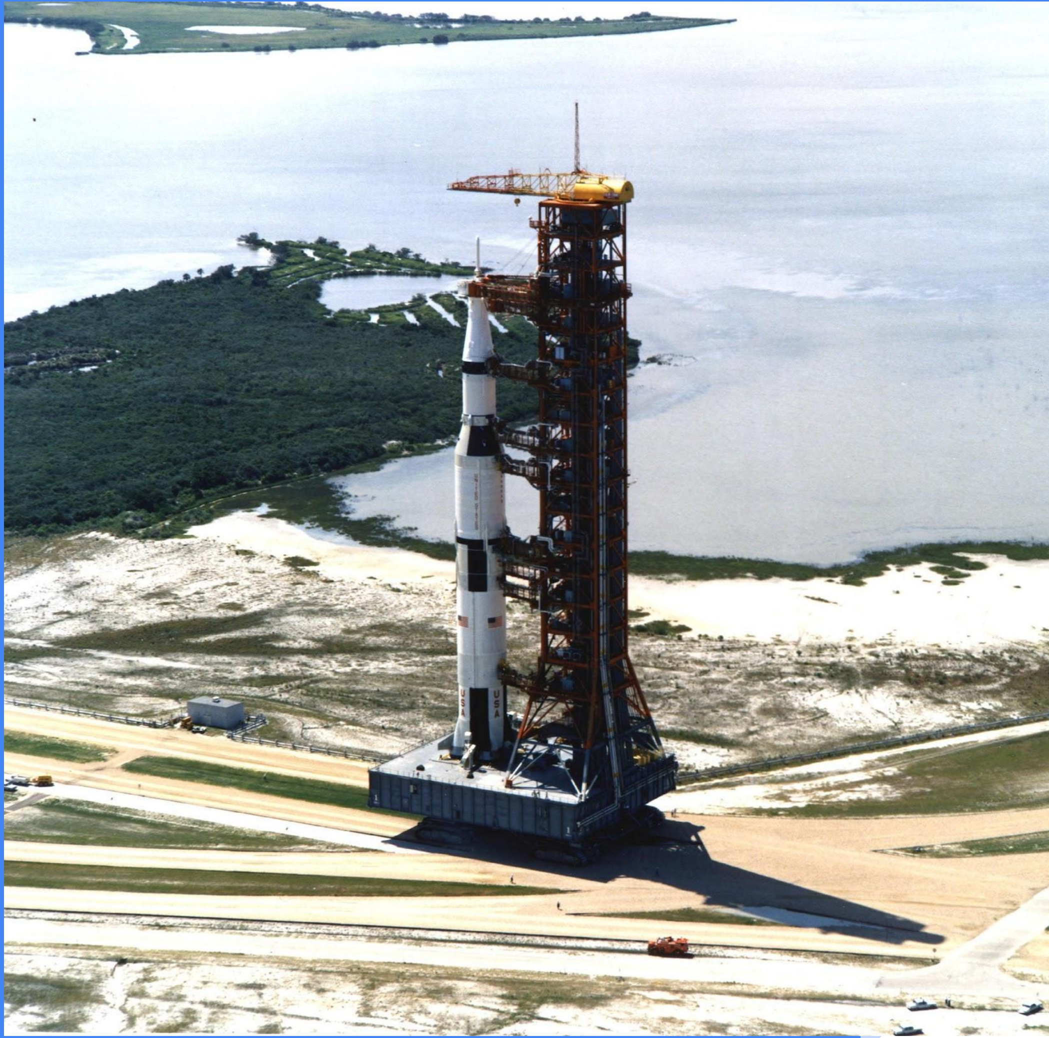
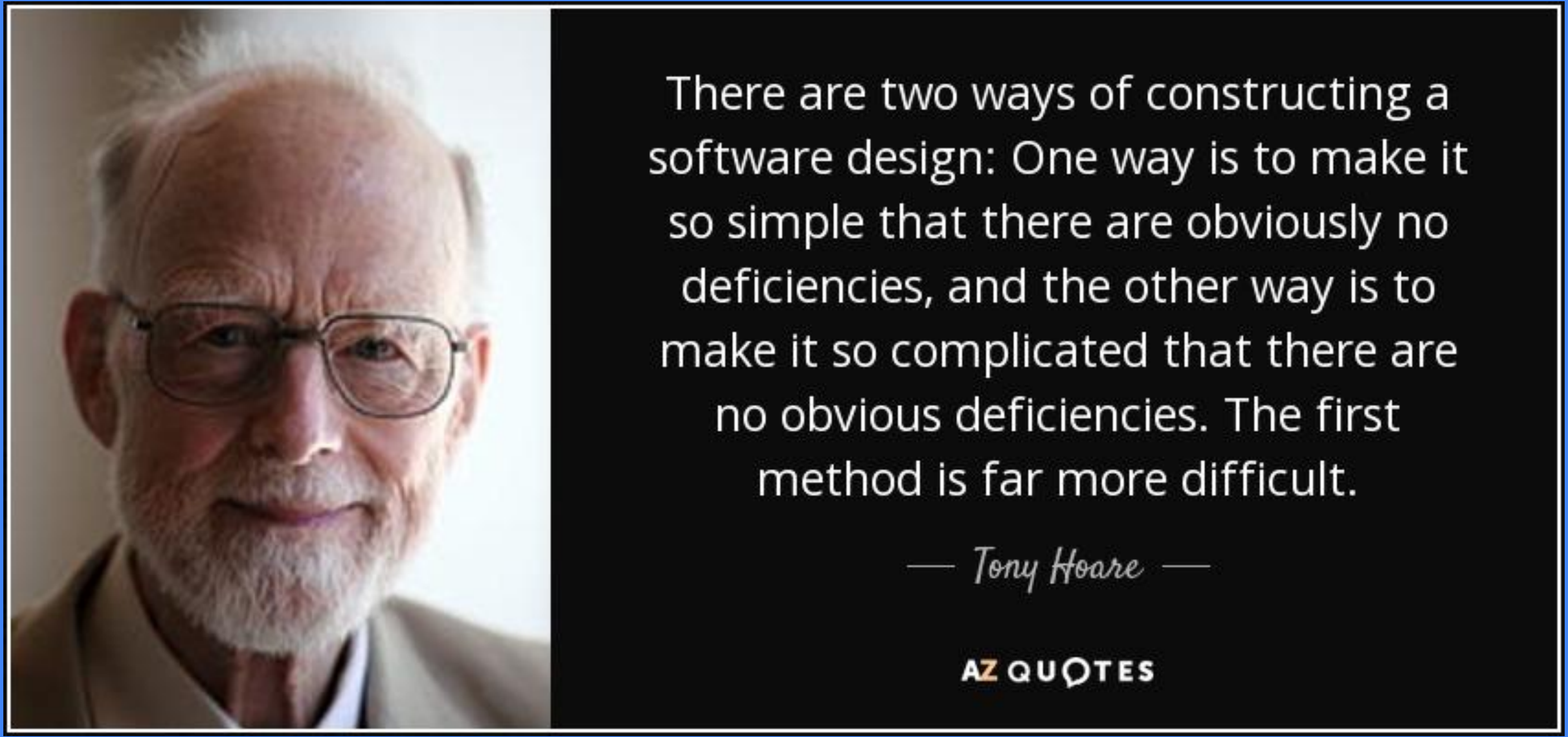
Spaceship!

SE101 / Ideas Clinic

Patrick Lam
Derek Rayside
John Harris
Sanjeev Bedi





> There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.
>
> — Tony Hoare —

What is Software Engineering?

What is software engineering?

IEEE

“The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"

—IEEE Standard Glossary of
Software Engineering Terminology

What is software engineering?

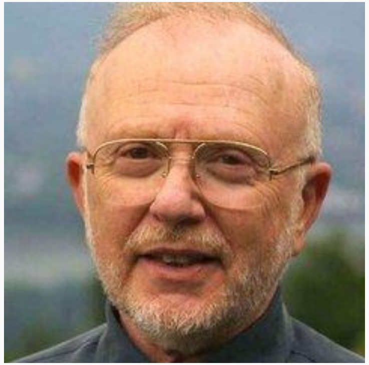
PEO

1. any act of planning, designing, composing, evaluating, advising, reporting, directing or supervising (or the managing of any such act);
2. that requires the application of engineering principles; and
3. concerns the safeguarding of life, health, property, economic interests, the public welfare or the environment.

—*Professional Engineers Act Ontario (section 1)*

What is software engineering?

“Multi-*person* development of multi-*version* programs”
—David L. Parnas & Brian Randell



CS137 is Computer Science / Programming

- *Single-person* development of *single-version* programs
- Concepts:
  - Data structures
  - Algorithms
  - Modularity mechanisms (e.g., procedures, classes, etc)
  - Pointers
  - Space + time complexity analysis
- Essential knowledge + skills!

SE101 is Software Engineering

1. Professional Responsibility
   - Duty to public welfare is paramount
   - Engineers build and safeguard the infrastructure of society
2. Teamwork
   - Multi-Person Development
3. Configuration Management
   - Multi-Version Software

Teamwork

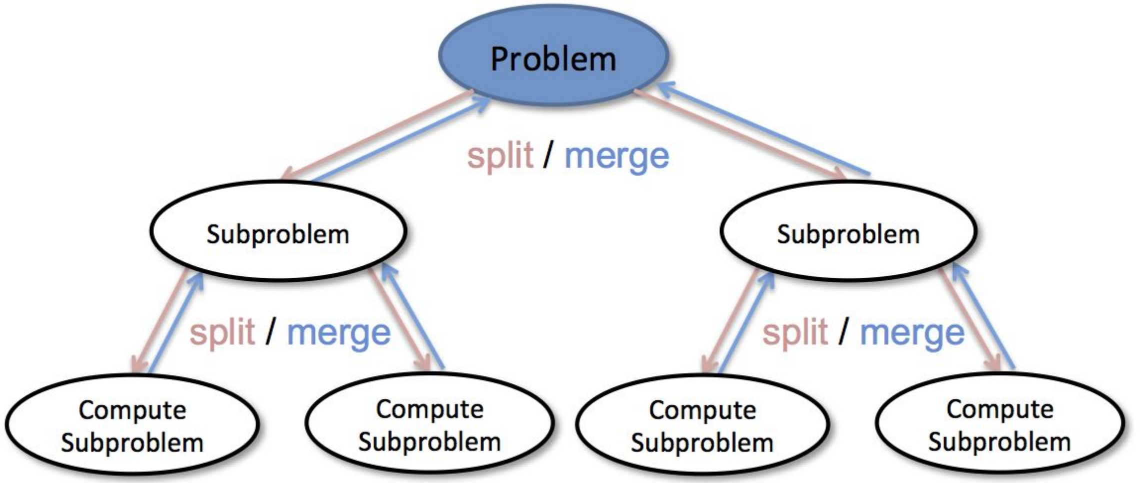
Teamwork

1. Organization of labour/tasks
2. Decision-making strategy
3. Communication protocols
4. Support Strategies

← Focus today

Divide & Conquer

1. What are the main algorithms required?
2. Each person works independently on one algorithm.



Problem

split / merge

Subproblem

Subproblem

split / merge

split / merge

Compute Subproblem

Compute Subproblem

Compute Subproblem

Compute Subproblem

Divide and ~~Conquer~~ Collide

2 UNIT TESTS, 0 INTEGRATION TESTS
via reddit.com/r/programmerhumor

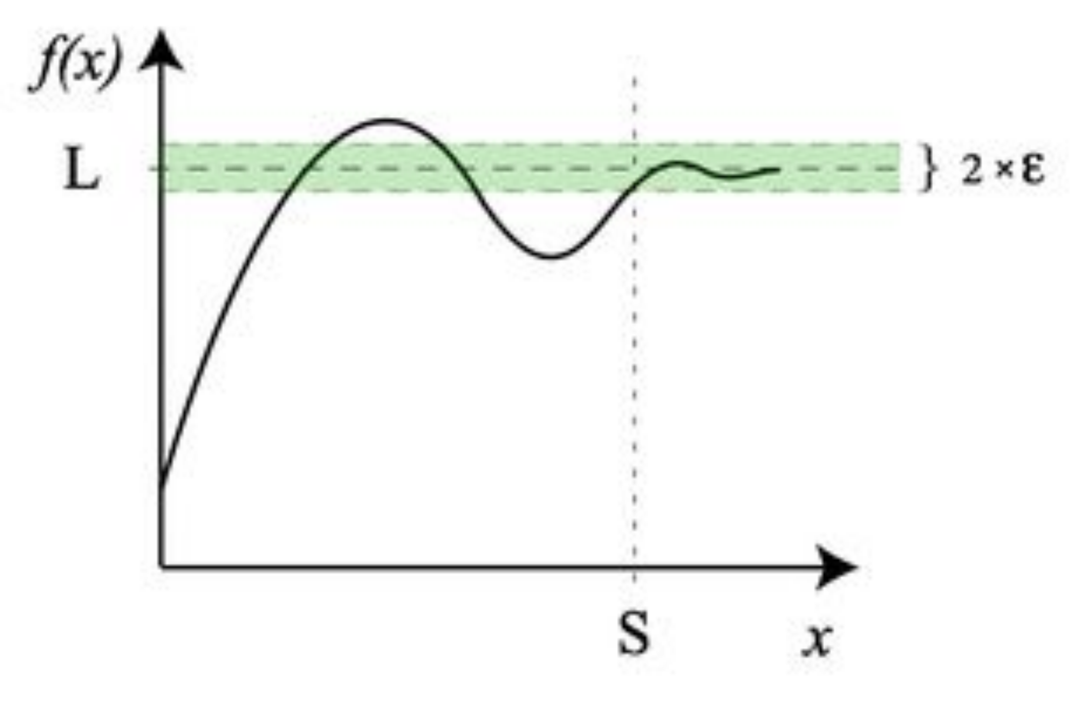
Divide and ~~Conquer~~ Disconnect

Divide and ~~Conquer~~ Confuse

Collaborate & Converge

Agile

1. Tests
2. Interfaces
3. Algorithms
4. Learn
5. Iterate



1st Iteration

1. *Tests* : One good one. Not too complicated.
2. *Interfaces* : **Focus of the 1st iteration!**
3. *Algorithms* : Simplest possible. Hard-code solution.
4. *Learn* : Merge, compile, run, observe, fix.
5. *Iterate* : Back to step 1.

TDD

2nd Iteration

1. *Tests* : Add another interestingly different one.
2. *Interfaces* : Revise as necessary.
3. *Algorithms* : **Generalize to cover both test cases.**
4. *Learn* : Merge, compile, run, observe, fix.
5. *Iterate* : Back to step 1.

Test-Driven Development

Teamwork

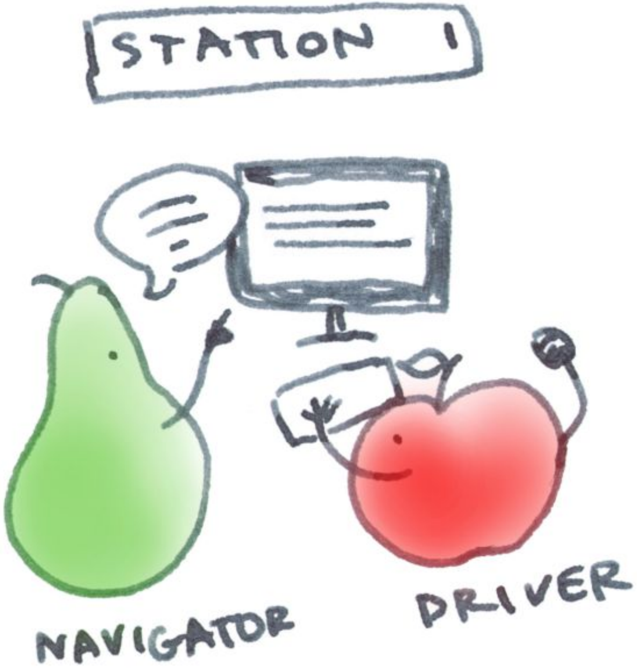
1. Organization of labour/tasks
2. Communication protocols
3. Decision-making strategy
4. Support Strategies

Collaborate & Converge

Pair Programming (Communication)

Today!

- Working together
- Better code
- Slower development
- Reduce
  - Bugs
  - Misunderstandings
  - Integration issues
  - Blame (unproductive)

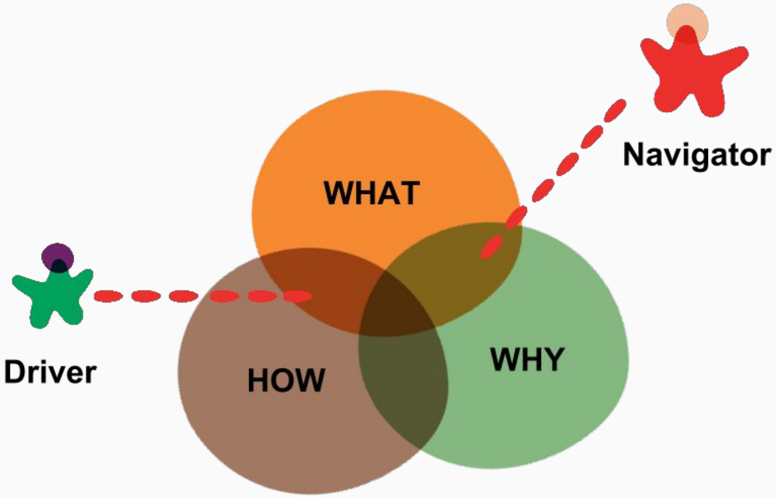


Pair Programming Roles

Driver has the keyboard.

Navigator focuses on big picture (and pedantic details).

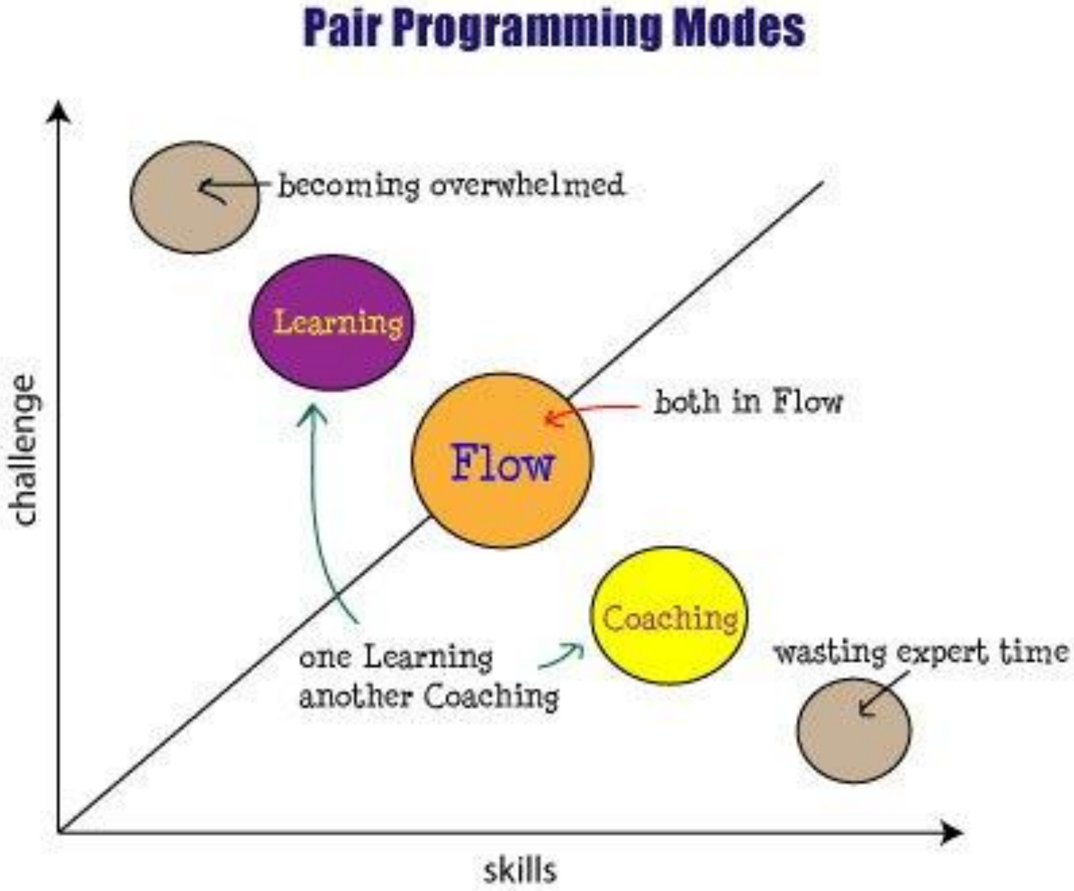
Swap roles.



Pair Modes

Aim for the middle!

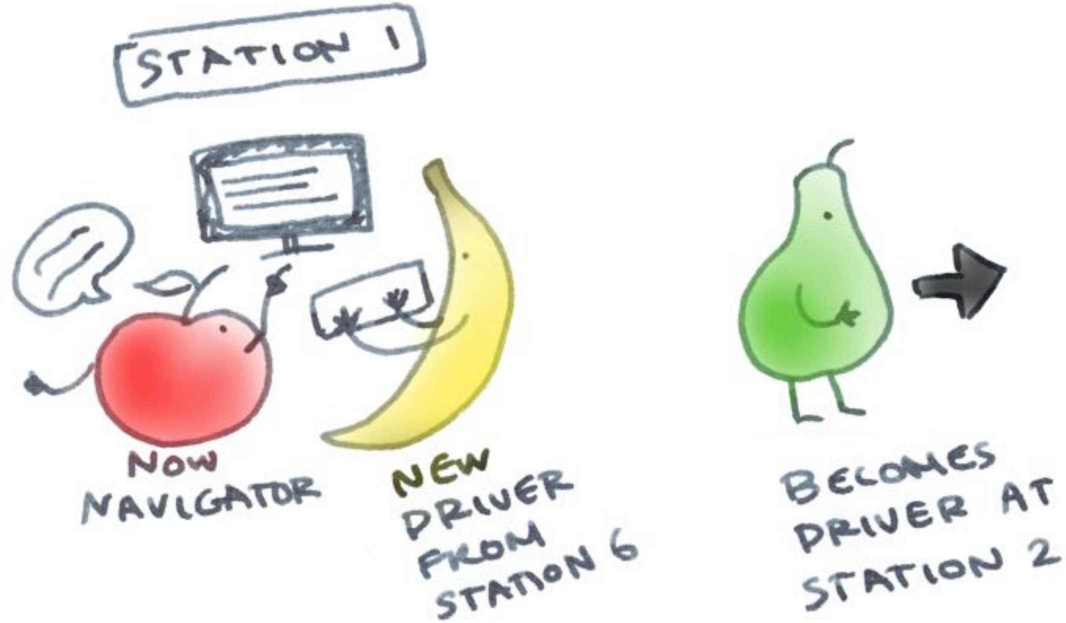
There is no reason to be at the extremes here.



Rotate Pairs

Learn more about the system as a whole.

- Reduce
  - Bugs
  - Misunderstandings
  - Integration issues
  - Blame (unproductive)



Teamwork

1. Organization of labour/tasks
2. Communication protocols
3. Decision-making strategy
4. Support Strategies

Pair Programming

Decision-Making Strategies

Pick one

1. By Leader
   - Dominant personality : Default
   - Elected : Representative democracy
2. By Majority : Direct democracy
3. By Consensus : Discuss until everyone agrees

https://uwaterloo.ca/centre-for-teaching-excellence/teaching-resources/teaching-tips/developing-assignments/group-work/group-decision-making

If you are the dominant personality ...

- Everyone already knows it
  - You don’t need to prove anything to anyone
  - You don’t need to pick fights with anyone
- You have the power to be ***kind***
- ***Consensus*** results in the best team outcomes
  - Use your power to build consensus
  - You can even build consensus around someone else’s ideas!
  - Ensure that everyone has a chance to speak their mind

Teamwork

1. Organization of labour/tasks
2. Communication protocols
3. Decision-making strategy
4. Support Strategies

Pick one

Support Strategies

Great engineers:

- Identify hard problems
- Ask for help
- Give help in a constructive way
- Succeed together

If the spaceship doesn’t launch, nobody cares that you think you are so smart.

Teamwork

1. Organization of labour/tasks : *Collaborate & Converge*
2. Communication protocols : *Pair Programming*
3. Decision-making strategy : *Pick One*
4. Support Strategies : *Succeed Together*

Software Engineering:

Multi-*Person* Development
Of
Multi-*Version* Software

Configuration Management

Mastering the basic operations

- Commit (with meaningful comment!)
- Push
- Pull
- Merge

Multi-Version Software

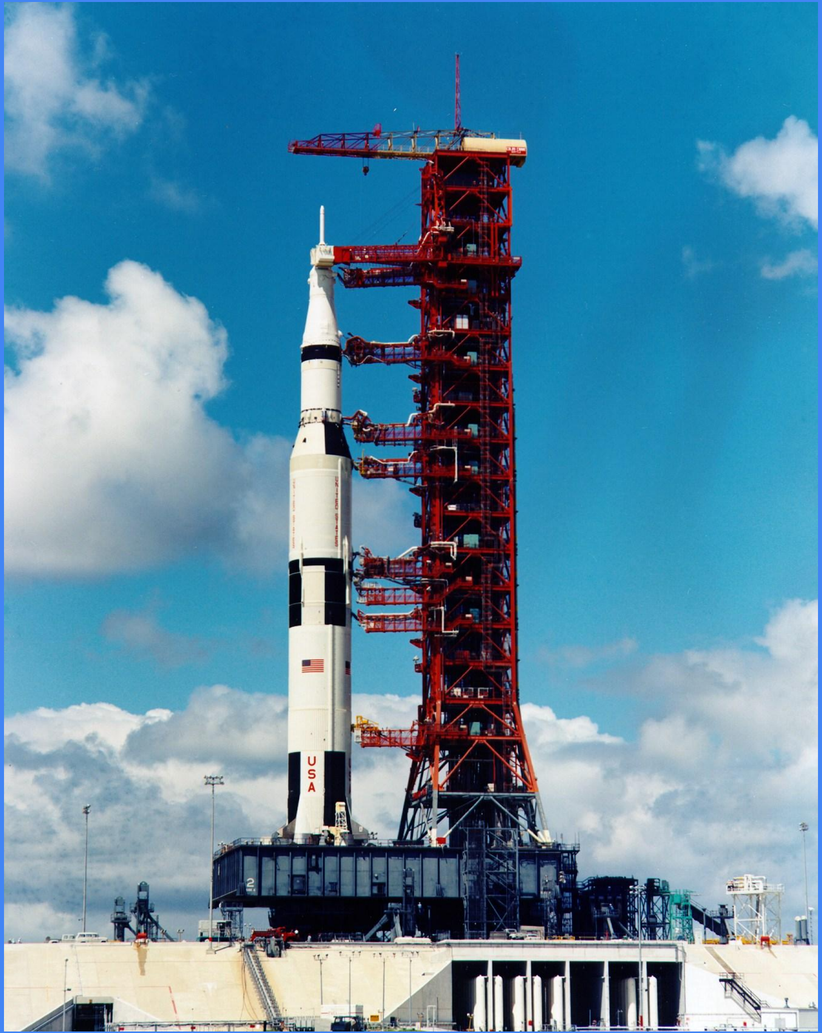
- Feature flags (*recommended -- do this today*)
  - A global variable or parameter that controls which code to run e.g., use first version of algorithm or improved version
  - Reduces merge/integration conflicts
- Branches (*not supported today*)
  - Develop next version of algorithm on a new branch
  - Facilitates code reviews

Feature Flags to Select Algorithm Variant

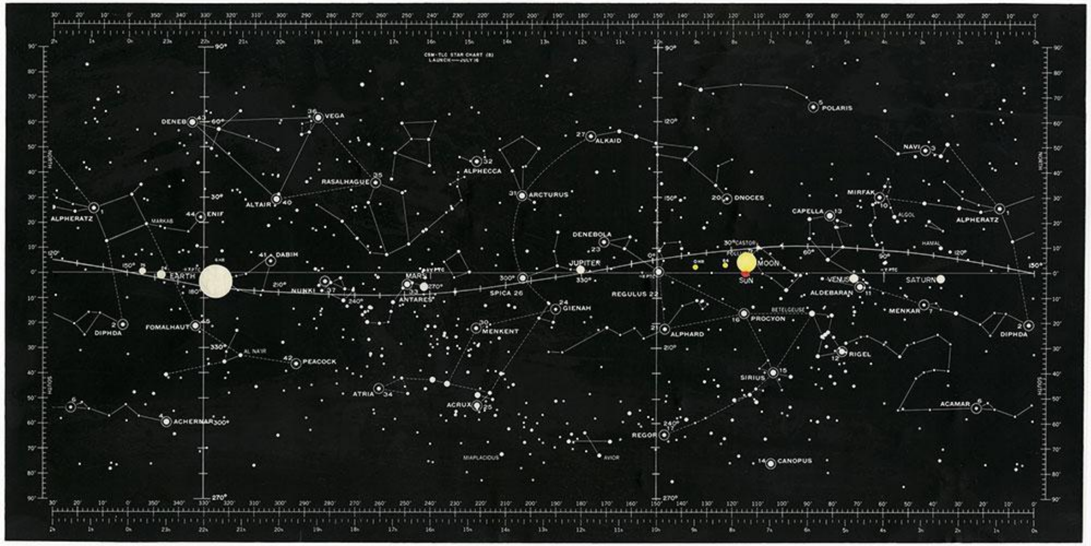
```
proc p(x, y) {
    if (Globals.AlgoA == 1)
        p1(x,y);
    else if (Globals.AlgoA == 2)
        p2(x,y);
}
```

There are many ways this code could be written, many of which would be more elegant. The point here is to communicate the idea in a way that everyone will understand.

T-10 minutes



Fly to distant planet in distant solar system

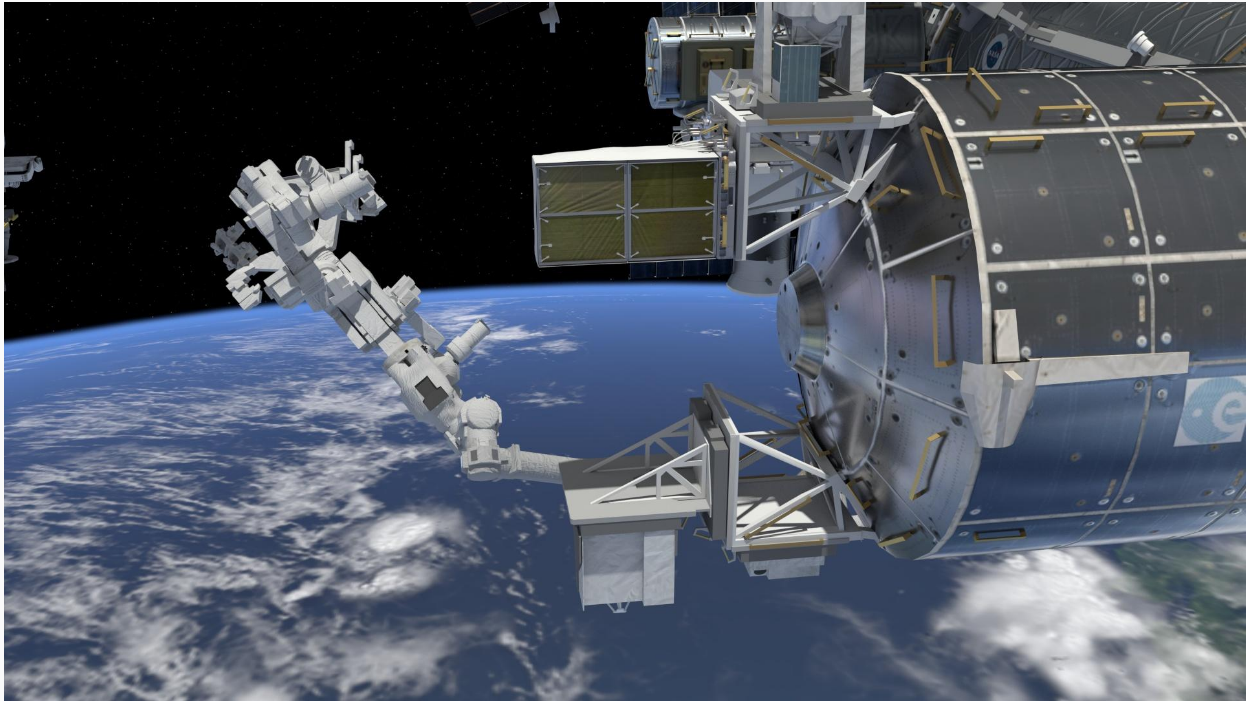


Within a solar system

1. Select target:
    - warp gate?
    - planet?
2. Fly there
3. Shoot asteroids on the way!

Sensors

- Planets
    - habitable?
- Warp gates
    - target?
- Asteroids
    - Position
    - Velocity

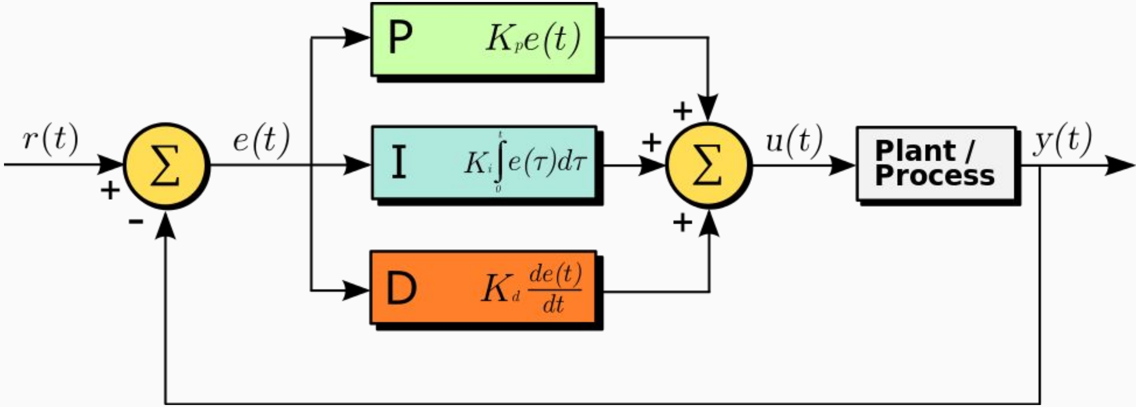


Navigation: warp-gate path through galaxy

1. First algorithm:
   - Solve map by hand
   - Hard-code which warp-gate to take
2. Second algorithm:
   - Dijkstra’s shortest path
   - https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Propulsion: Flying within a solar system

1. Algorithm 1:
   - UFO mode
   - Just set velocity
2. Algorithm 2:
   - PD Controller
   - Preview of SE380

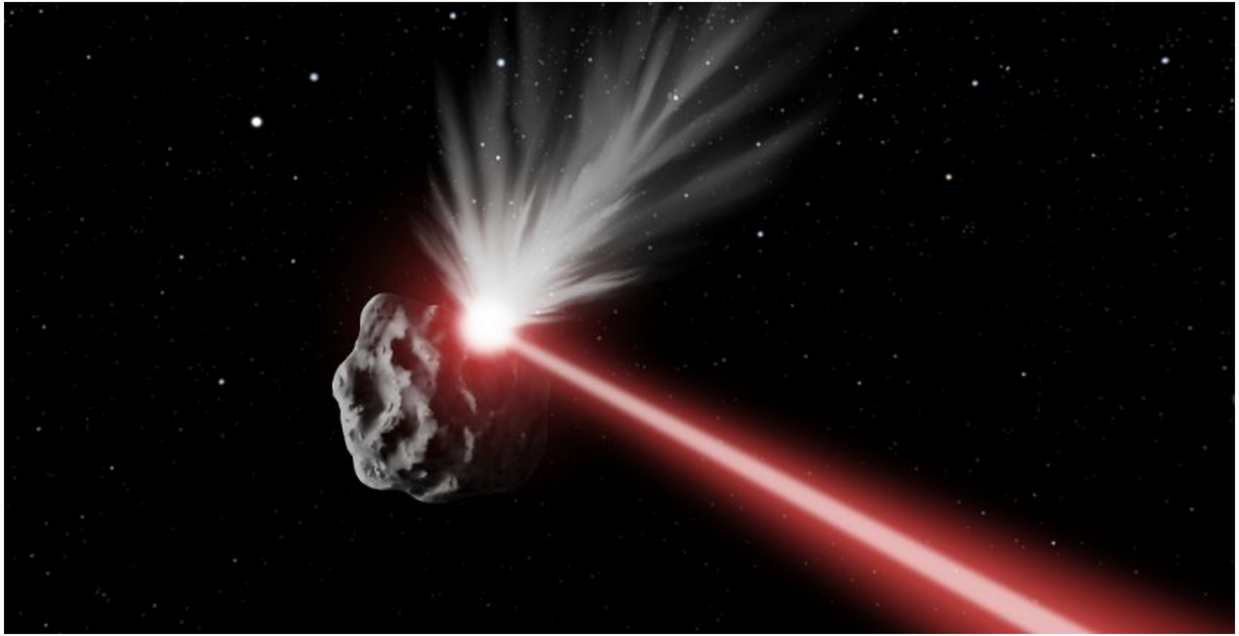


Defence: shoot the asteroids!

Asteroid is moving

Ship is moving

Good luck!



Four Subsystems

- Sensors
- Navigation
- Propulsion
- Defence

Which ones communicate with each other?
What information do they share?
Interfaces!

Spaceships & Squads

1. ~16 students per spaceship
   - randomly assigned by TA
   - 1 repo per spaceship
2. 4 squads per spaceship
   - ~4 students per squad
3. Random assignment of students to initial squads:
   - Sort by student ID
   - Label list ABCD ABCD ABCD ABCD
   - A’s together; B’s together; C’s together; D’s together

First Iteration

1. Arbitrary assignment of squads to subsystems
   - By students
2. Split each squad into two pairs: *inputs* + *outputs*
   - Input pair from squad X talks to output pair from squad Y
   - Define interfaces between subsystems. Learn spaceship API.
3. Each squad develops first iteration of their subsystem.
4. Pull/merge/commit/push/run/observe
5. Perform on the big screen!

Learn Unity

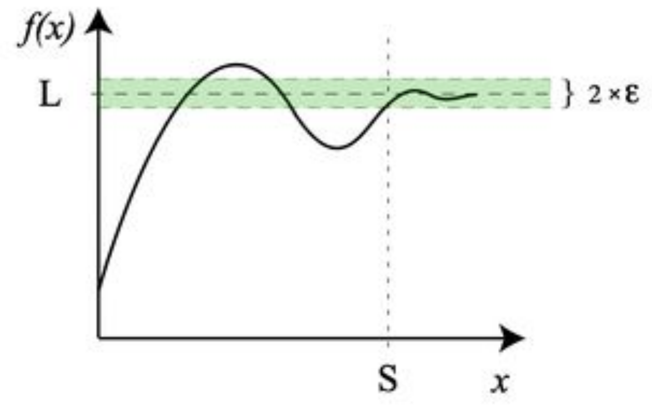
If you want to

Second Iteration

1. Switch squads/pairs/subsystems
    - Squads are now ABCD
    - Arbitrary assignment of squads to subsystems by students
    - 1 person in each new squad will be familiar with subsystem
2. Get 2nd universe. Run your spaceship. Observe failures.
3. Develop second iteration
4. Pull/merge/commit/push/run/observe
5. Perform on the big screen!

If you want to

Third Iteration



1. Swap pairs (keep squads and subsystems)
2. Get 3rd universe. Run your spaceship. Observe failures.
3. Develop third iteration
4. Pull/merge/commit/push/run/observe
5. Perform on the big screen!
6. Repeat

Maybe next week?

Have we done this before?

Will there be bugs?

Lift-off!

Day 1 Wrap-up

Software Engineering:

Multi-*Person* Development
Of
Multi-*Version* Software

Collaborate & Converge

+

Version Control

Teamwork

Technology

Now to next week

- Reflect
- Discuss
- No development
    - No typing
    - Pencil and paper only
      (if something must be written)

Engineering 101 is about professional behaviour and responsibility. Professionals are trustworthy. We trust you to not work on this code until next Thursday.

Collaborate & Converge
+
Version Control

Teamwork

Technology