SWEN438 - DevOps

Lecture 9 - Lean Development

Dr David J. Pearce

School of Engineering and Computer Science
Victoria University of Wellington

“Think big, act small, fail fast; learn rapidly”

–Poppendieck & Poppendieck’03

Lean Manufacturing

Precisely specify **value** *by specific product, identify the* **value** *stream for each product, make* **value** *flow without interruptions, let customer pull* **value** *from the producer, and pursue perfection.*

–Womack & Jones, 2003.

Q) *How does this translate to Software Development?*

Lean Development

1. **Eliminate Waste.**
2. **Amplify Learning.**
3. **Decide late as possible.**
4. **Deliver as fast as possible.**
5. **Build integrity in.**
6. **Optimize the whole.**

–Wikipedia

- Adapted from lean manufacturing and the Toyota Production System
- Promoted initially through agile methods
- More recently adopted for DevOps

Eliminate Waste

Waste

Anything which does not **add value** *to the customer!*

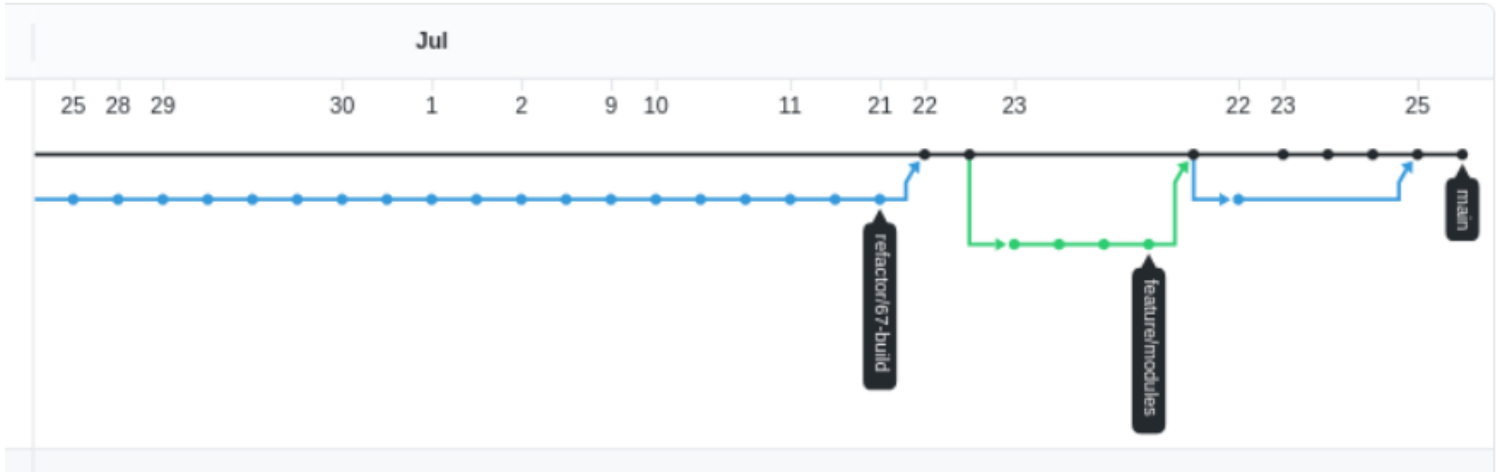
- **Unnecessary Features**. To do this effectively means deciding which features are useful to customers.
- **Partially Done Work**. Work which takes a long time to complete might be abandoned.
- **Task Switching**. Working on many features simultaneously is challenging!
- **Defects**. Testing can reduce feedback loop on finding problems without waiting for Q/A team.

Example Waste: Unnecessary Features

How do we know what is important?

- *More language features?*
- *Better verification capability?*
- *Better automated testing?*
- *Better compiler architecture?*
- *Support for incremental compilation?*
- *Better website?*
- *More libraries?*
- *Better IDE support?*
- *More backend compiler targets?*

Example Waste: Partially Done Work



- New features can take **months** to be implemented!
- Work taking a long time incurs overheads. *Why?*
  - » Costs associated with merge conflicts
  - » Work might be abandoned (e.g. if no-one can progress it)
  - » Task switching is a side-effect!

Example Waste: Defects

```
public export method test() :
    int x = 1
    assert x == 1
```

- **All Valid Tests** — *770 Whiley files which should compile and execute correctly*
- **All Valid Verification Tests** — *600 Whiley files which should compile and verify*
- **All Invalid Tests** — *436 Whiley files which should fail to compile or fail to verify*

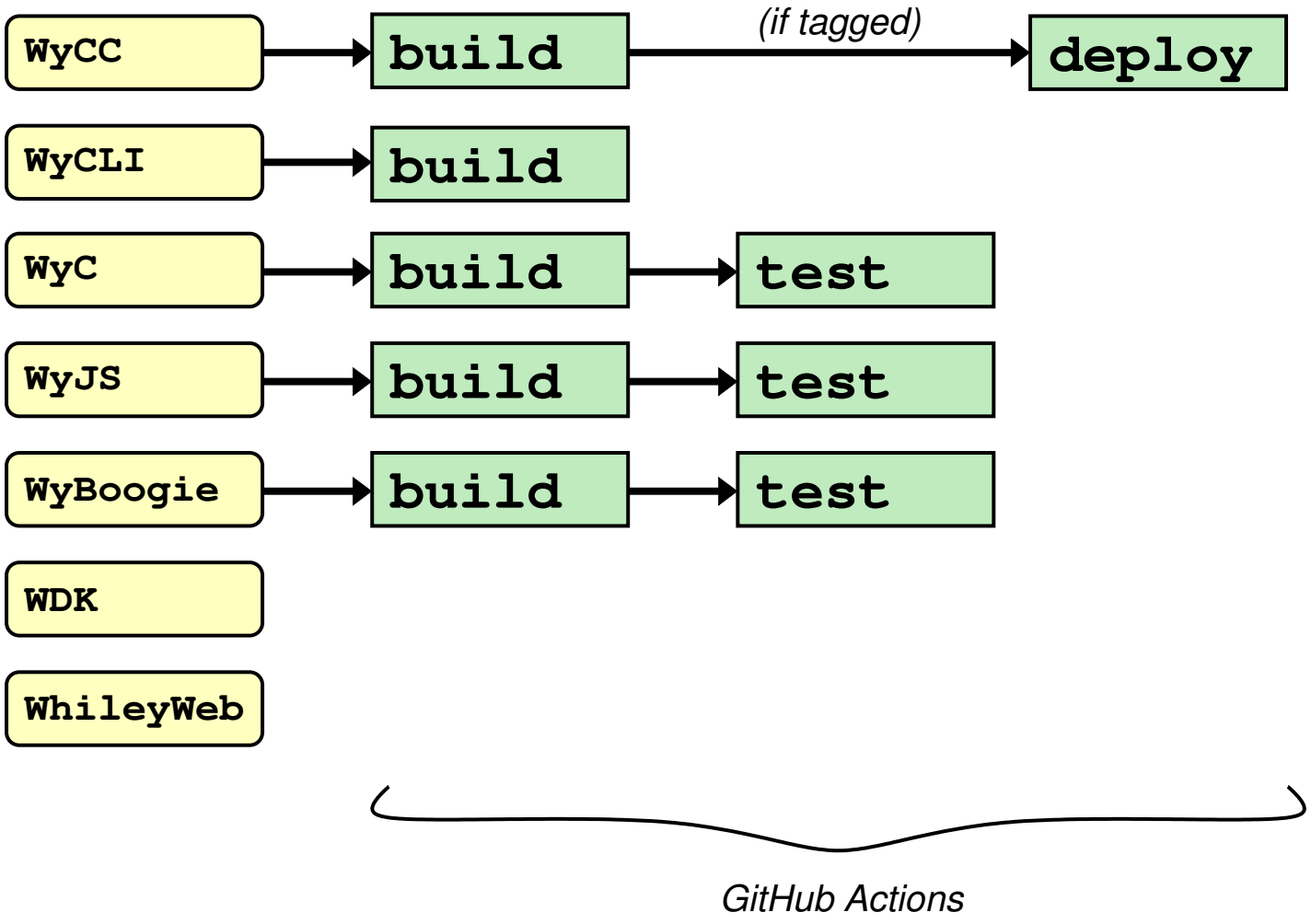
Over the years, have proved their weight in gold! **But what's missing?**

Example Waste: Defects (Continued)

> **May 19th, 2021.** *All tests passing, so pushed changes to GitHub. Moved straight into release and deploy, as no need to wait for CI/CD. But, after manually testing website,* something is wrong*.*

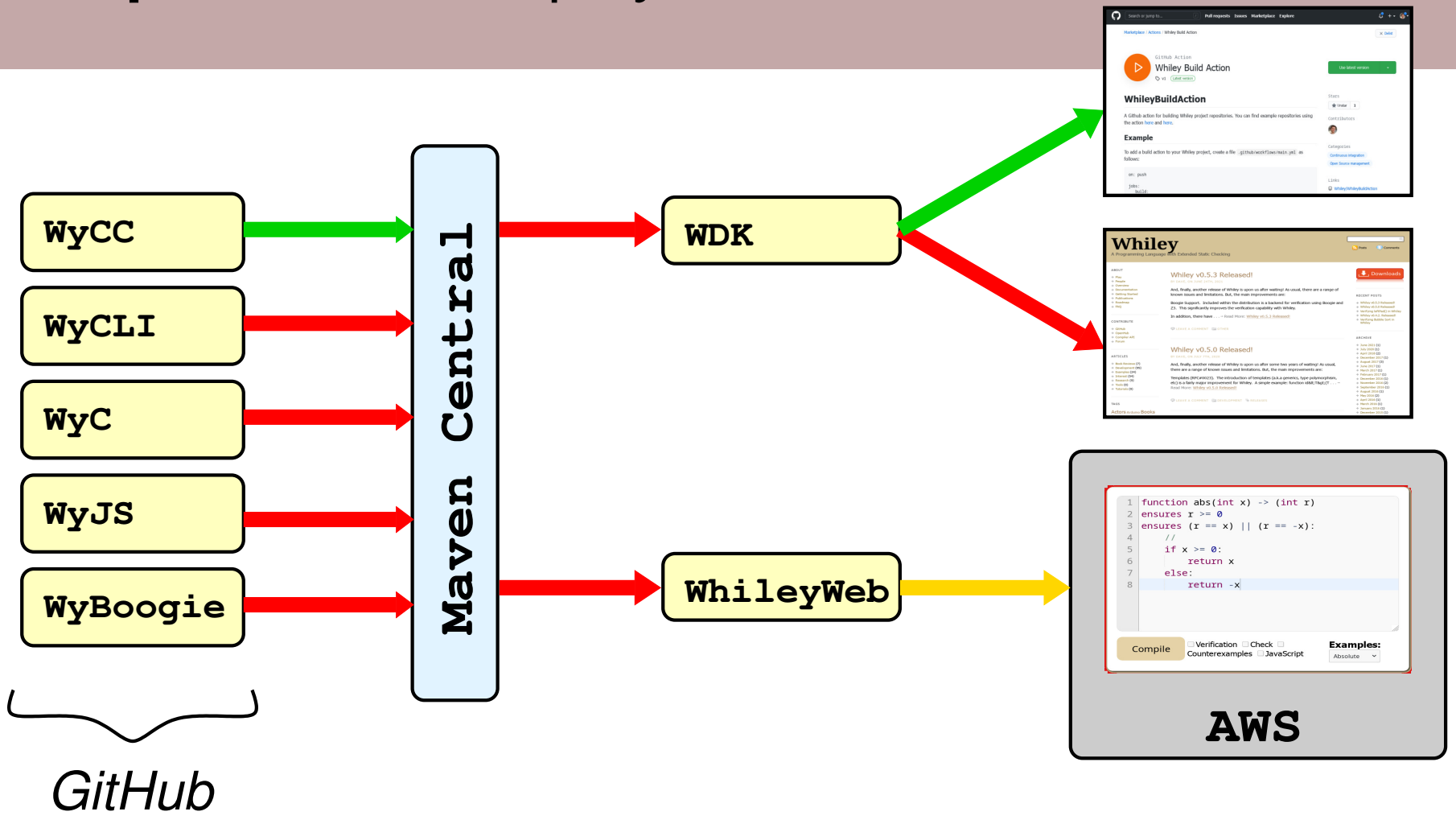
- Cannot test **separate files**
- Cannot test **runtime errors**
- Cannot test **QuickCheck**
- Don't forget to run all three!
- Doesn't fully test **generated IL**.
- Tests **duplicated** across repositories!

Example Waste: Build Automation



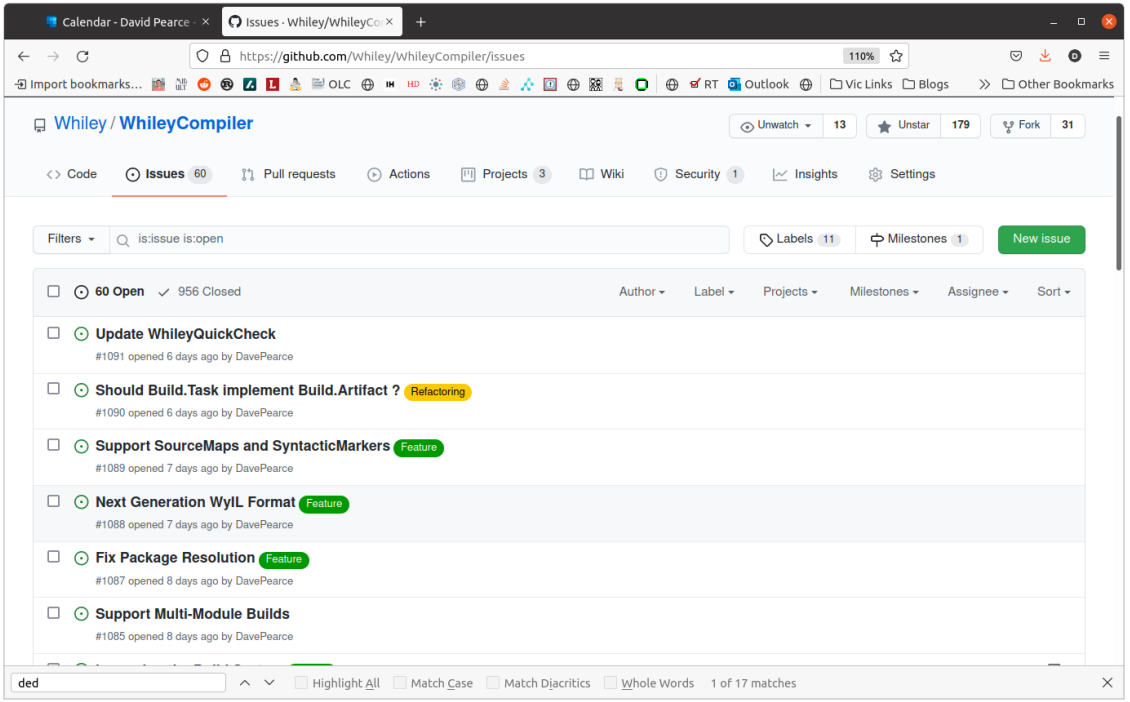
June 2nd, 2021. *Move from Travis CI to GitHub Actions.*

Example Waste: Deployment



June 11th, 2021. *Enable deployment to Maven Central. Pipeline failing for WyCLI ???*

Example Waste: Management Activities



- Issues for features **never** to be implemented?
- Lots of **duplicate issues**?
- Lots and lots of **Kanban boards**?

Amplify Learning

> *"**Development is quite different than production**. Think of development as creating a recipe, and production as following the recipe. These are very different activities, and they should be carried out with different approaches. Developing a recipe is a learning process involving trial and error ...*
> *... In order to solve problems that have not been solved before, it is necessary to* **generate information***"*
>
> *–Poppendieck & Poppendieck'03*

What should we be **learning** from?

- *Users of Whiley?*
- *Realistic Benchmarks for Whiley?*
- *Development Velocity of the Compiler?*
- *Metrics (e.g. performance, coverage)?*

Amplify Learning

"There are two schools of thought in developing software. One is to encourage developers to be sure that each design and each segment of code is **perfect the first time**. *The second school of thought holds that it is better to have small, rapid* **try-it test-it fix-it** *cycles than it is to make sure the design and code are perfect the first time."*

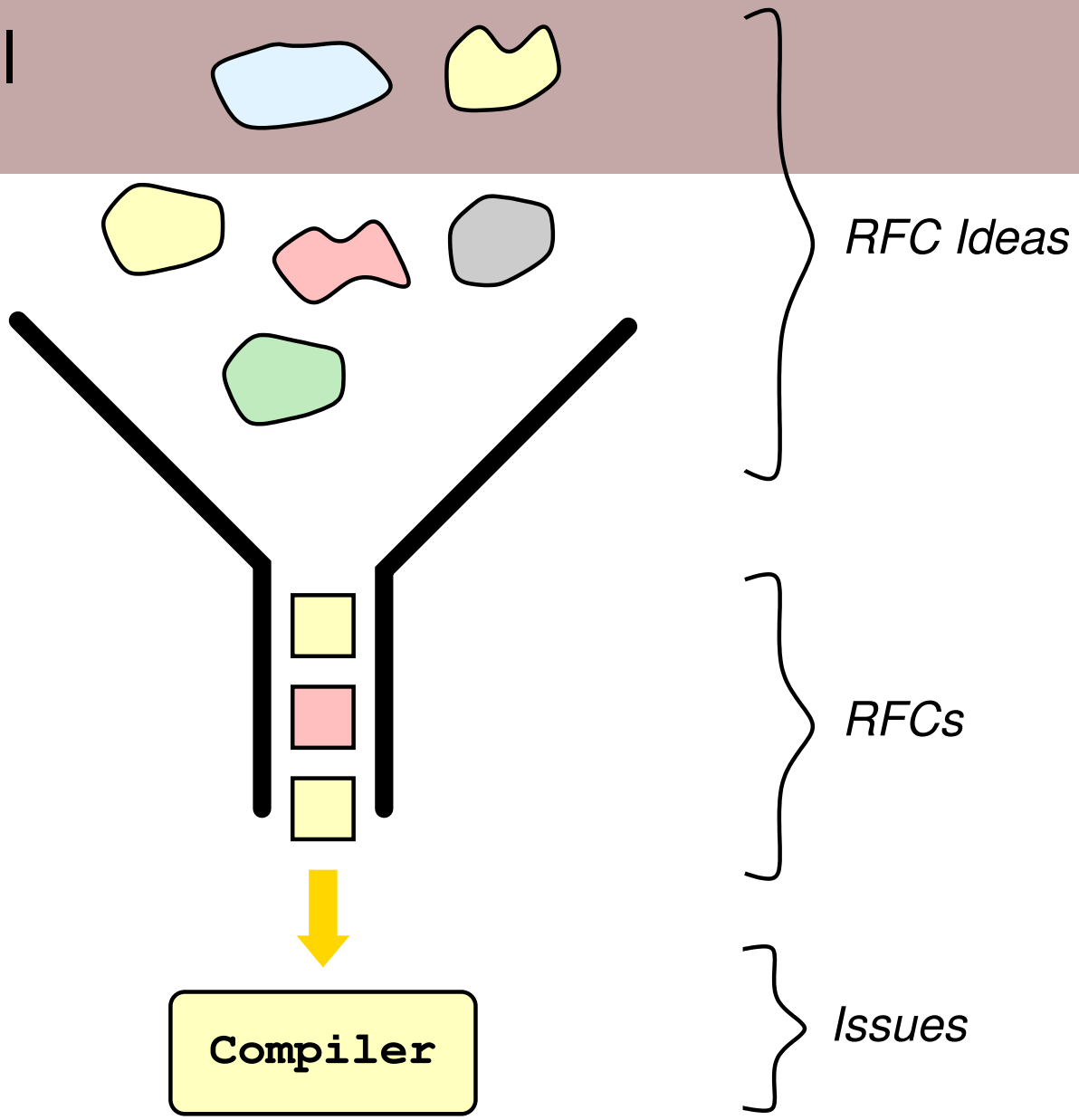
–Poppendieck & Poppendieck'03

Decide Late

> *"delaying decisions as much as possible until they can be made based on facts and not on uncertain assumptions and predictions. The more complex a system is, the more capacity for change should be built into it, thus enabling the delay of important and crucial commitments."*
>
> *–Wikipedia*

- *What approach to take for memory management?*
- *What should the concurrency model be?*
- *What is the Minimal Viable Product?*

Feature Funnel



June 2nd, 2021. *Unsigned bitwise shift would be handy*

Deliver Fast

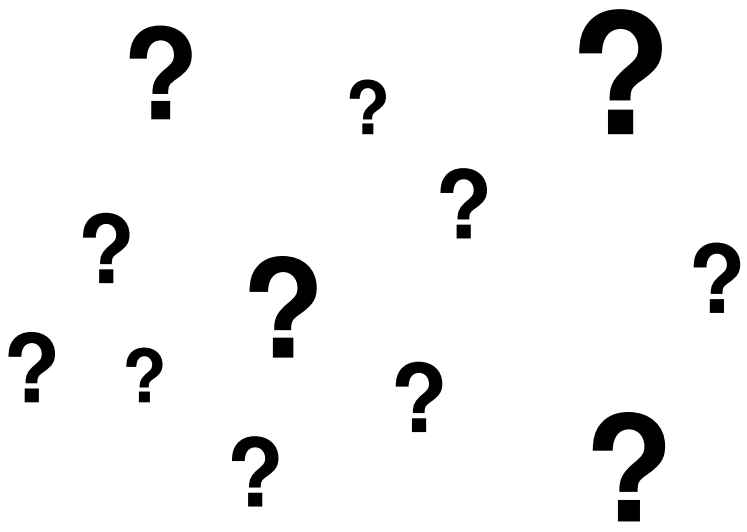
"In the era of rapid technology evolution, it is not the **biggest** *that survives, but the* **fastest***. The sooner the end product is delivered without major defects, the sooner feedback can be received, and incorporated into the next iteration."*

–Wikipedia

Reducing Batch Size

June 11th, 2021 *New versions of WyCC, WyCLI and WyC released. These included partially completed feature. The old design and the new design were released together in the same packages as part of a transitional period.*

Examples



- Want to add memory management feature (e.g. `&`**`int`**, **`new`** `e`, **`delete`** `p`, etc).
- Want to redesign **build system**. *This provides core functionality across all parts of the compiler!*
- Want to **rewrite** Whiley compiler in Whiley! *How to even begin?*

References

- Mary Poppendieck; Tom Poppendieck (2003). *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional