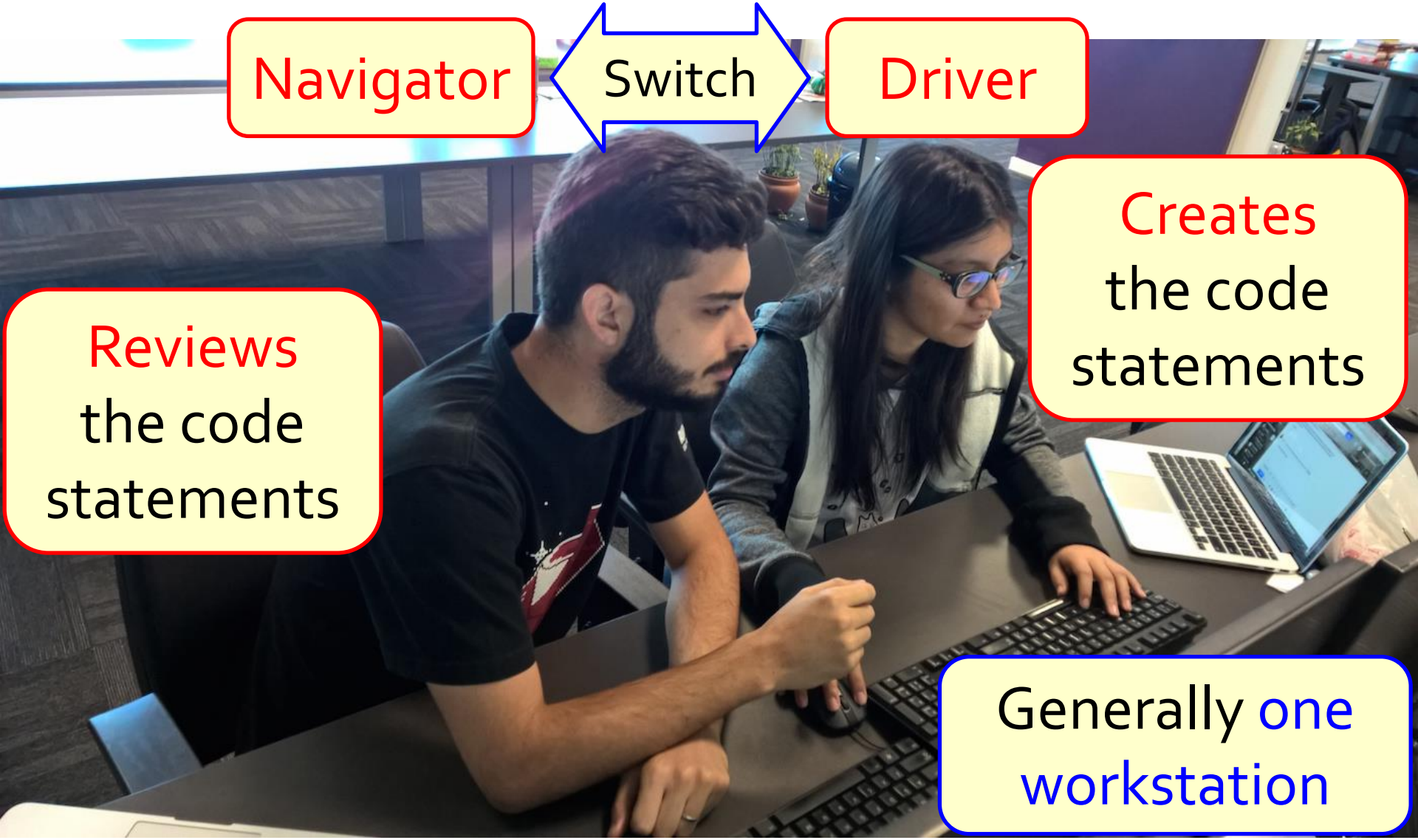
Pair Programming

(Part of XP – Extreme Programming)

Navigator Driver

http://cameracourage.com/2012/11/what-does-a-rally-co-driver-do/

Pair Programming



https://www.thoughtworks.com/insights/blog/effective-navigation-in-pair-programming
https://medium.com/@weblab_tech/pair-programming-guide-a76ca43ff389

Pair Programming

- Pair Programming really occurs as you type in the source code (even before saving the file)

- Pair of programmers continuously collaborate on:
  - analysis
  - design
  - architecture
  - algorithm
  - implementation
  - testing
  - etc.

Pair Programming: Why it Works

Pair Programming: Why it Works

- Pair (peer) pressure
  - e.g., don't want to let partner down

- Collaboration through:
  - negotiation (of design decisions, etc.) and
  - brainstorming

- Pair courage

Williams and Kessler. *Pair Programming Illuminated*. Addison Wesley. 2003

Pair Programming: Why it Works

- Pair reviews (similar to code reviews)
- Pair debugging (solves problem by explaining your code)
- Pair learning (from each other)
  - Knowledge transfer about system among team members ⇒ reduces knowledge "silos"
- Promotes Shared Code Ownership (XP)

Williams and Kessler. *Pair Programming Illuminated*. Addison Wesley. 2003

Pair Programming: Equivalent vs. Complementary Activities

- Activities that may remove the same defects

- Activities that may remove different defects

Pair Programming: Equivalent vs. Complementary Activities

- Activities that may remove the same defects
  - Code Review
  - Static Analysis

- Activities that may remove different defects
  - Beta Testing

(Remote) Pair Programming

- Each member works separately (e.g., in their own office), communicating via VoIP (e.g., Skype) and

- Editing the same shared file in an IDE

(Remote) Pair Programming Tools

eclipse ⇒ Eclipse Che

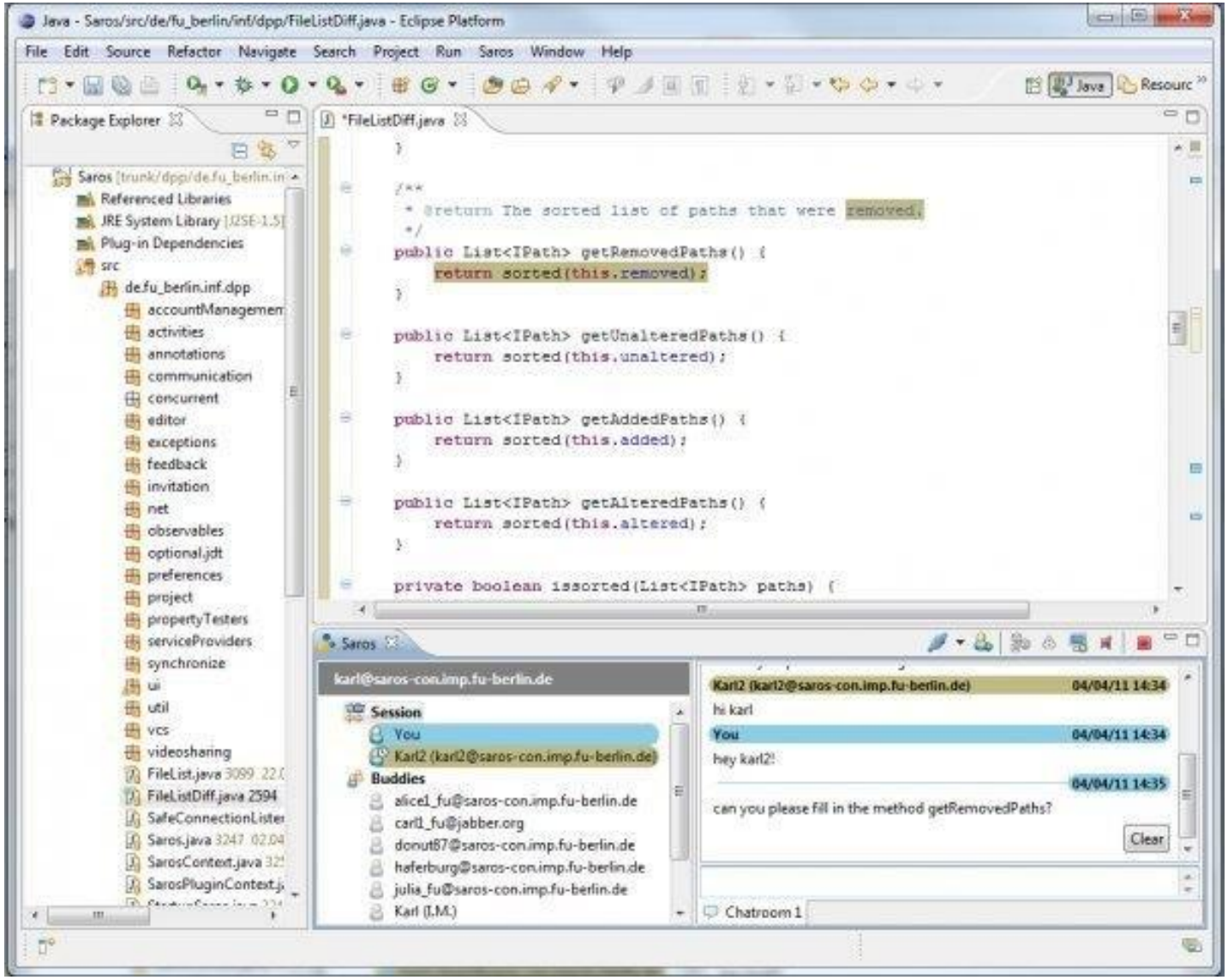
- Saros - https://sourceforge.net/projects/dpp/
- DocShare - http://wiki.eclipse.org/ECF/DocShare_Plugin

ATOM

- motepair - https://atom.io/packages/motepair

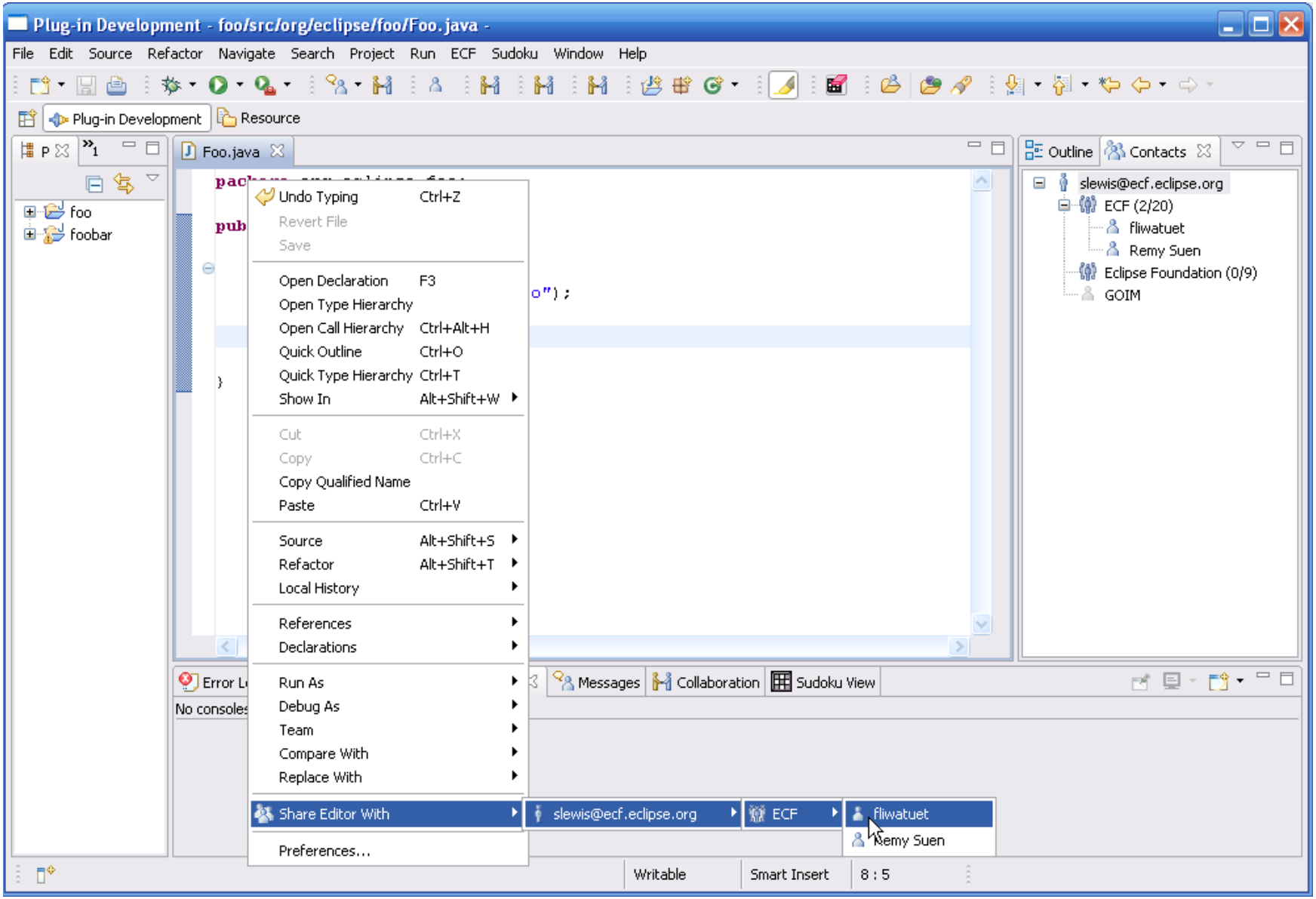
- "any" remote screen sharing/collaboration applications (e.g., TeamViewer)
  - https://www.slant.co/topics/5613/~tools-for-screen-sharing-for-remote-pair-programming

Saros



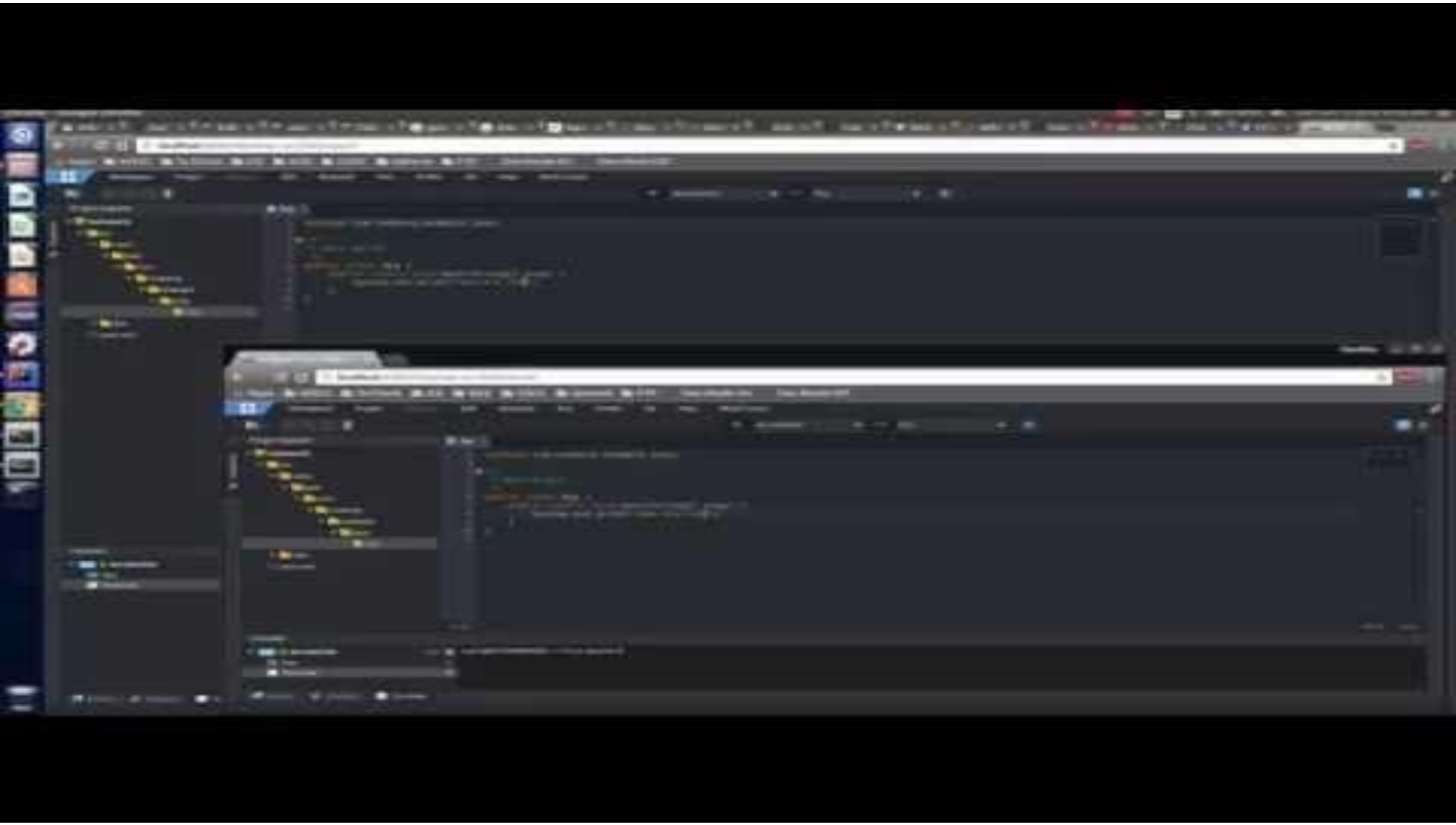
https://sourceforge.net/projects/dpp/

DocShare



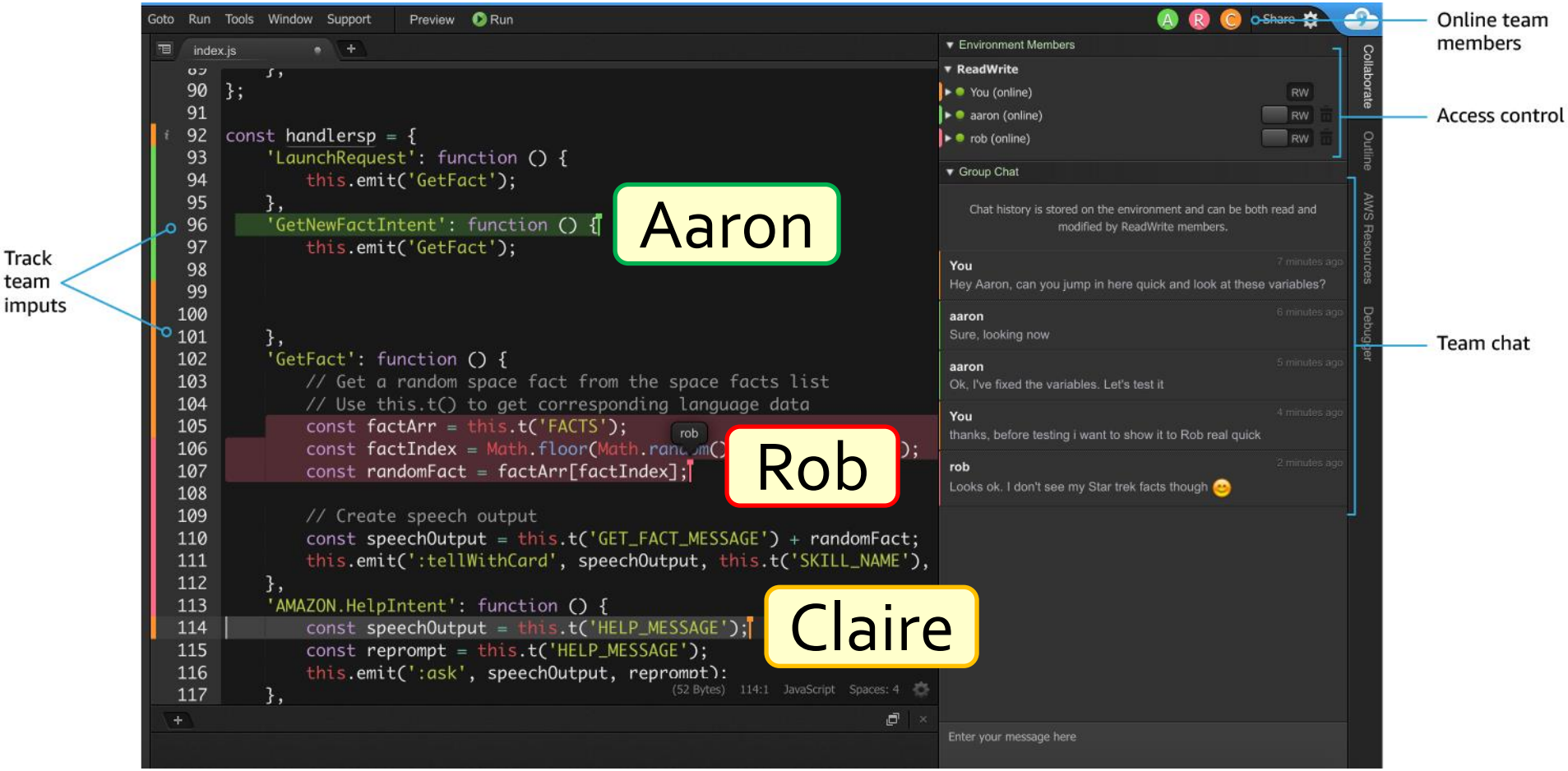
http://wiki.eclipse.org/ECF/DocShare_Plugin

Eclipse Che



Pair-programming on Eclipse Che - https://www.youtube.com/watch?v=mms4nGOYcGo

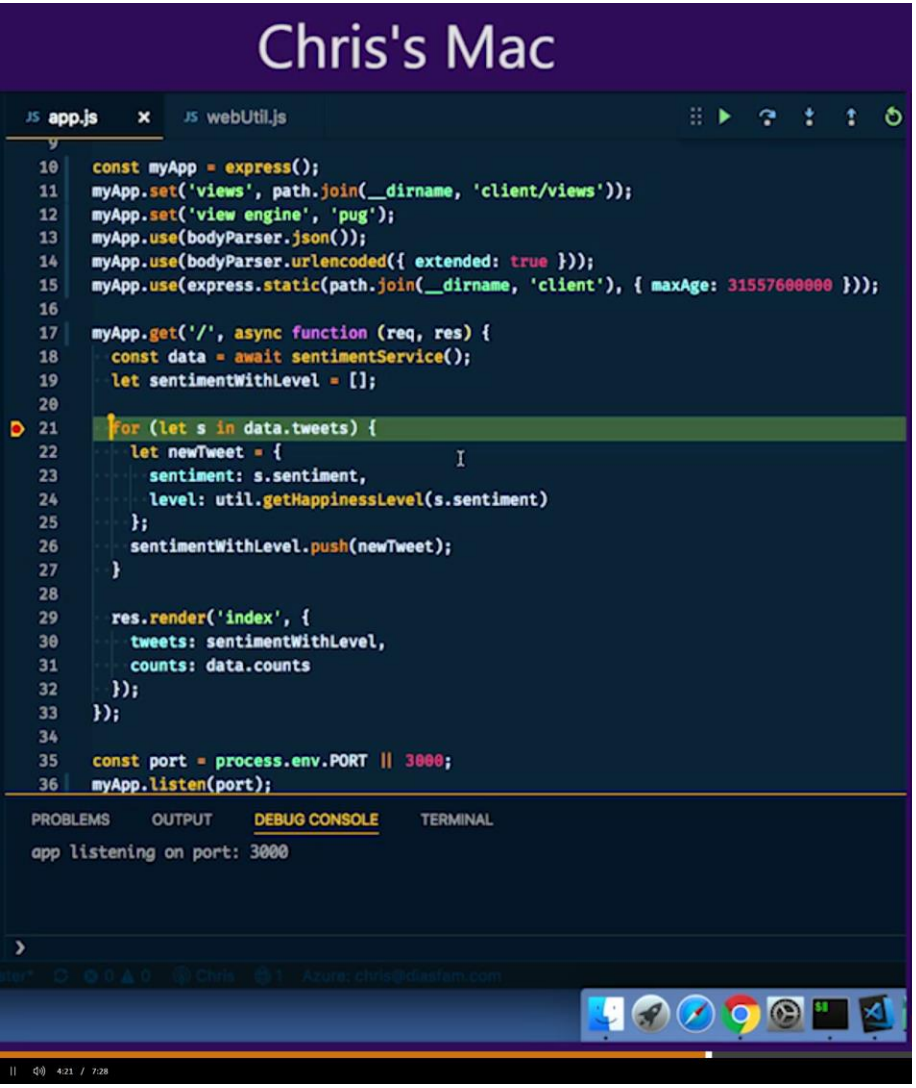
Amazon AWS Cloud9



https://aws.amazon.com/cloud9

Visual Studio Live Share

Visual Studio Code



https://code.visualstudio.com/visual-studio-live-share
https://code.visualstudio.com/blogs/2017/11/15/live-share

Visual Studio Live Share

Visual Studio Code

Visual Studio

Chris's Mac

Chris started debugging session

```
for (let s in data.tweets) {
  let newTweet = {
    sentiment: s.sentiment,
    level: util.getHappinessLevel(s.sentiment)
  };
  sentimentWithLevel.push(newTweet);
}

res.render('index', {
  tweets: sentimentWithLevel,
  counts: data.counts
});
});

const port = process.env.PORT || 3000;
myApp.listen(port);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

app listening on port: 3000

Amanda's PC

Amanda accesses Chris's debug session

```
myApp.set('views', path.join(__dirname, 'client/views'));
myApp.set('view engine', 'pug');
myApp.use(bodyParser.json());
myApp.use(bodyParser.urlencoded({ extended: true }));
myApp.use(express.static(path.join(__dirname, 'client'), { maxAge: 31557600000 }));

myApp.get('/', async function (req, res) {
  const data = await sentimentService();
  let sentimentWithLevel = [];

  for (let s in data.tweets) {
    let newTweet = {
      sentiment: s.sentiment,
      level: util.getHappinessLevel(s.sentiment)
    };
```

Chris Dias

s undefined

Locals

Name	Value	Type
constructor	function Object() { ... }	Function
hasOwnProperty	function hasOwnProperty() { ... }	Function
isPrototypeOf	function isPrototypeOf() { ... }	Function
propertyIsEnumeral	function propertyIsEnumerable() { ... }	Function
toLocaleString	function toLocaleString() { ... }	Function
toString	function toString() { ... }	Function

Call Stack

Name

(anonymous fu

[PROMISE]

init

emitInitNative

(anonymous fu

handle

next

Autos Locals Watch 1

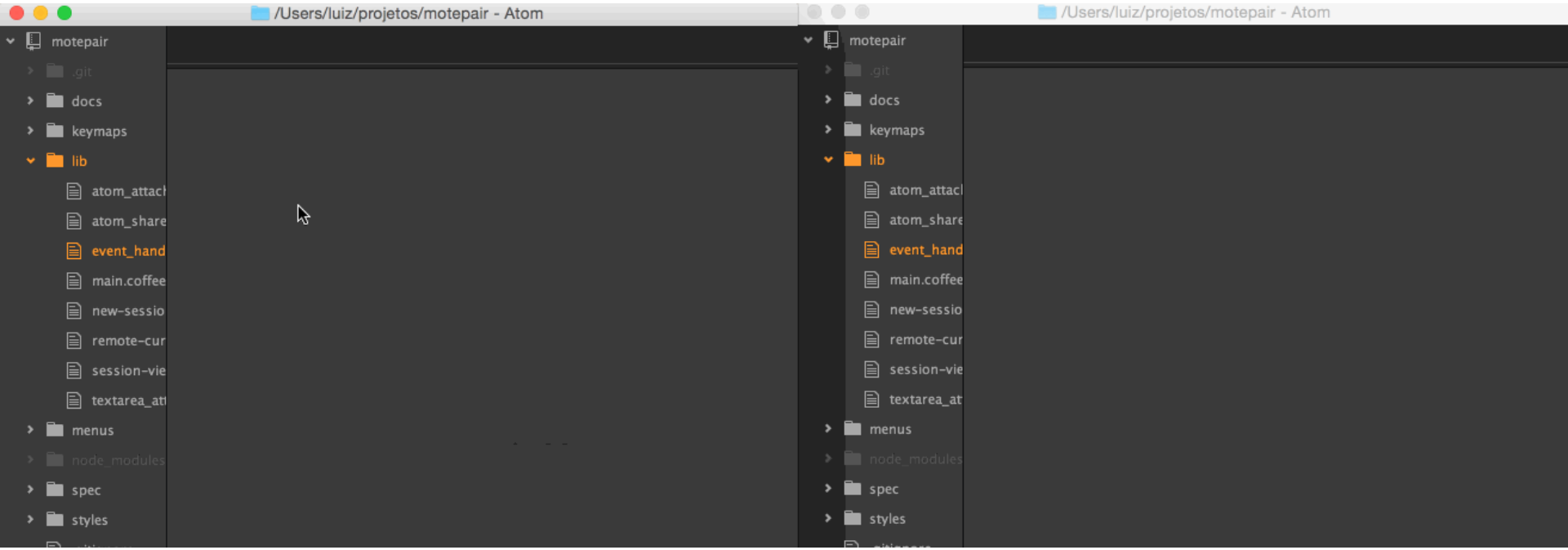
Call Stack

Ready

https://code.visualstudio.com/visual-studio-live-share

https://code.visualstudio.com/blogs/2017/11/15/live-share

motepair – ATOM Package to Remote Pair Programming



What do you think about

Remote Pair Programming

vs.

Traditional Pair Programming?

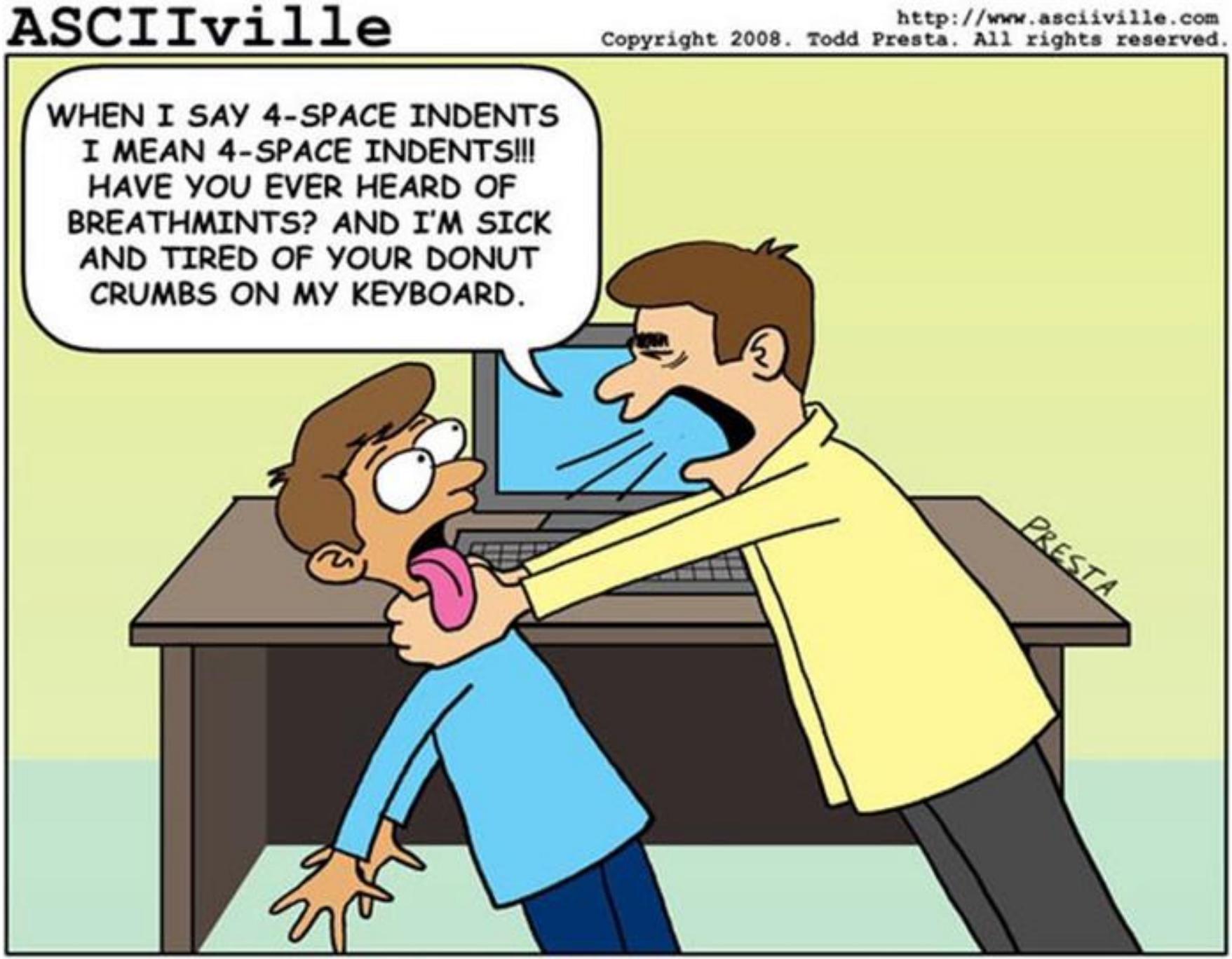
https://atom.io/packages/motepair

(Remote) Pair Programming

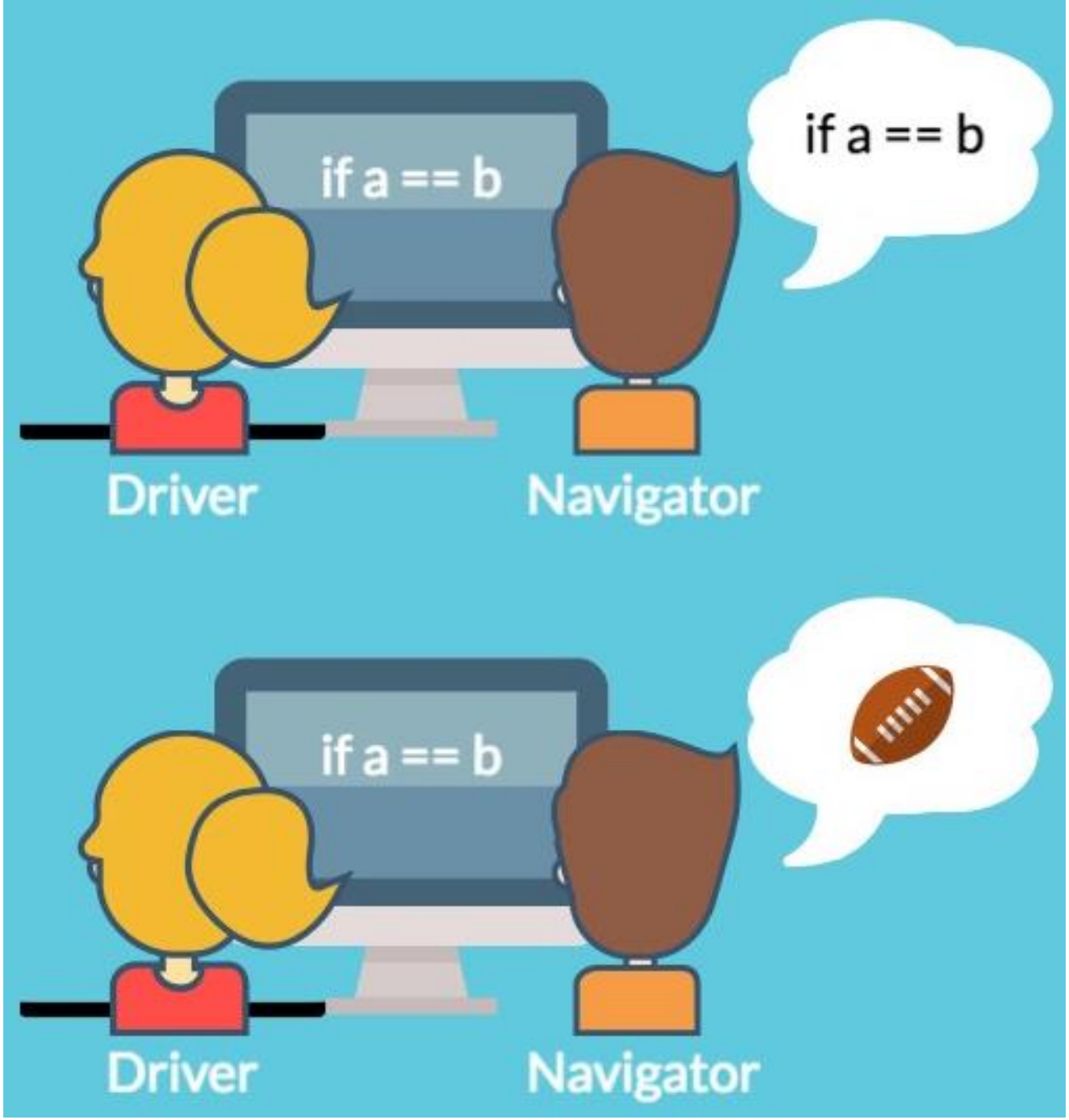
- Not as wide-spread as traditional Pair Programming

- Can be more challenging
  - same reason as communicating remotely (e.g., via email, phone, etc.) is not as efficient as communicating in person

However, even when Pair-Programming in person…



The dark side of pair programming.



https://medium.com/@weblab_tech/pair-programming-guide-a76ca43ff389

Other Factors Need to be Considered…

- behavioral
- psychological
- social
- logistical
  - types of pairings (novice vs. senior)
  - pair rotations (ad-hoc, per story, per sprint, knowledge matrix, etc.)
  - switching driver ⇔ navigator roles…

"Bus factor"

Will a project "fail" (e.g., development is severely hindered) if one of its members is "hit by a bus" (promoted, left company, etc.)?

https://en.wikipedia.org/wiki/Bus_factor

"Bus factor"

Will a project "fail" (e.g., development is severely hindered) if one of its members is "hit by a bus" (promoted, left company, etc.)?

"Bus factor" = metric that counts # of developers:

- with indispensable (siloed) knowledge
- who are a single point of failure to the project

Solution ⇒ Promote Shared Code Ownership (XP)

https://en.wikipedia.org/wiki/Bus_factor

Knowledge Silo Matrix (KSM)

Name	Priority:	Silo 1	Silo 2	Silo 3	Silo 4	Silo 5	Silo 6
		1	4	3	2	4	1
Person 1							
Person 2							
Person 3							
Person 4							
Person 5							
Person 6							
Person 7							
Person 8							
Person 9							
Person 10							

- Expert to be replicated
- Consistent with expert
- Actively learning
- Not working in that silo
- Risk

http://stevetrautman.com/3-step-solution/

Knowledge Silo Matrix (KSM)

Name	Priority:	Silo 1	Silo 2	Silo 3	Silo 4	Silo 5	Silo 6
		1	4	3	2	4	1
Person 1							
Person 2							
Person 3							
Person 4							
Person 5							
Person 6							
Person 7							
Person 8							
Person 9							
Person 10							

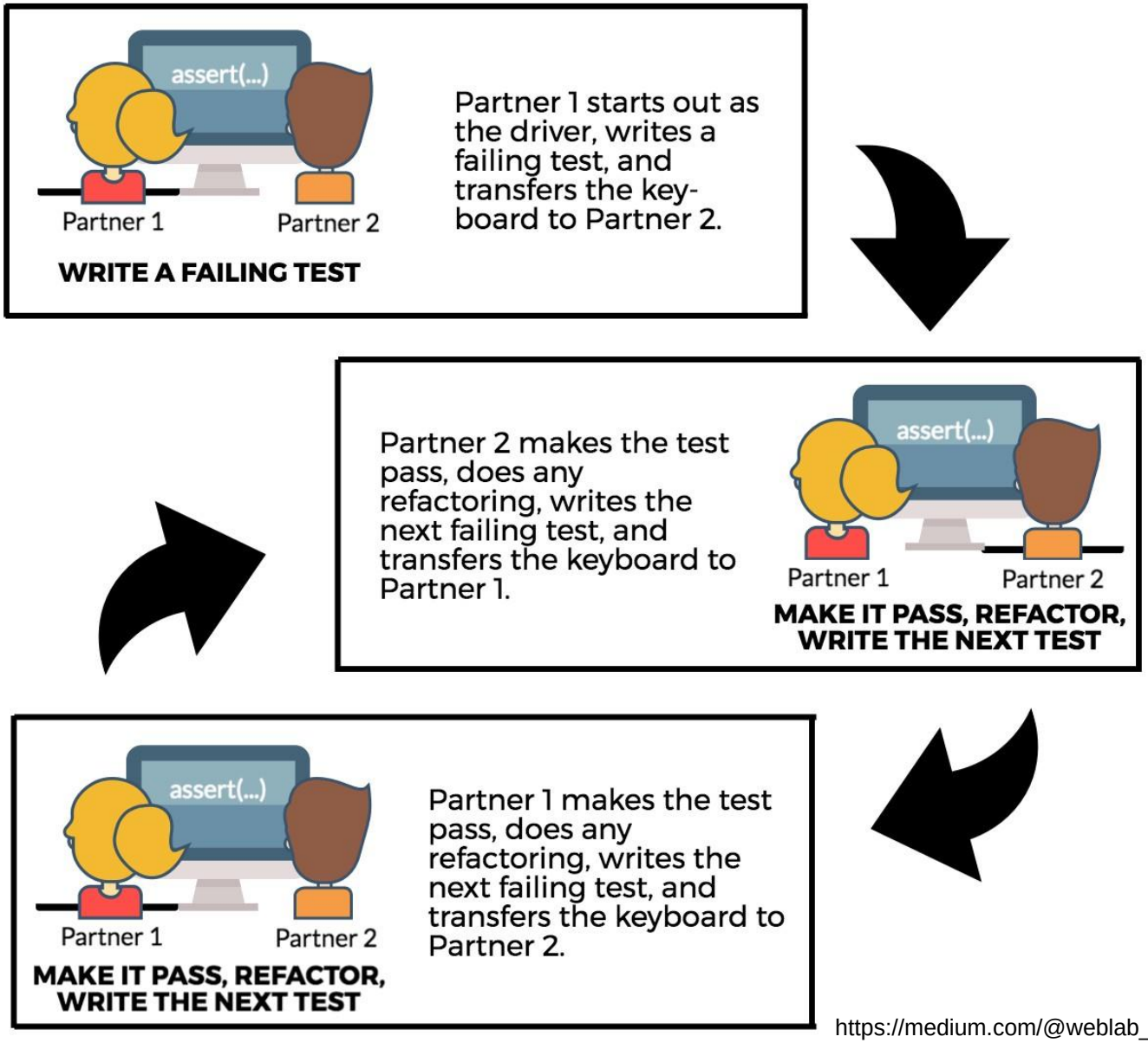
- Expert to be replicated
- Consistent with expert
- Actively learning
- Not working in that silo
- Risk

Deliberately assign pairs based on knowledge "gaps" in the system

http://stevetrautman.com/3-step-solution/

"PING PONG" PAIRING

In "ping pong" pairing, the "write a failing test", "make it pass", "refactor" loop of Test-Driven Development is used.



Useful to switch roles during Test Driven Development (TDD)

https://medium.com/@weblab_tech/pair-programming-guide-a76ca43ff389

Pair Programming: Measuring Defect Removal Effectiveness

- How can we measure the effectiveness?

- We could start with a simple experiment:
  - https://blog.elpassion.com/the-pair-programming-experiment-part-i-334d93524944
  - https://blog.elpassion.com/the-pair-programming-experiment-part-iii-the-results-176014ee24a

Pair Programming: Measuring Defect Removal Effectiveness

- Direct Measurement:
  - Observer tallies removed defects
  - CS481 Students: 28%

- Indirect Measurement:
  - Measure the reduction in down-stream defect density in a large code population before/after introducing Pair Programming
  - Williams et al.: 15%

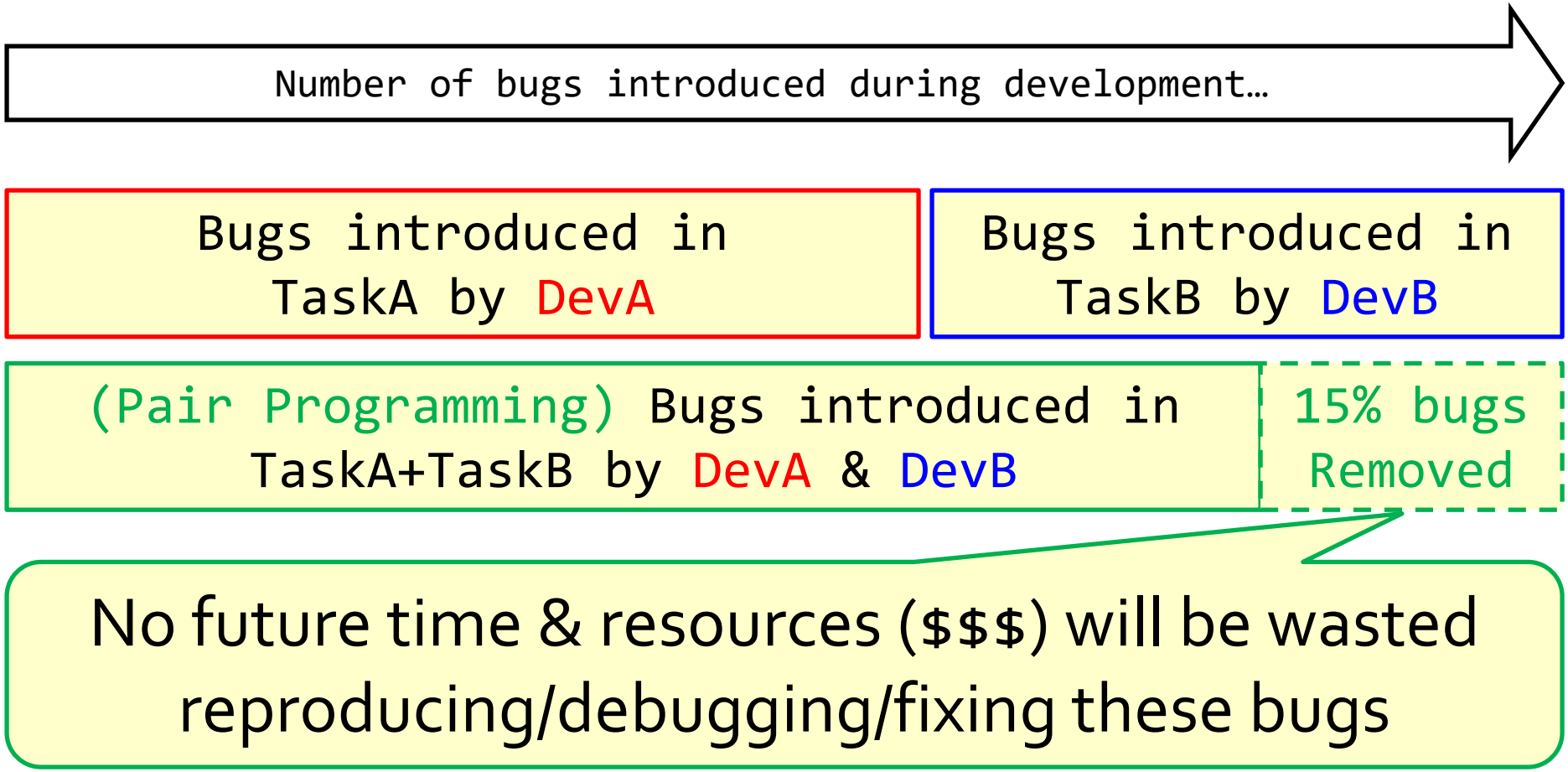
Pair Programming: Advantages

Pair Programming: Advantages

- Finds defects promptly, shortly after their introduction
- Finds defects cheaply, see the root cause right before your eyes without investing hours to reproduce/debug the problem
- Evidence of enhanced team morale, teamwork, learning and mentoring
- Shared-code ownership (XP)

Pair Programming: Advantages

- Improves quality by removing ~15% of defects*



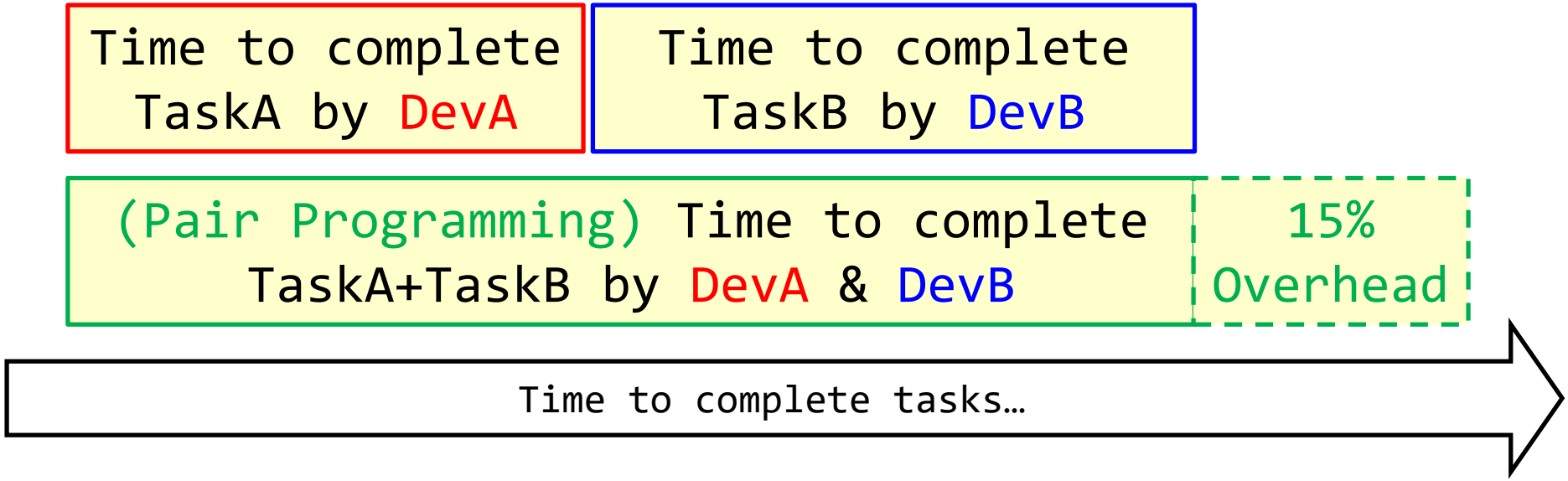
* Williams et al.

Pair Programming: Disadvantages

- Potential personality (ego) issues
- "Soft skills" issues
- Noise in the office (if everyone is pair-programming)

Pair Programming: Disadvantages

- Increased effort/overhead: 15%*
  - Note that the amount of time needed to complete the task does not double!



* (Williams and Kessler 2000)

Unit-Level Testing

Unit-Level Testing: Description

- Exercises the smallest "unit" (e.g., a class or static method) of product functionality
- Ideally, each unit-level test case is independent of all others
- Test case should:
  - Setup the initial state (i.e., context)
  - Exercise the target functionality
  - Confirm the expected result (may include a state change within the object under test)
- Strong support from the IDEs (e.g., JUnit, NUnit)

Unit-Level Testing: Measuring Defect Removal Effectiveness

- Direct Measurement:
  - Tally the removed defects
  - Published Results: 15..50% (Jones96)

- Mutation Testing

Mutation Testing (Mutation Analysis)

- Mutation Testing
  - injects defects (known as mutants) into the product code

Mutation Testing (Mutation Analysis)

- Mutation Testing
  - injects defects (known as mutants) into the product code

PIT*

*PIT (http://pitest.org/) for Java (see HW5), but available for many programming languages

Mutation Testing (Mutation Analysis)

- Mutation Testing
  - injects defects (known as mutants) into the product code

```
if (a && b) {
    c = 1;
} else {
    c = 0;
}
```

Java Bytecode
Before mutation

PIT* ⇨

```
if (a || b) {
    c = 1;
} else {
    c = 0;
}
```

Java Bytecode
After mutation

*PIT (http://pitest.org/) for Java (see HW5), but available for many programming languages

Mutation Testing (Mutation Analysis)

- Mutation Testing
  - injects defects (known as mutants) into the product code

```
if (a && b) {
    c = 1;
} else {
    c = 0;
}
```

PIT* ⇨

```
if (a || b) {
    c = 1;
} else {
    c = 0;
}
```

Types of defects that can be injected:

- Replace operators
- Delete/duplicate statements
- Replace values (e.g., numerical, Boolean, String)
- etc.

*PIT (http://pitest.org/) for Java (see HW5), but available for many programming languages

Mutation Testing (Mutation Analysis)

- Mutation Testing
  - injects defects (known as mutants) into the product code
  - executes the tests
  - determines the fraction of the mutants found by the tests

- Infers test effectiveness for finding real defects from its effectiveness for finding mutants

Mutation Testing (Mutation Analysis)

- It is a stochastic process
  - What does it mean?

Mutation Testing (Mutation Analysis)

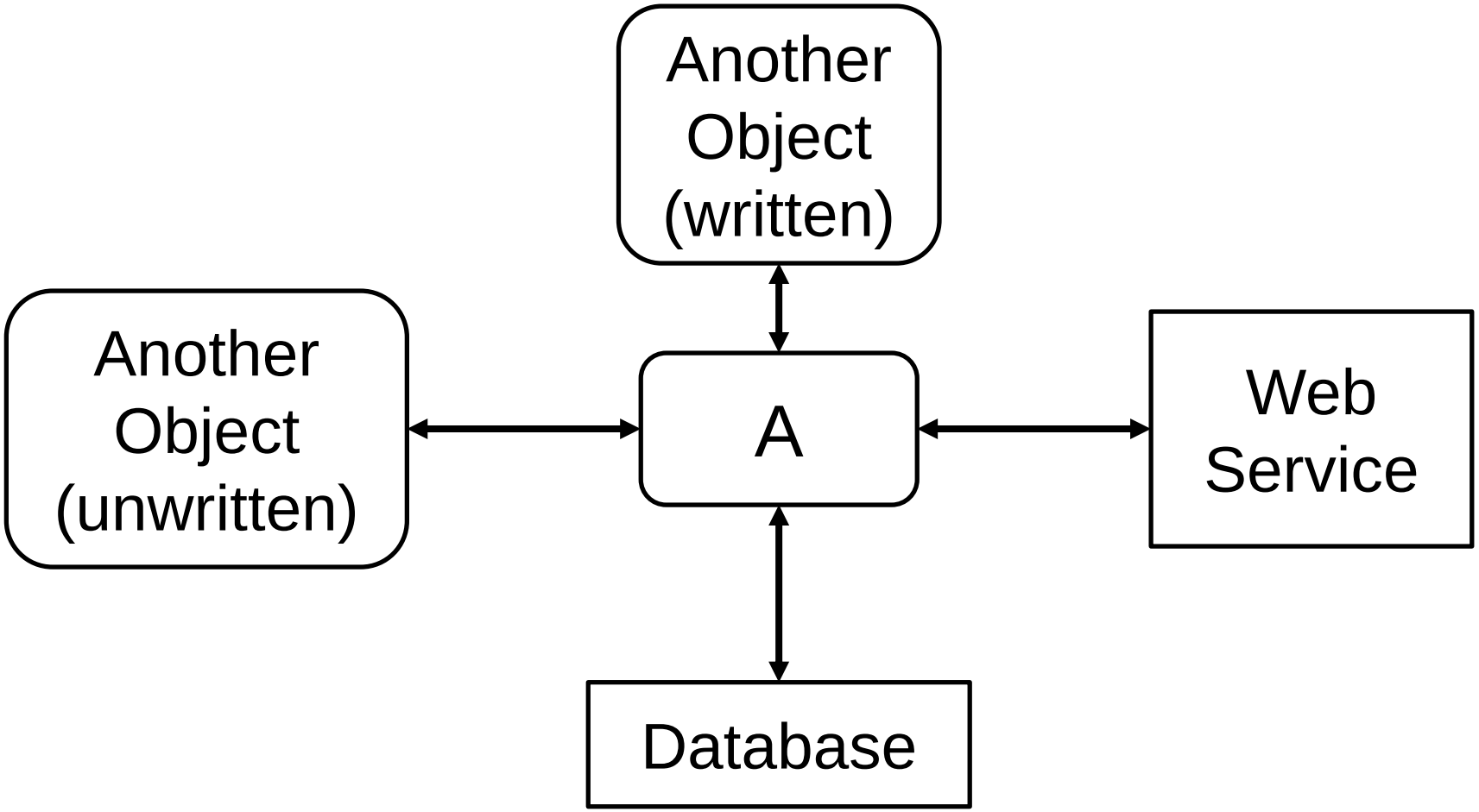
- It is a stochastic (random) process
  - different runs may inject different bugs ⇒ generate different test effectiveness
  - Average test effectiveness over multiple runs

Mutation Testing Terminology

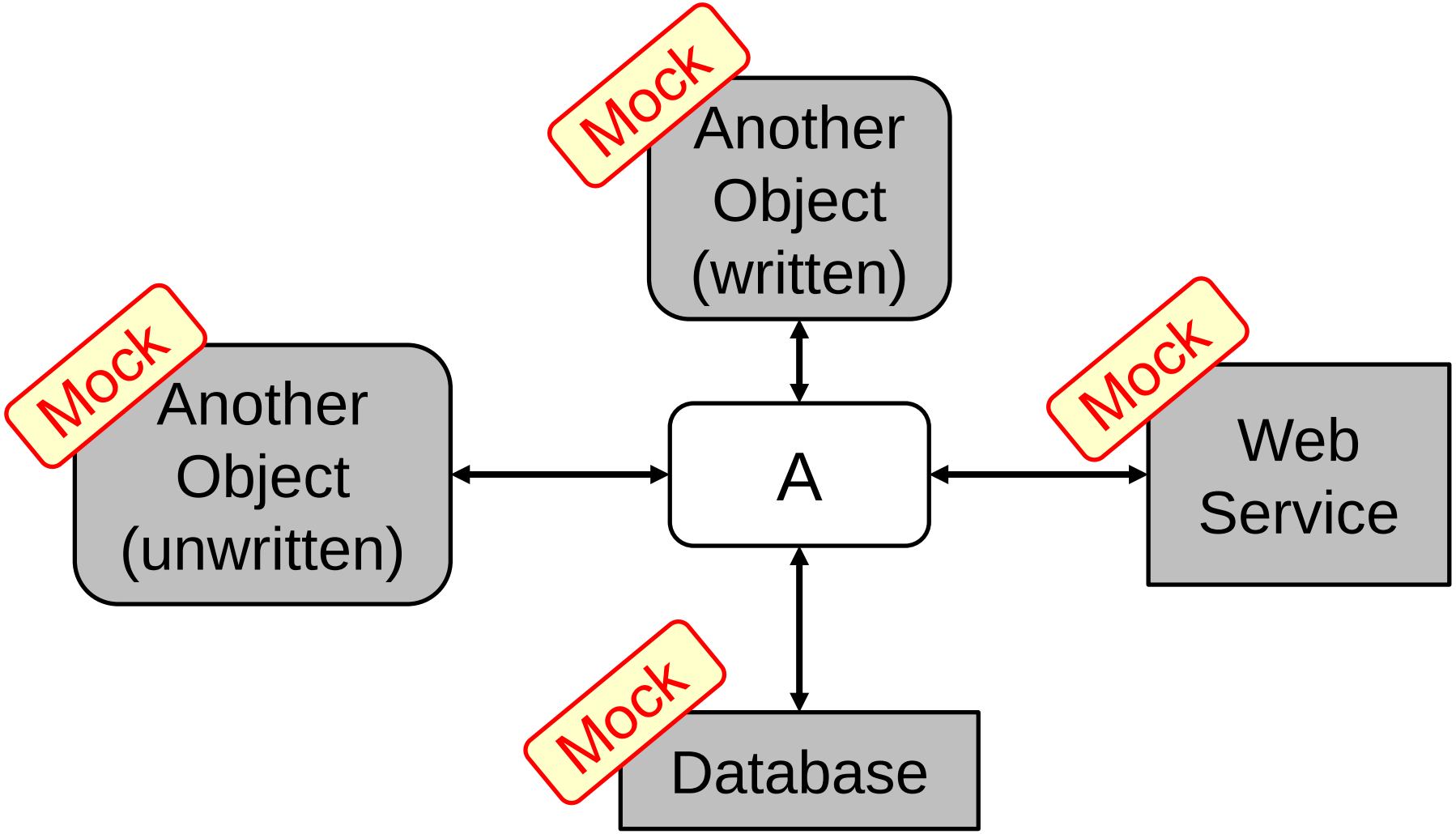
- Mutant: Defect randomly injected into a copy of the product code
- Killed Mutant: A mutant found by the test under evaluation
- Live Mutant: A mutant that escaped detection by the test
- Mutation Score: The fraction of mutants discovered (killed) by the test

How Would we Write a Unit Test for an Object Requiring Complex Interactions?

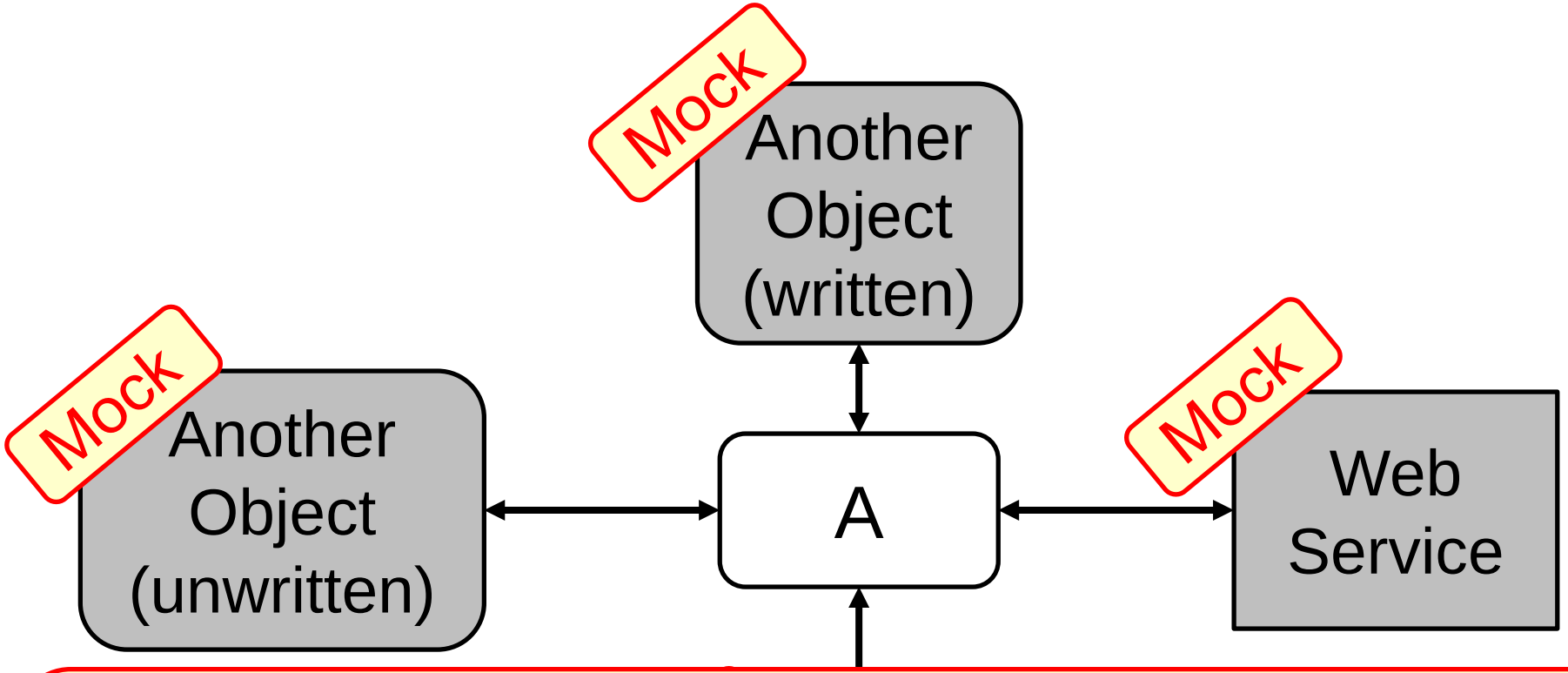
Example of Expected Interaction of A. How Would we Write Unit Tests for A?



When Testing A, all Collaborators are Replaced with Mocks



When Testing A, all Collaborators are Replaced with Mocks



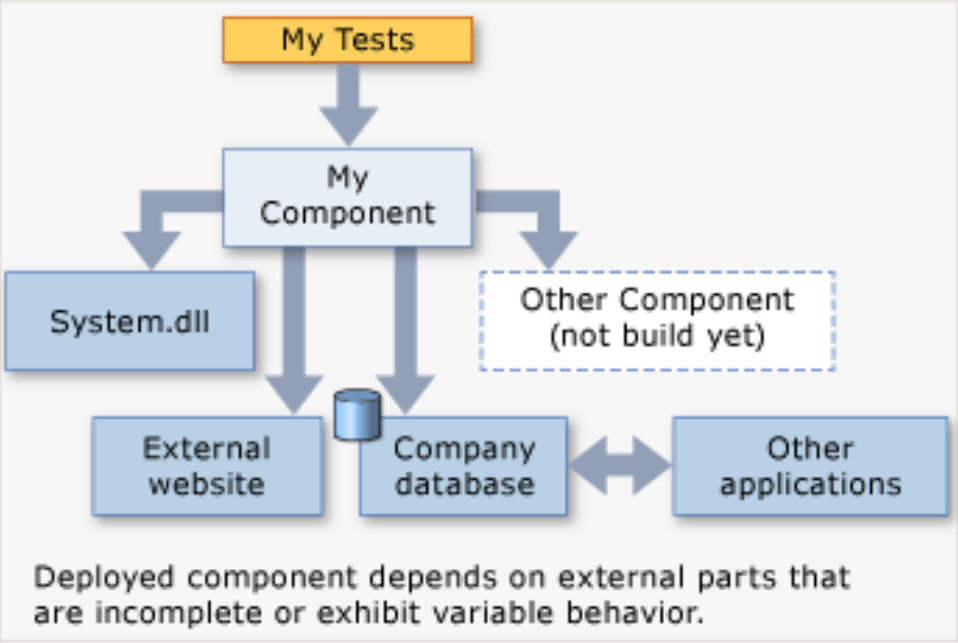
Mocks = "objects pre-programmed with expectations which form a specification of the calls they are expected to receive"

– Martin Fowler

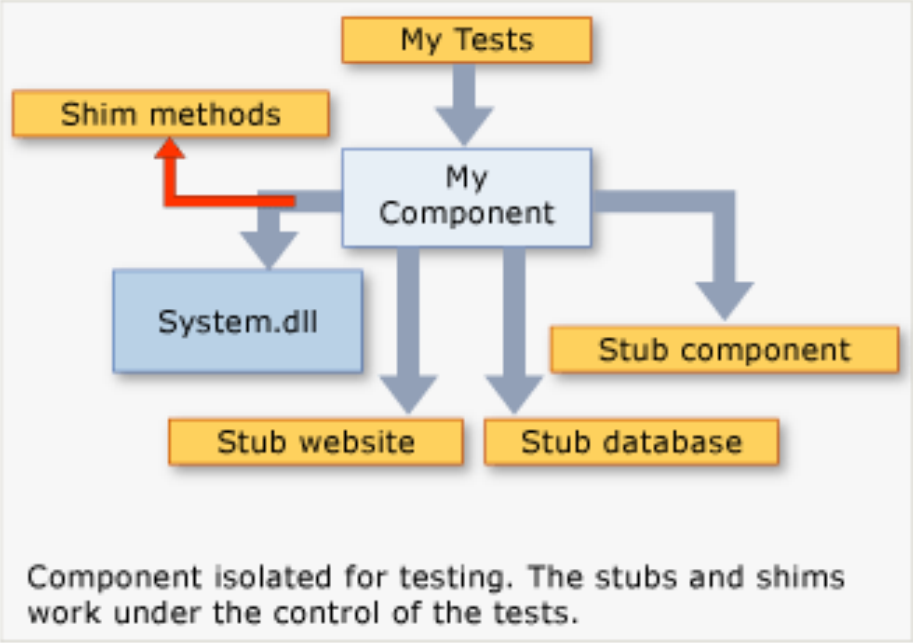
Test Doubles vs. Mocks

- A Test Double is a simplified implementation of a complex class, and includes:
  - dummy
  - fake
  - stubs
  - spies
  - mocks
  - double
  - https://martinfowler.com/articles/mocksArentStubs.html#TheDifferenceBetweenMocksAndStubs

Microsoft Fakes



Deployed component depends on external parts that are incomplete or exhibit variable behavior.



Component isolated for testing. The stubs and shims work under the control of the tests.

https://docs.microsoft.com/en-us/visualstudio/test/isolating-code-under-test-with-microsoft-fakes

Reasons for using Mocks

- The functionality of the class under test has not been written yet (TDD)

- Context of classes being called is difficult to setup
  - Require multiple classes
  - Objects being called has UI / requires human interaction
  - Require external resources: file system, database, network, etc.
  - Has non-deterministic (stochastic) behavior

Example Mock

```java
@Test
public void voteForRemovals() {

  IMocksControl ctrl = support.createControl();
  collaborator = ctrl.createMock(Collaborator.class);
  classUnderTest.setListener(collaborator);

  collaborator.documentAdded("Document 1");

  expect(collaborator.voteForRemovals("Document 1")).andReturn((byte) 20);

  collaborator.documentRemoved("Document 1");

  support.replayAll();

  classUnderTest.addDocument("Document 1", "content");
  assertTrue(classUnderTest.removeDocuments("Document 1"));

  support.verifyAll();
}
```

http://easymock.org/user-guide.html

Mocking Frameworks

- http://easymock.org/
- http://www.jmock.org/
- http://site.mockito.org/
- https://github.com/moq/moq4
- https://pypi.python.org/pypi/mock
- https://docs.microsoft.com/en-us/visualstudio/test/isolating-code-under-test-with-microsoft-fakes

Unit-Level Test: Advantages

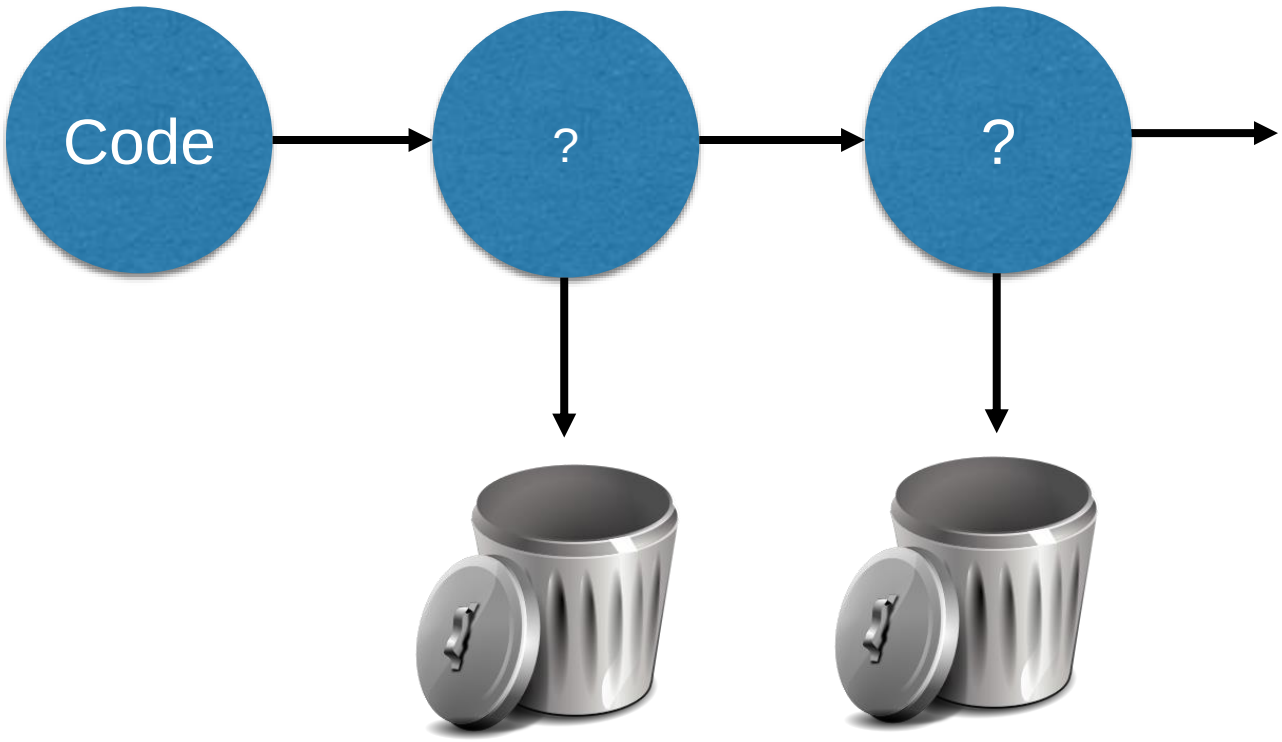
- Easily automated
- Initial Advantages
  - Finds simple, localized coding errors inside a class
  - Easy to debug
- Long-Term Advantages
  - Executable specification for your class
  - Protects your class from blunders introduced by developers unfamiliar with its operation
- In many projects, the long-term advantages outweigh the initial advantages

Unit-Level Test: Disadvantages

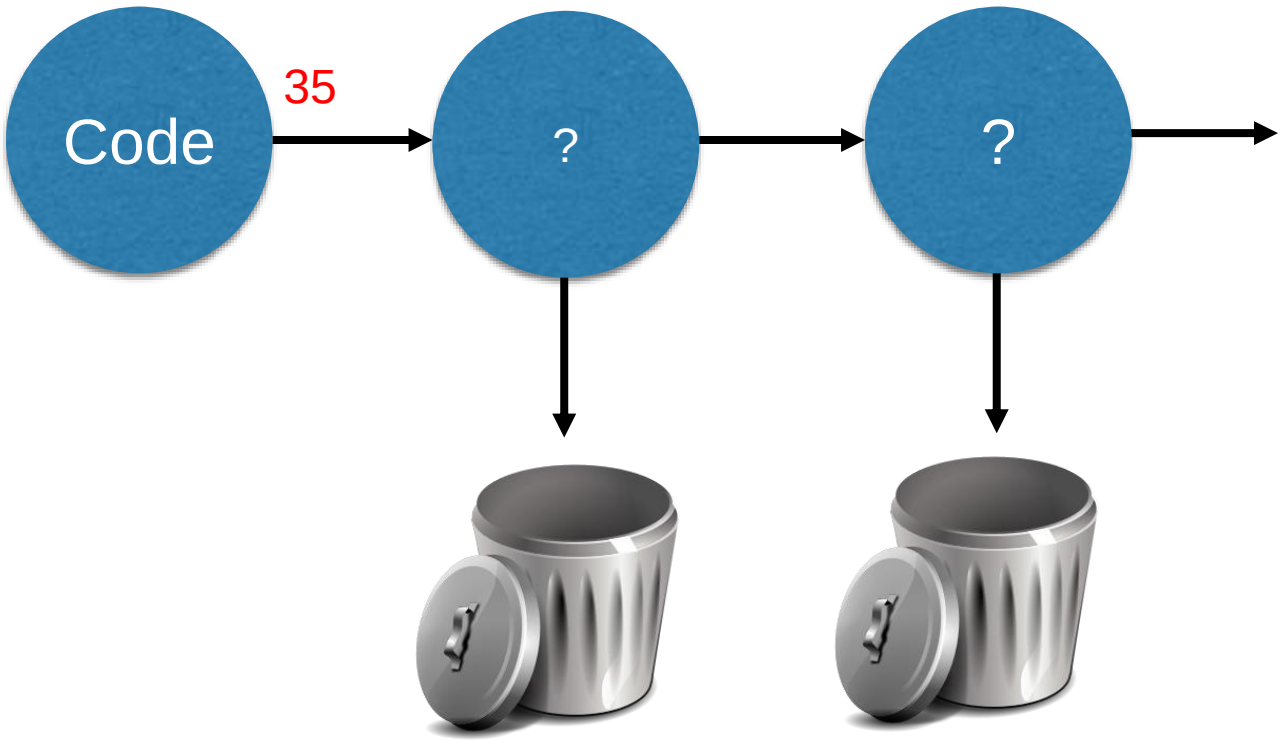
- Test and product code are written by the same developer (or pair), effectively limiting their thoroughness to their skill
  - (similar to a Code Review)
- May require custom tooling (e.g., test doubles) to facilitate true unit-level testing of many classes
- Unit-level testing of some classes (e.g., GUI code) may not be economically desirable
  - e.g., Java GUI Testing Framework http://abbot.sourceforge.net/

Modeling Pair Programming and Unit Testing as an Enhancement to Coding

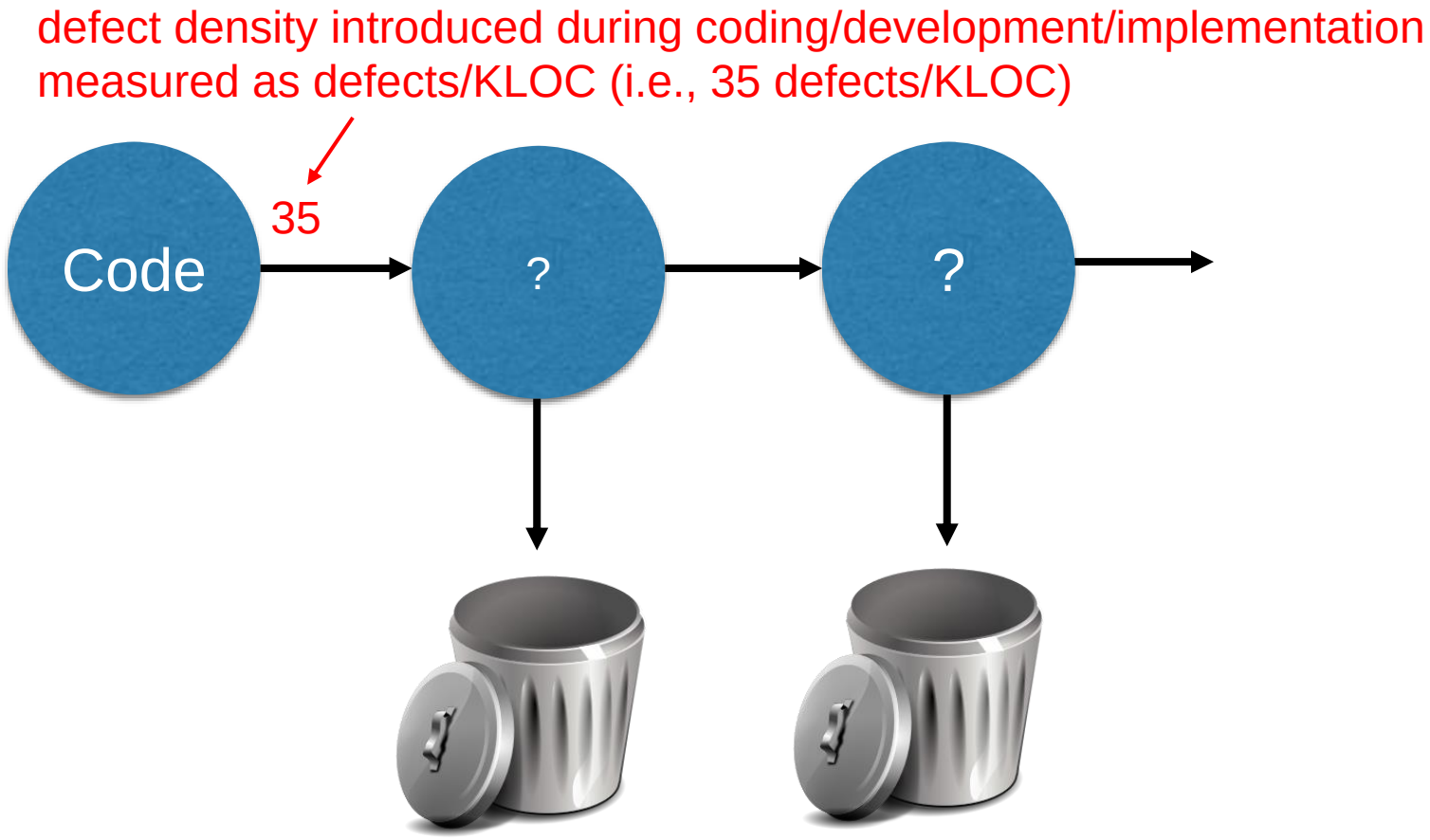
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



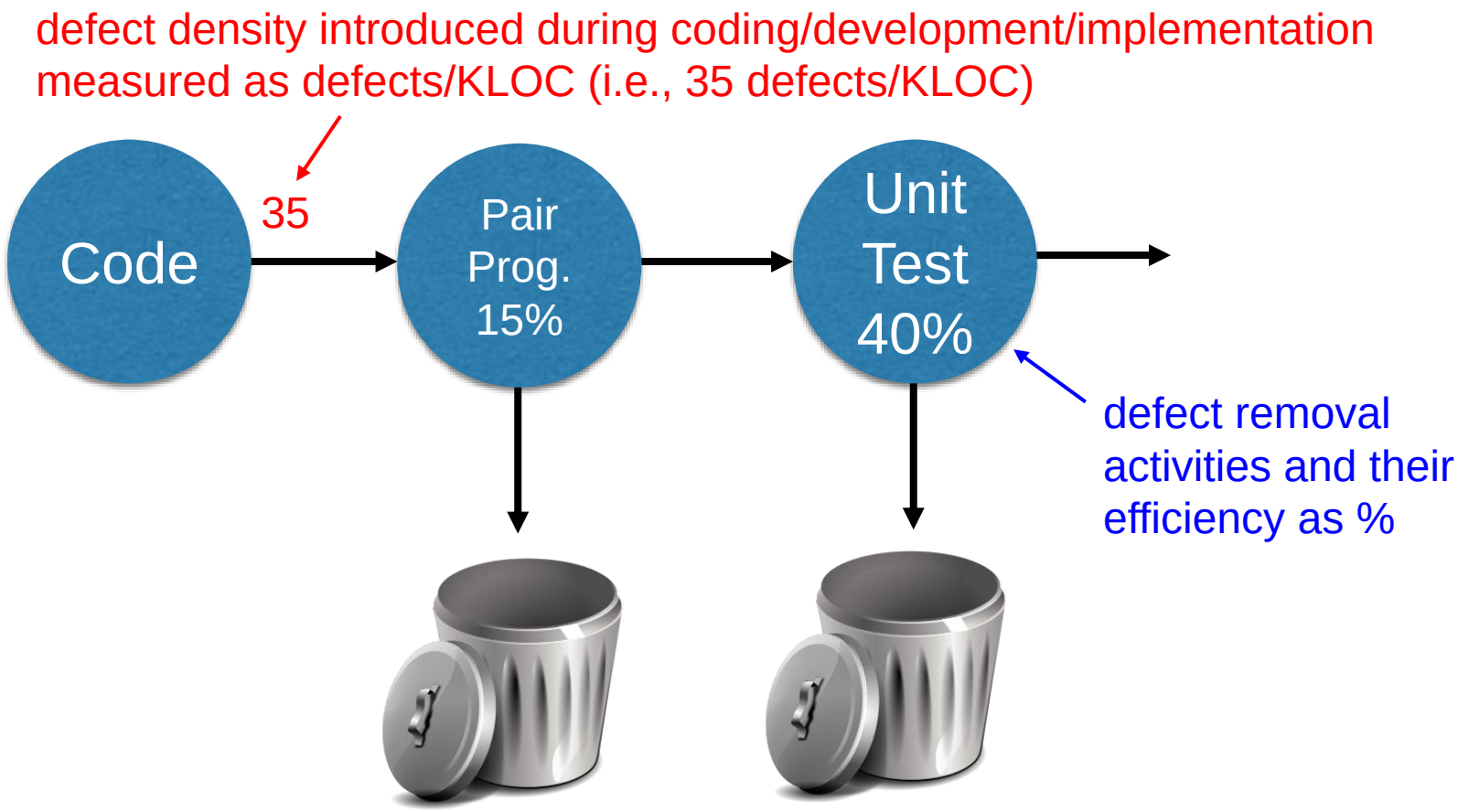
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



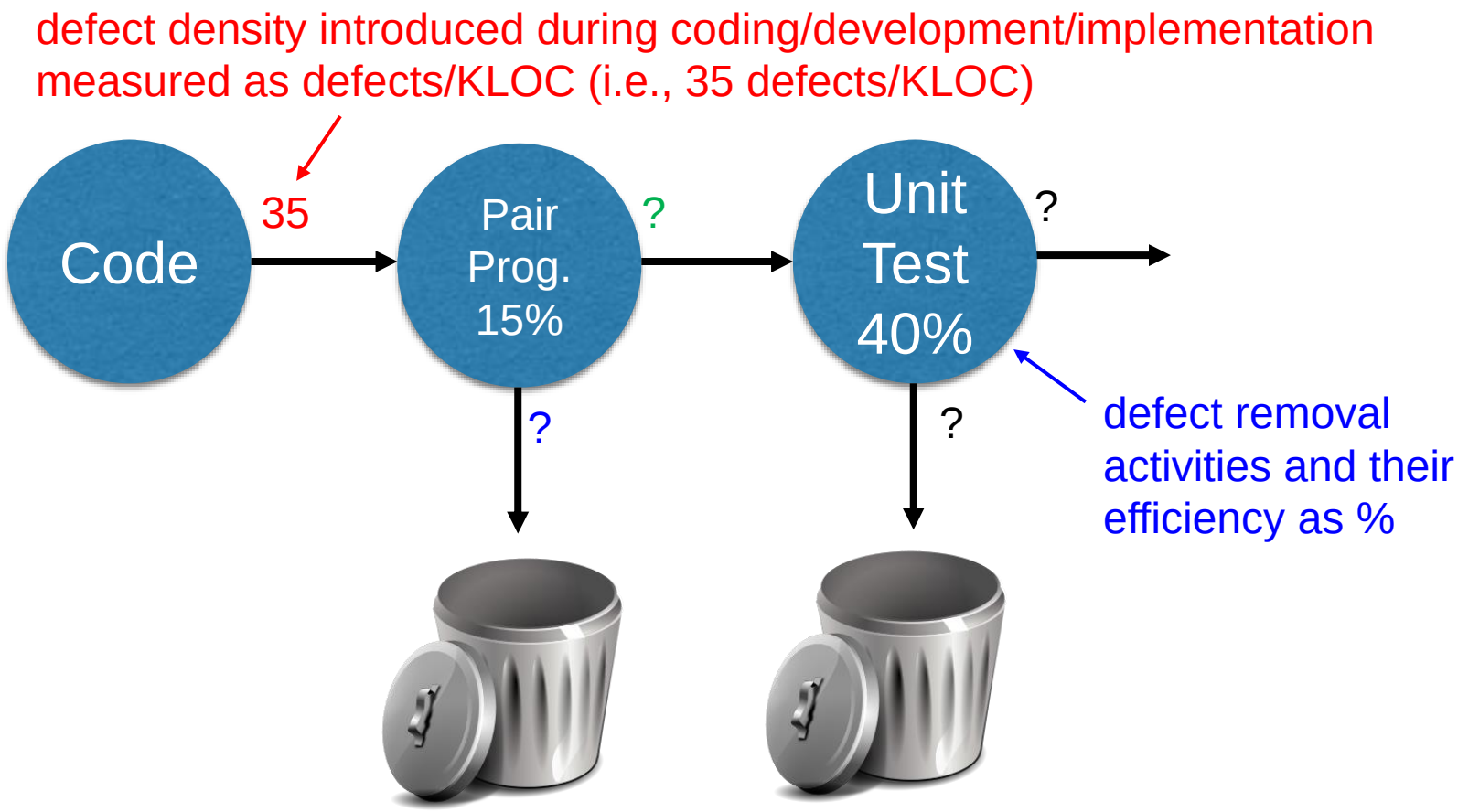
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



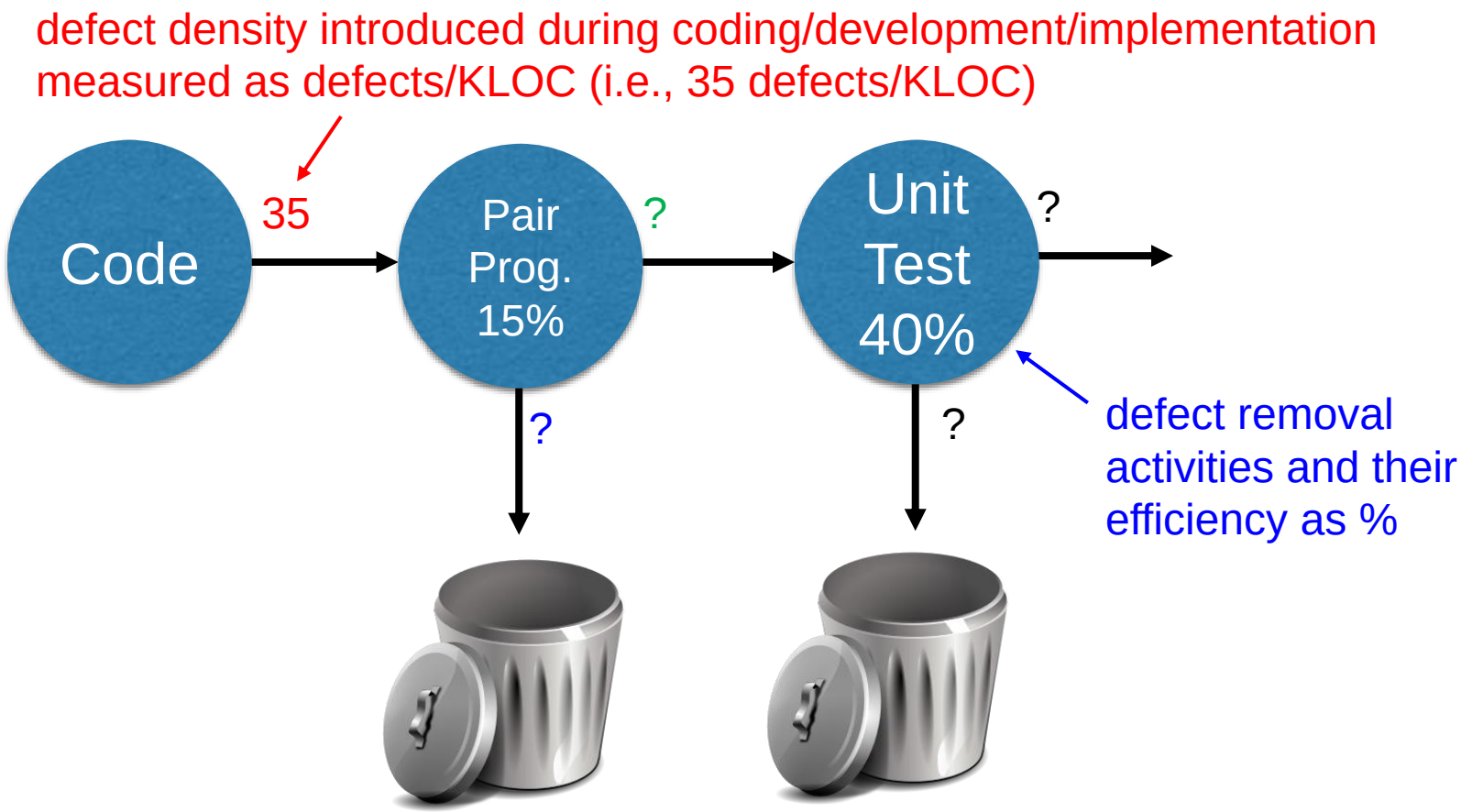
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



Modeling Pair Programming and Unit Testing as an Enhancement to Coding



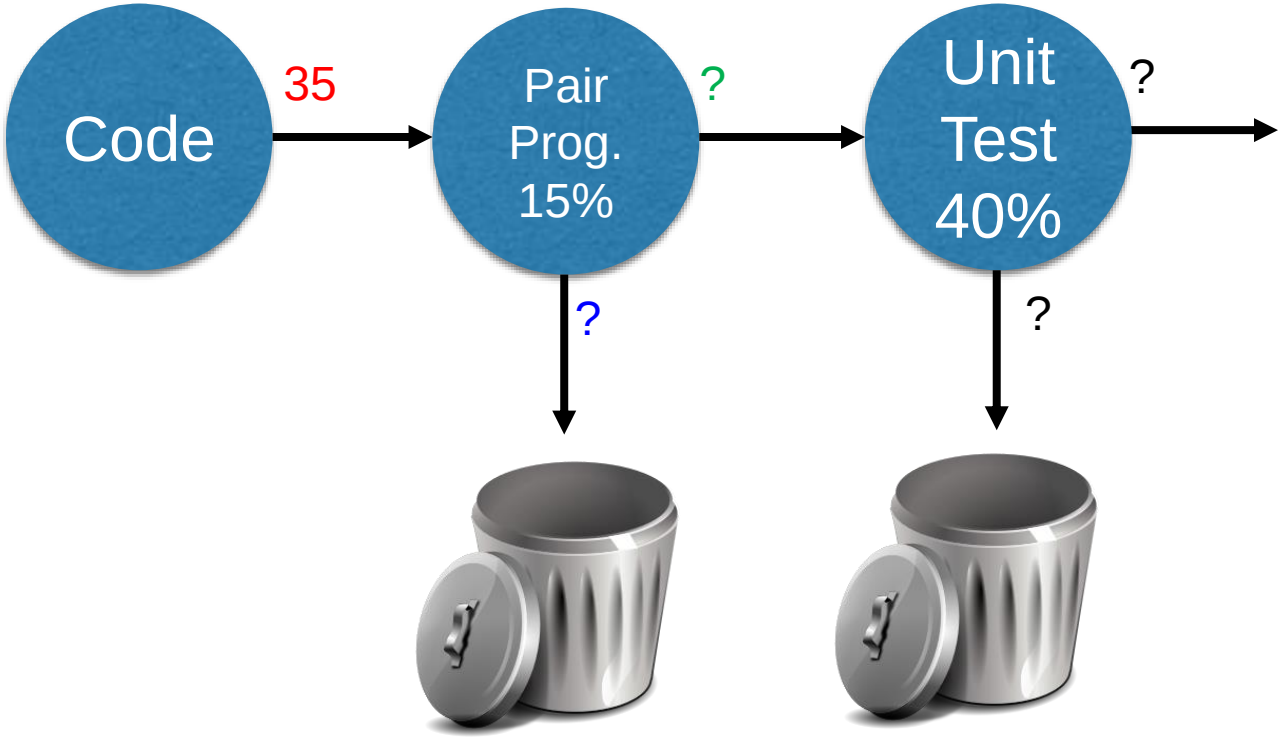
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

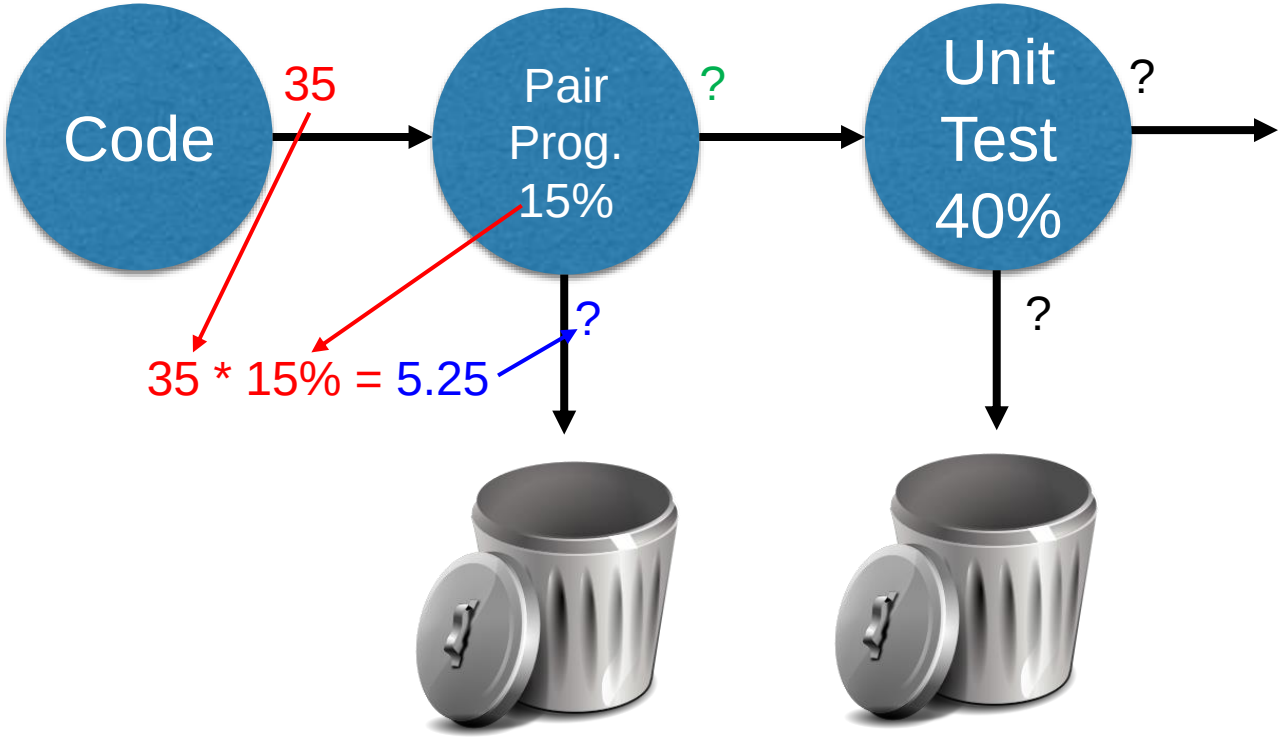
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

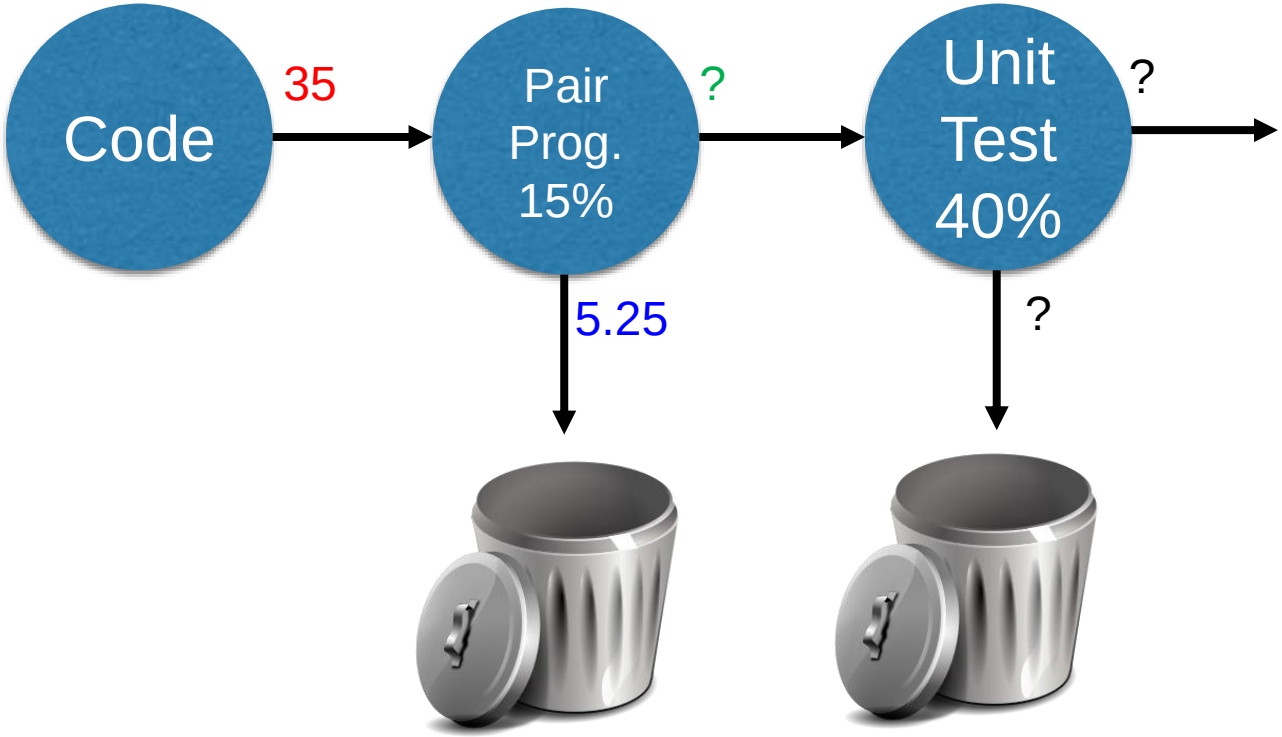
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

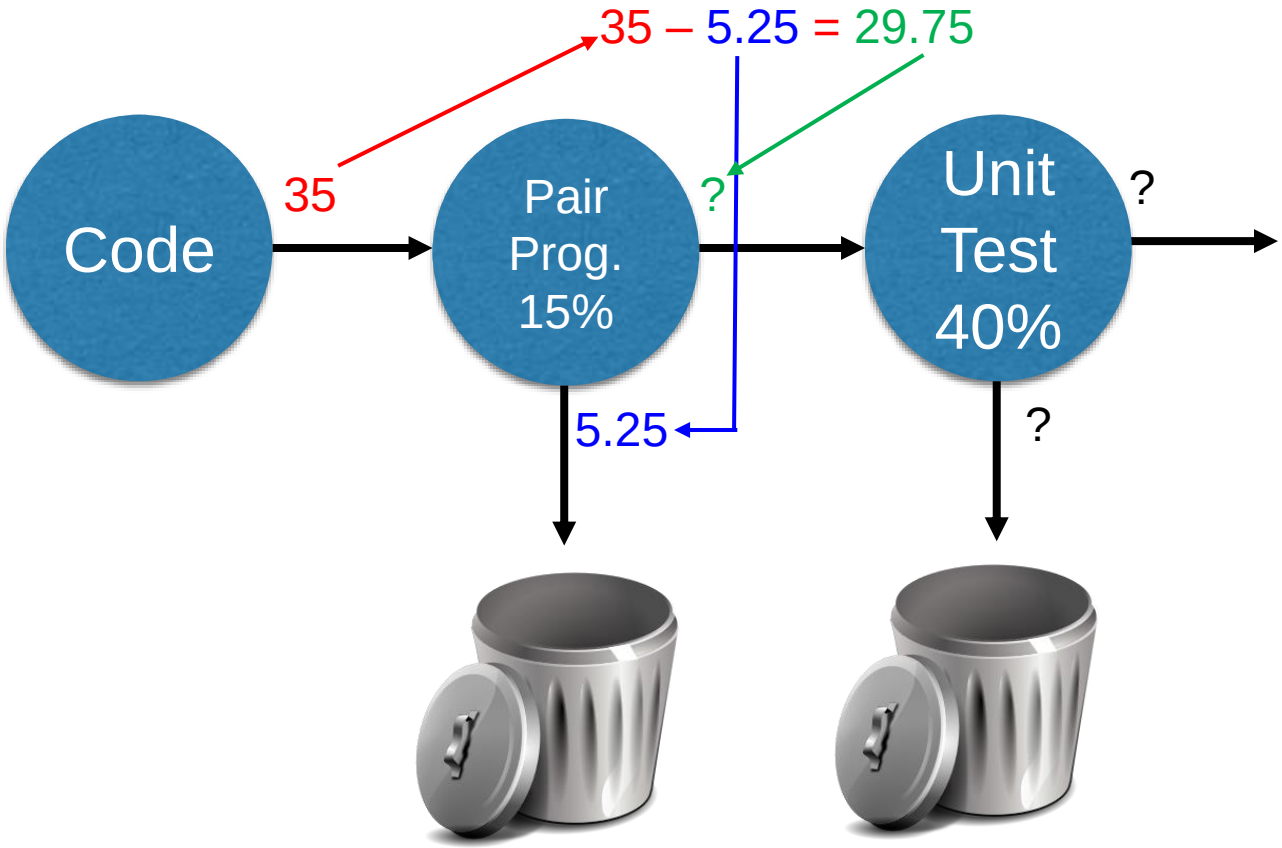
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

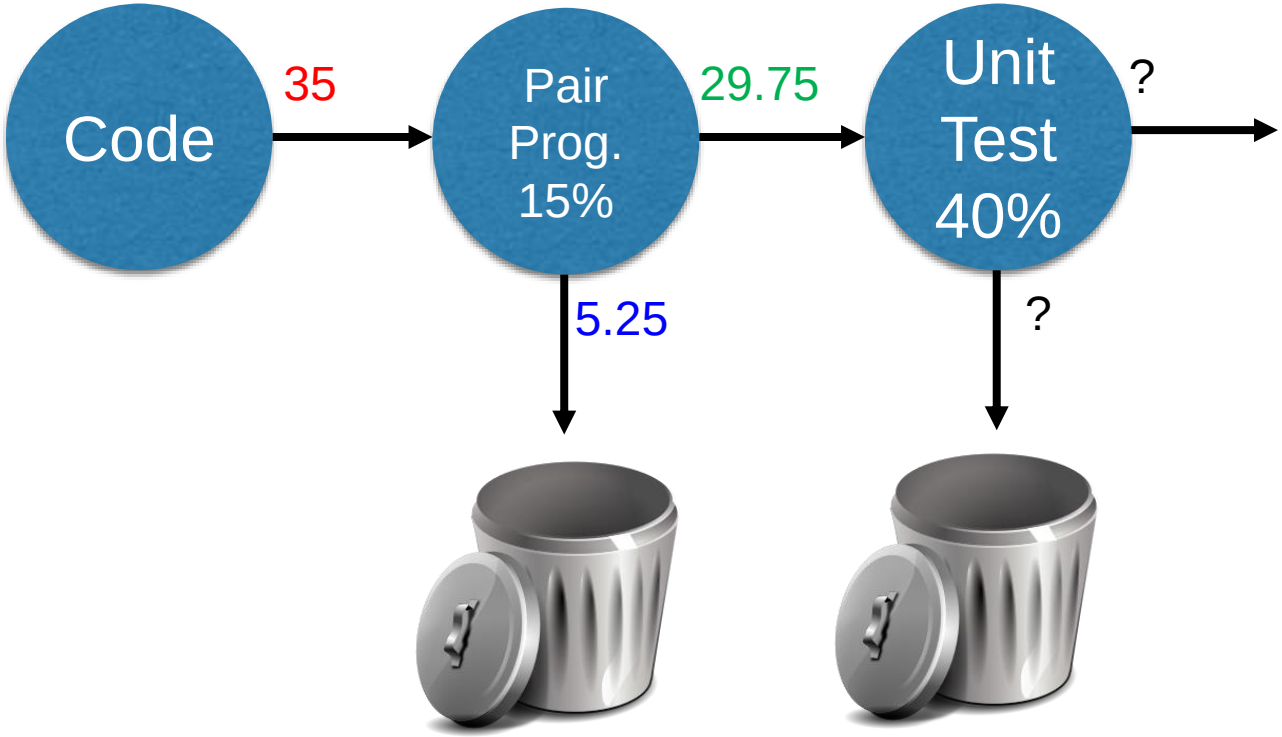
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

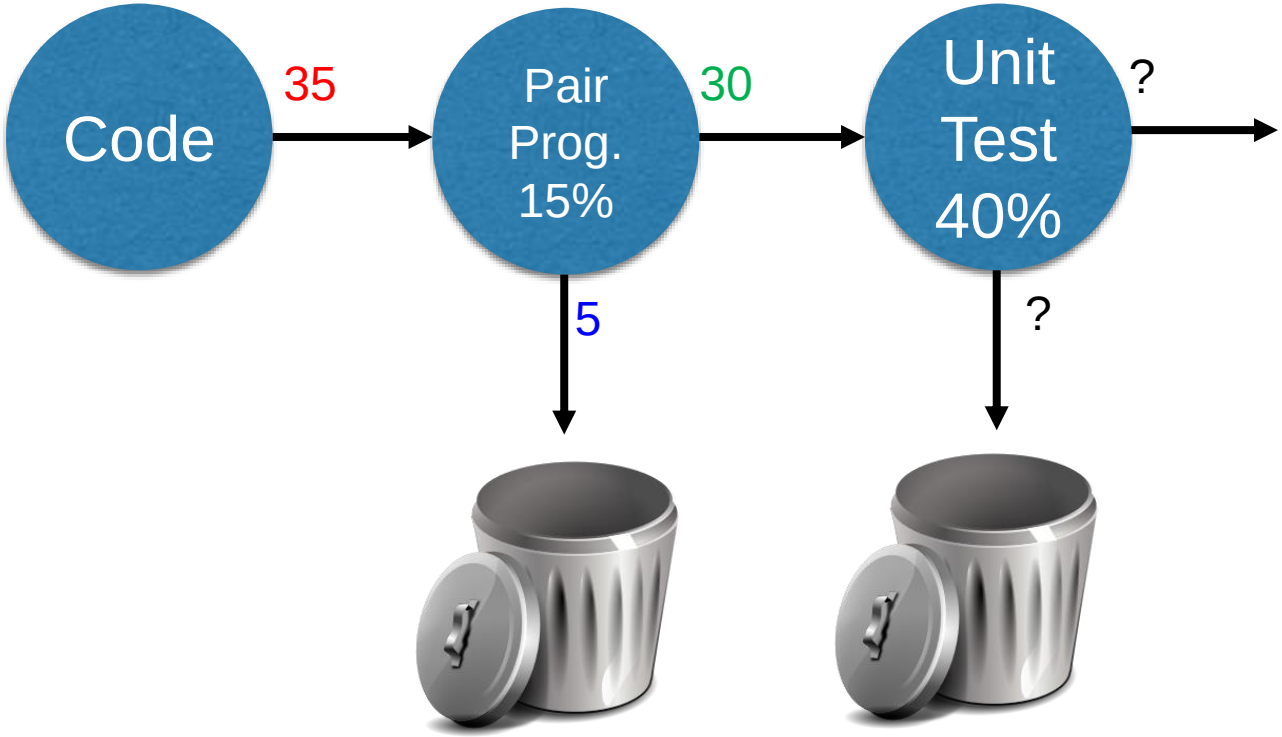
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

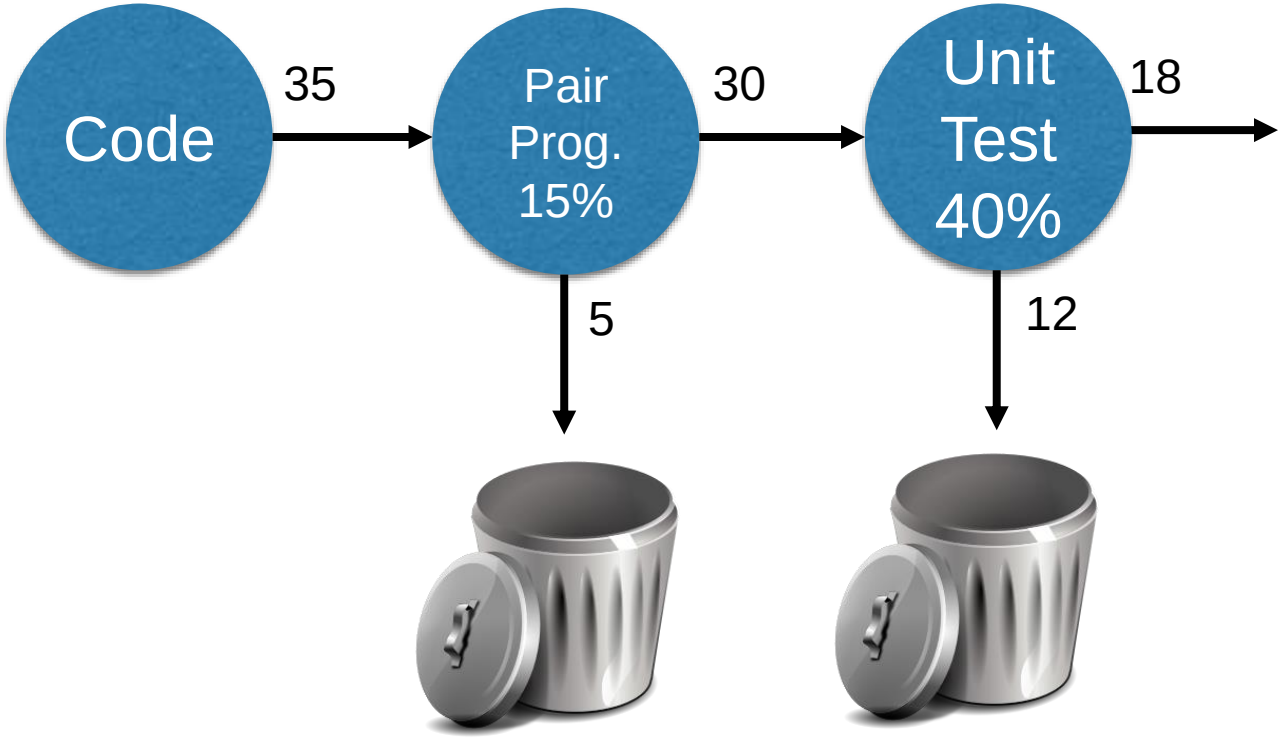
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



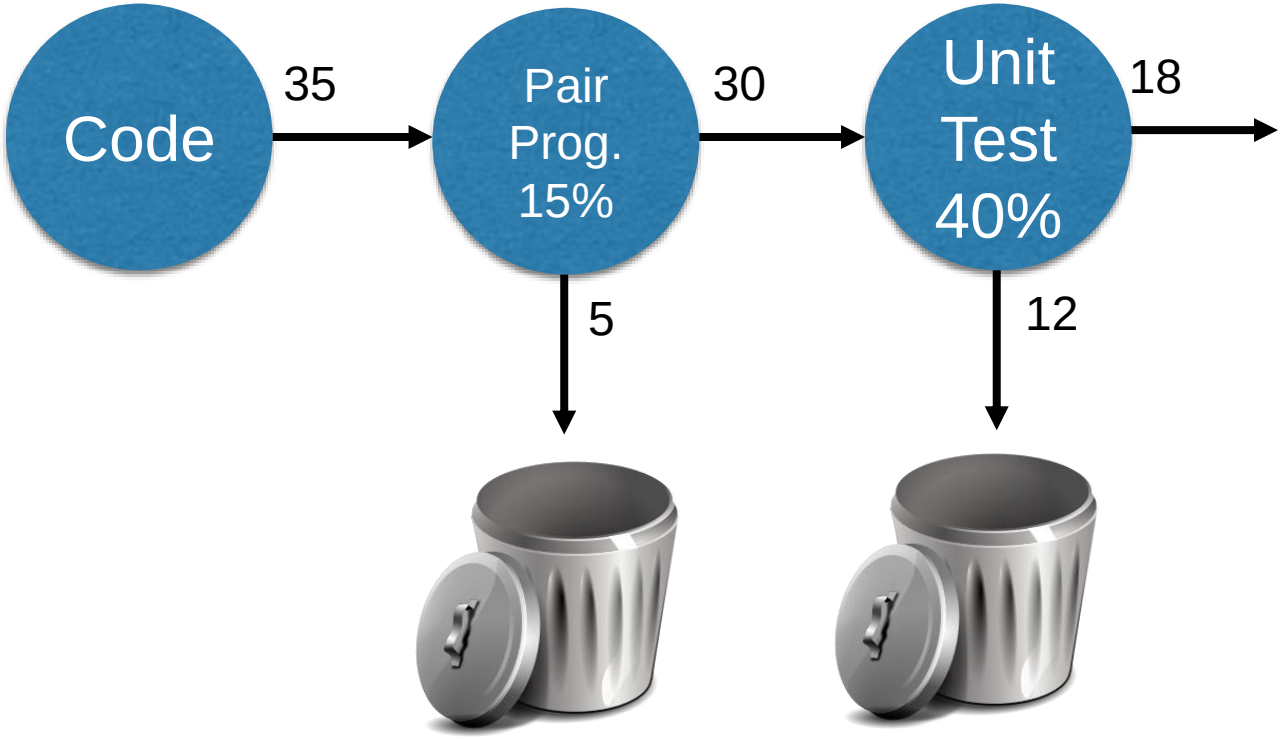
? = defect density removed by pair programming (defects/KLOC)

? = defect density remaining in the system after performing pair programming (defects/KLOC)

Modeling Pair Programming and Unit Testing as an Enhancement to Coding

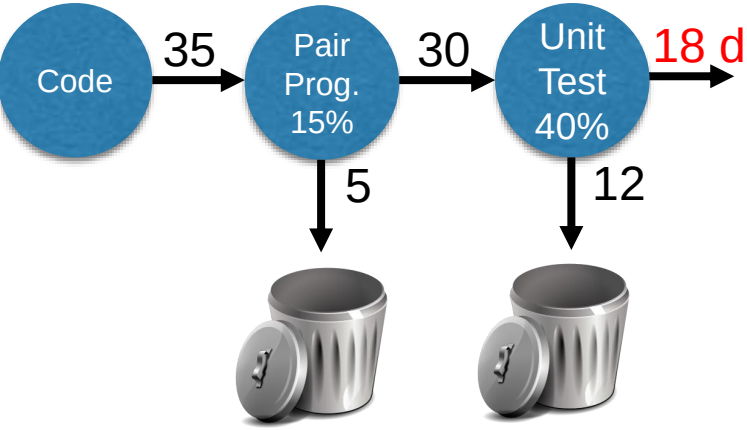


Modeling Pair Programming and Unit Testing as an Enhancement to Coding



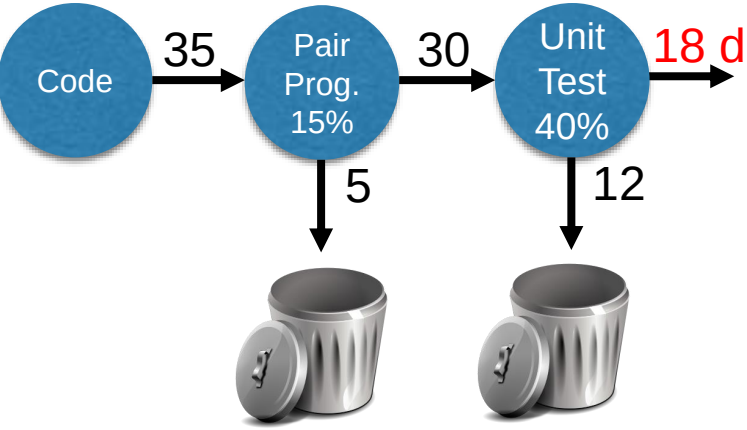
There is one important value in this Defect Removal Model…

Modeling Pair Programming and Unit Testing as an Enhancement to Coding



18 defects/KLOC represents:

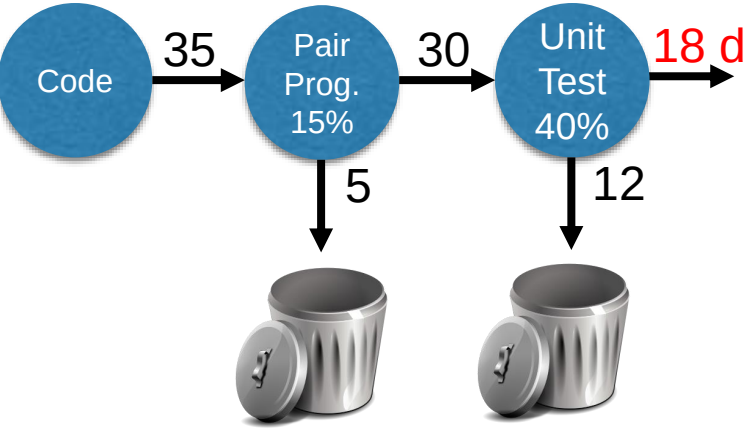
Modeling Pair Programming and Unit Testing as an Enhancement to Coding



represents:

- defect density remaining in the system after performing pair programming and unit tests
  - i.e., defect density delivered to the client

Modeling Pair Programming and Unit Testing as an Enhancement to Coding



represents:

- defect density remaining in the system after performing pair programming and unit tests
  - i.e., defect density delivered to the client

- Does this defect density satisfy my proposed Quality Goal for the project?
  - (see Sprint 2 assignment)