
 - With personal class names matching package
 - Modified for different programming language if needed
- Programming assignment should be working by class Thursday

Further Reading

- <u>https://github.com/minhloc2011/books/blob/master/People</u> ware%2C%203rd%20Edition.pdf
- <u>https://martinfowler.com/articles/continuousIntegration.ht</u>
 <u>ml</u>