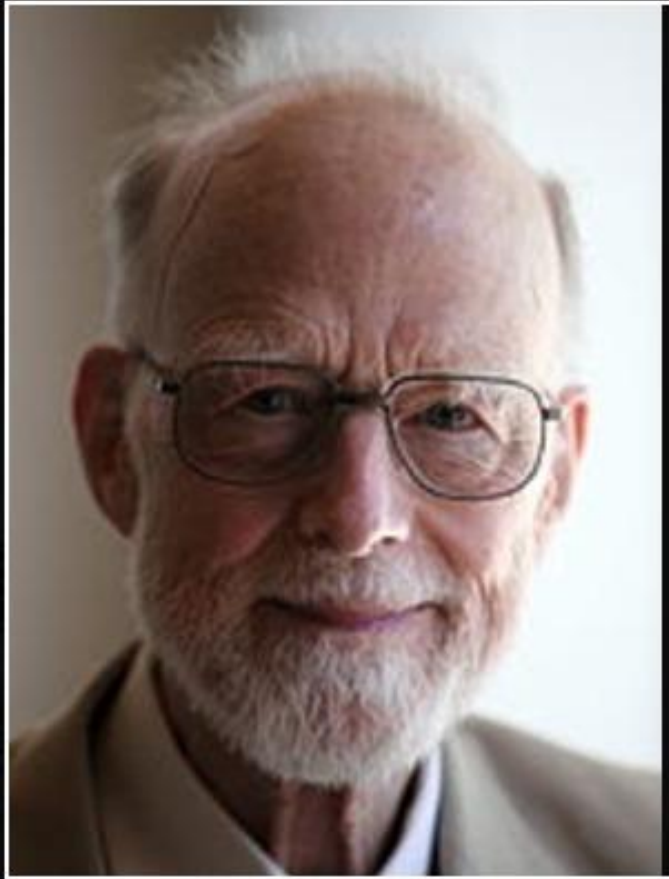
SE101
Ideas Clinic

Prof Derek Rayside
Future SE Director (January 2020)



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 —

AZ QUOTES

Engineer an AI for a space ship!

Engineering is Teamwork

Software Engineering:

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

What is Teamwork in SE?

Teamwork in SE

- Interpersonal Skills
- Version Control (e.g., Git)
- Pair Programming
- ***Division of Labour -- Today!***

How are you being graded?

Marks based on Personal Reflections

Every Day on LEARN

1. What did you learn?
2. What went well?
3. What could improve?

Write about ***Teamwork*** (and this Activity).

Not Graded

- Your spaceship
- Your code
- Your algorithms
- Your Git history

All of the technical material is beyond 1A

You will be graded on this stuff after 1A, but not now:

- Object-Oriented Programming [CS138 + CS247]
- Dijkstra's algorithm [CS240]
- PD controllers [SE380]
- Testing [SE465]
- Unity game engine [nope]

Gives you a preview of the future curriculum.

What is Teamwork in SE?

Teamwork in SE

- Interpersonal Skills
- Version Control (e.g., Git)
- Pair Programming
- ***Division of Labour -- New Today!***

Teamwork:
Interpersonal Skills

- Listening
- Respect
- Communication
- Conflict Resolution
- **Be nice!**

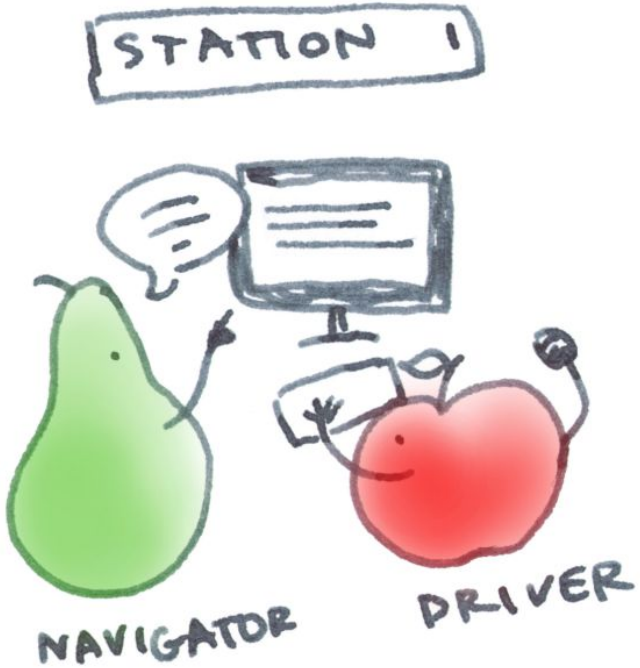
Teamwork: Personality & Conflict Analysis

- More advanced teamwork instruction often involves personality analysis:
  - Myers-Briggs Type Indicator
  - The Big 5 Personality Inventory
- We can do a simpler and easier framework with PowerPuff Girls:
  - Bubbles: *cooperator* **(sugar** / blue**)**
  - Buttercup: *fighter* **(spice** / green**)**
  - Blossom: *leader* **(everything nice** / red**)**
- What's your default? (Derek is Bubbles.)
- In a given situation, when should you exercise a different part of your personality to help the team move forward?

Pair Programming (Communication)

Today!

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

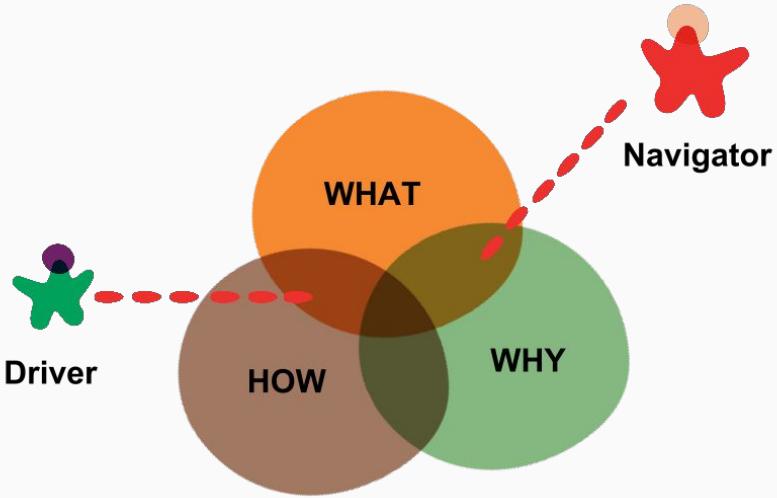


Pair Programming Roles

Driver has the keyboard.

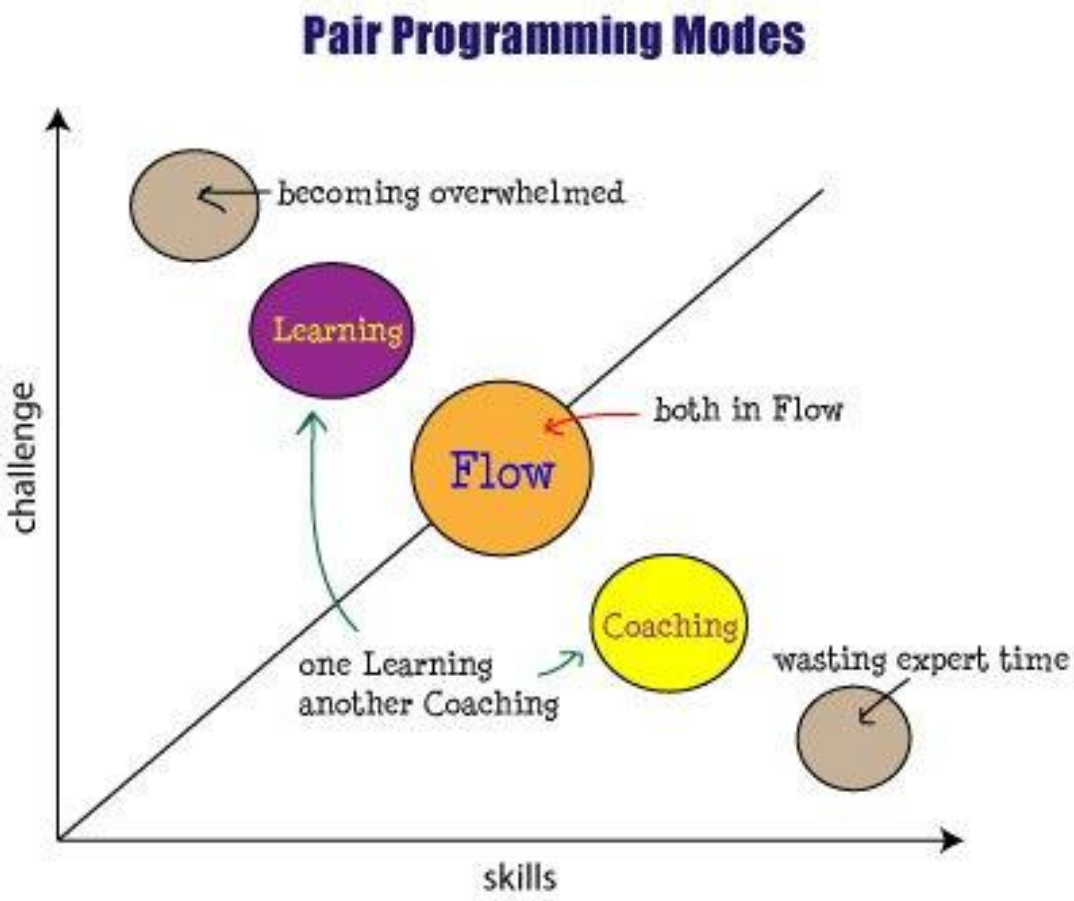
Navigator focuses on big picture (and pedantic details).

Swap roles.



Pair Modes

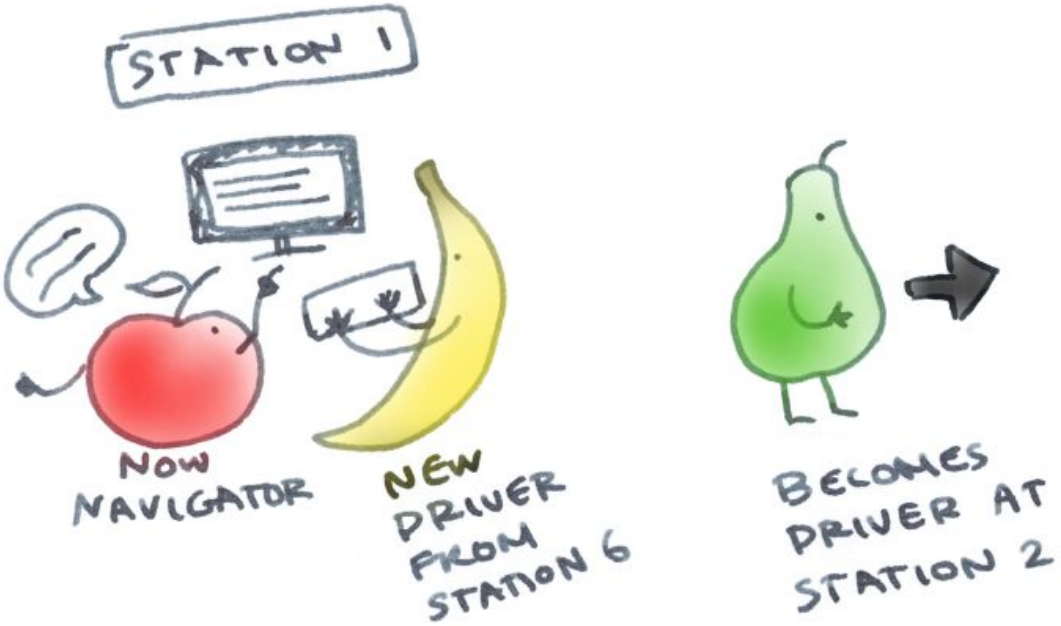
We are here to learn. There is no wasting expert time.



Rotate Pairs

Learn more about the system as a whole.

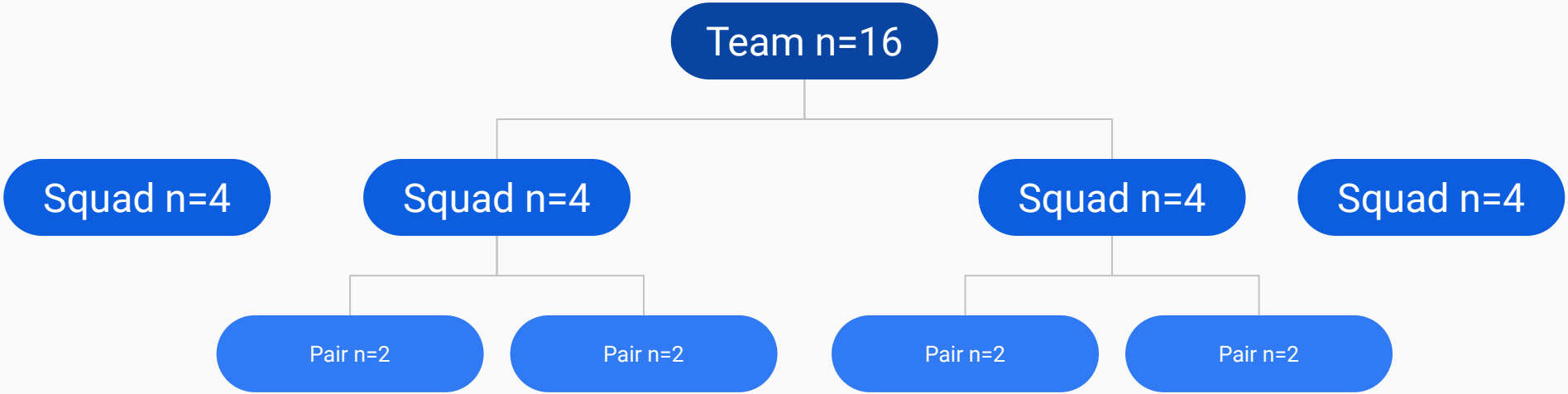
- Reduce
  - Bugs
  - Misunderstandings
  - Integration issues
  - Blame (unproductive)



Division of Labour: Who does What?

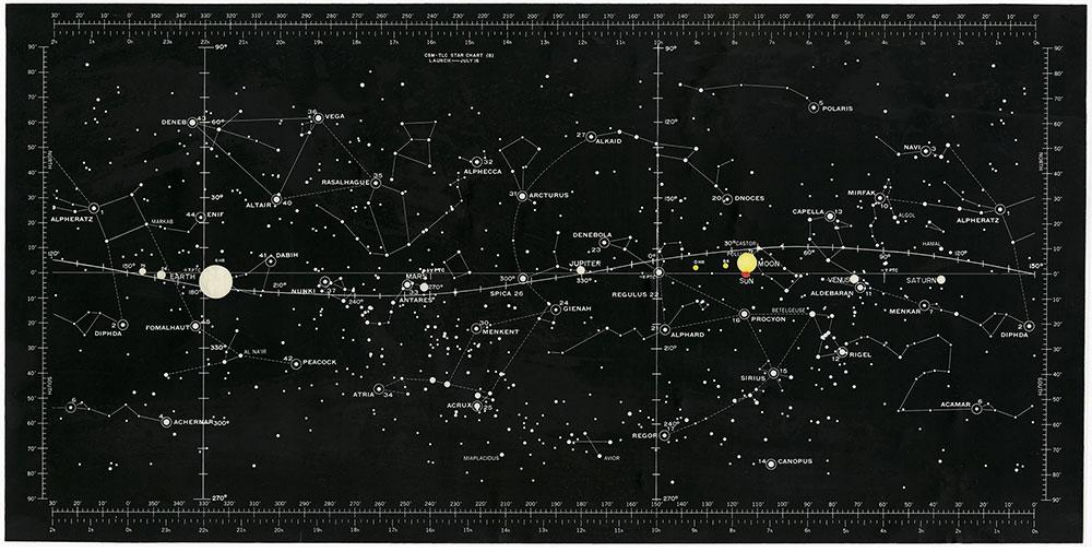
Who

Team / Squad / Pair



What

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!

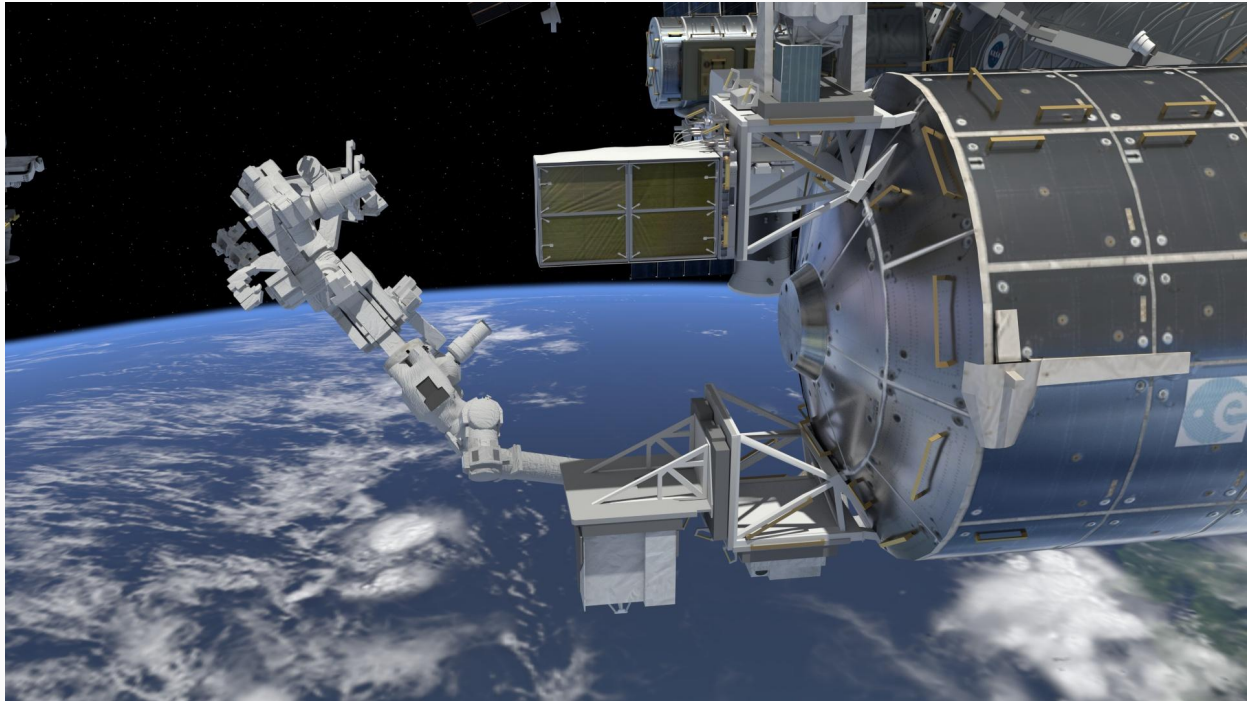
Four Subsystems

- Sensors
- Navigation
- Propulsion
- Defence

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

Sensors

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

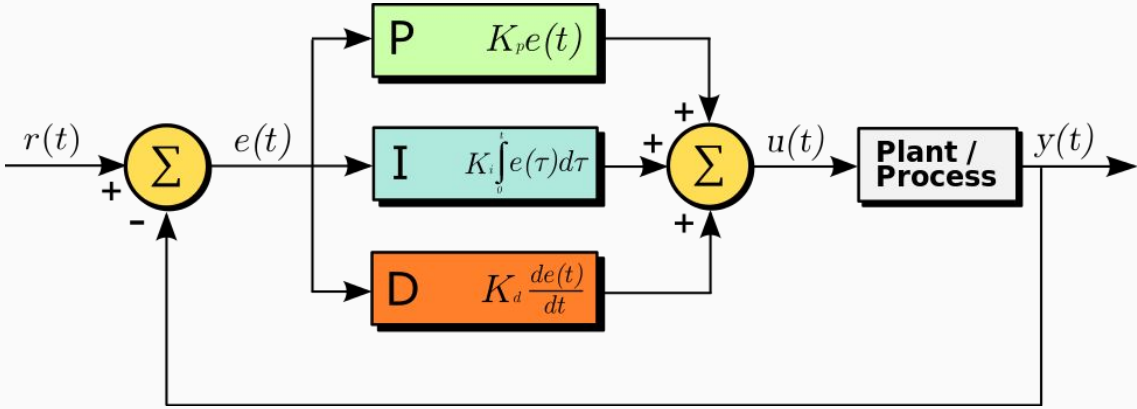


Navigation: warp-gate path through galaxy

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

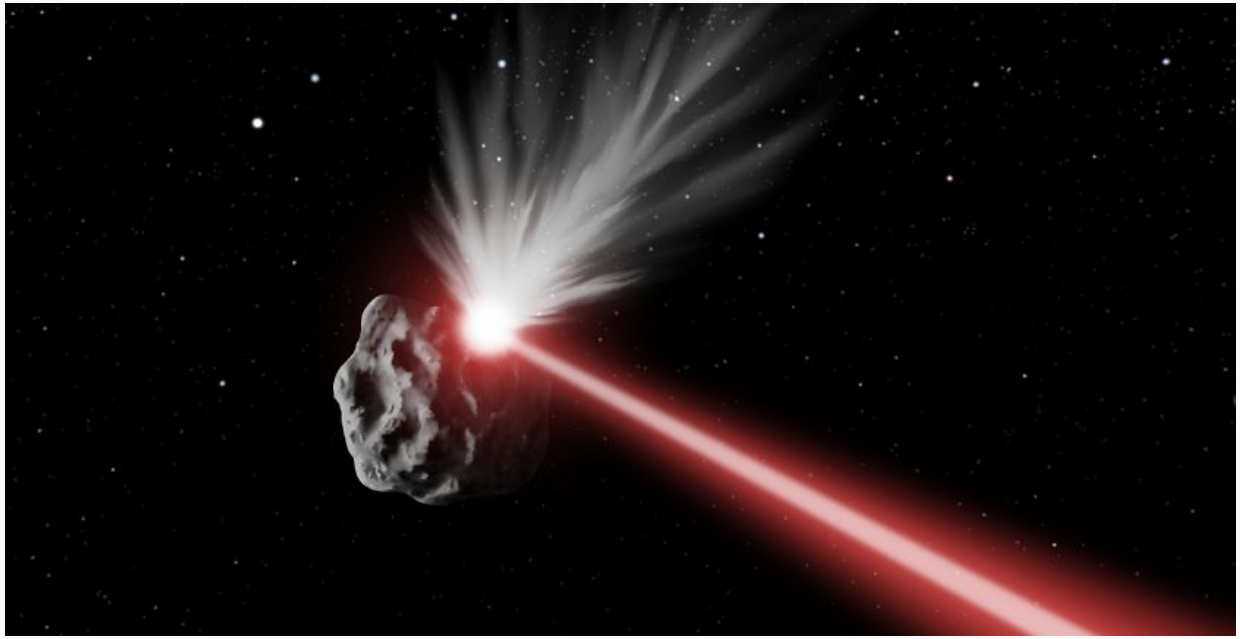


Defence: shoot the asteroids!

Asteroid is moving

Ship is moving

Good luck!

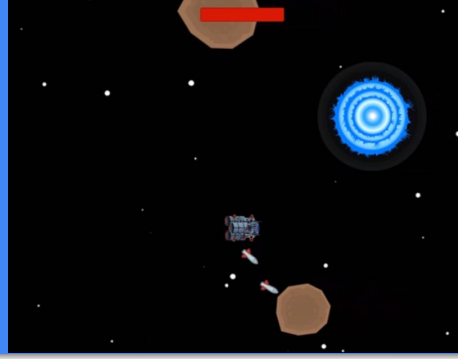


Four Subsystems

- Sensors
- Navigation
- Propulsion
- Defence

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

Demo Video



https://1drv.ms/v/s!Av09ni7UtRI3h9gt2U0zPUy5lupkXw?e=shNgJF

Fun video:

https://1drv.ms/v/s!Av09ni7UtRI3h9gu7vsB2eyyO4bPzQ?e=VYu5nS

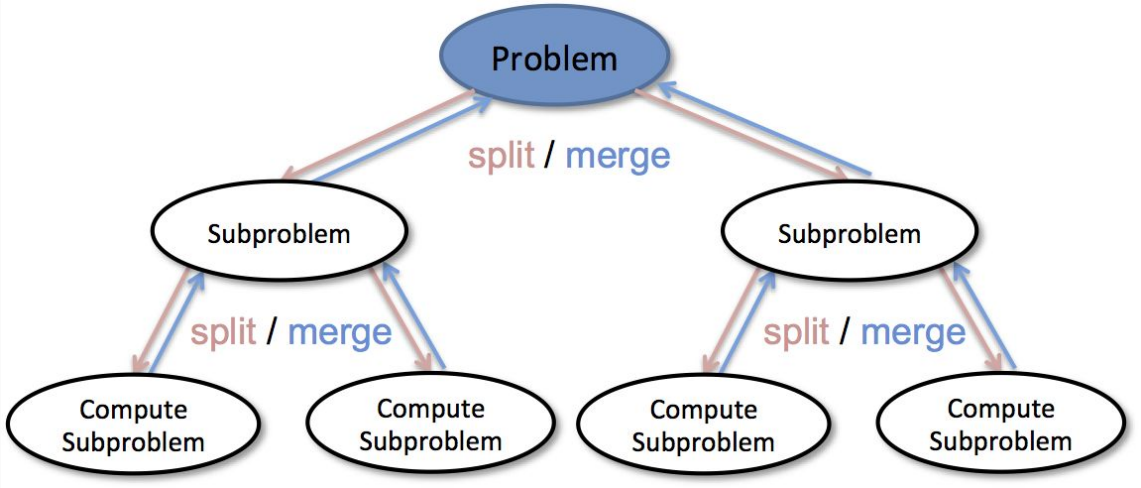
Who does What?

Two Ways to Assign Tasks

Divide & Conquer	Collaborate & Converge
Functionality First	Integration First
Doesn't scale!	***What we will learn.***
Leads to integration failures	Build functionality after integration.

Divide & Conquer

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



Divide and ~~Conquer~~ Collide

Divide and ~~Conquer~~ Disconnect

Divide and ~~Conquer~~ Confuse

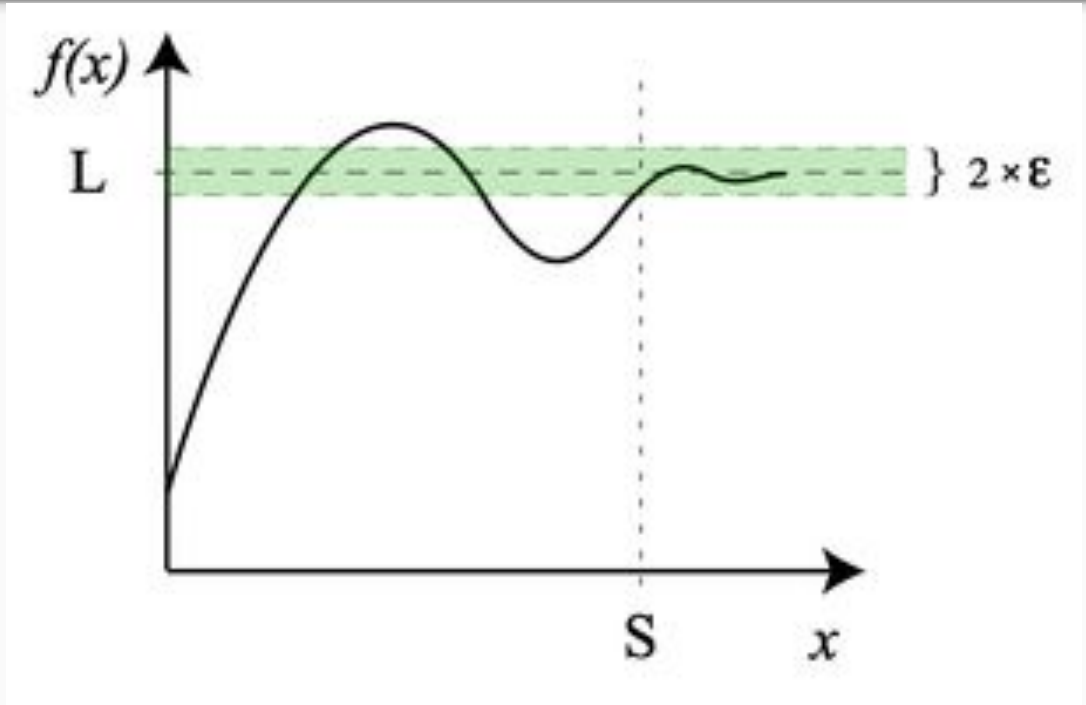
Functionality First Doesn’t Scale

Integration failures

Collaborate & Converge

Agile

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



1st Iteration

Parts don’t do much.
But they are integrated.

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.

Fake it ‘til you make it.

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.

Each Squad on a Subsystem

- Pair1: learn API to use, experiment
  - What can you learn/test/do without sensor data?
- Pair2: design interfaces with other Squads/Subsystems
- Today’s Goal:
  - Basic/fake functionality for each subsystem
  - Initial agreement on interfaces

Lift-off!

Goals For Monday

Better Network on Monday

- Git installed + working
- Unity installed + working
- Spaceship repo cloned to your computer
  - Read Activity Manual .docx
  - Explore Assets/Student Scripts --- *where your code will go on Monday*
  - Explore Assets/Sandbox --- *library code you will use*
- ***DO NOT WRITE CODE***
  - ***Our learning objective is Teamwork***

Learning Reflection

Software Engineering:

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