Principles of Software Construction: Objects, Design, and Concurrency

Part 4: Et cetera

Toward SE in practice: People and process

Josh Bloch **Charlie Garrod**

Carnegie Mellon University
School of Computer Science

institute for SOFTWARE RESEARCH

Administrivia

- Homework 6 available
  - Checkpoint deadline Thursday, December 3rd
  - Due Wednesday, December 9th

Key concepts from last Tuesday

- Java lambdas and streams

Use caution making streams parallel

Remember our Mersenne primes program?

```
static Stream<BigInteger> primes() {
    return Stream.iterate(TWO, BigInteger::nextProbablePrime);
}

public static void main(String[] args) {
    primes().map(p -> TWO.pow(p.intValueExact()).subtract(ONE))
        .filter(mersenne -> mersenne.isProbablePrime(50))
        .limit(20)
        .forEach(System.out::println);
}
```

Runs in 10.1s on my 12-core, 24-thread Ryzen 9 3900X

Does not reasonably terminate if the stream is .parallel()

Lambdas and streams summary

- When to use a lambda
  - Always, in preference to CICE
- When to use a method reference
  - Almost always, in preference to a lambda
- When to use a stream
  - When it feels and looks right
- When to use a parallel stream
  - When you've convinced yourself it has equivalent semantics and demonstrated that it's a performance win

What Josh didn't show you…

Stream interface is a monster (1/3)

```
public interface Stream<T> extends BaseStream<T, Stream<T>> {
 // Intermediate Operations
 Stream<T> filter(Predicate<T>);
 <R> Stream<R> map(Function<T, R>);
 IntStream mapToInt(ToIntFunction<T>);
 LongStream mapToLong(ToLongFunction<T>);
 DoubleStream mapToDouble(ToDoubleFunction<T>);
 <R> Stream<R> flatMap(Function<T, Stream<R>>);
 IntStream flatMapToInt(Function<T, IntStream>);
 LongStream flatMapToLong(Function<T, LongStream>);
 DoubleStream flatMapToDouble(Function<T, DoubleStream>);
 Stream<T> distinct();
 Stream<T> sorted();
 Stream<T> sorted(Comparator<T>);
 Stream<T> peek(Consumer<T>);
 Stream<T> limit(long);
 Stream<T> skip(long);
```

Stream interface is a monster (2/3)

```
// Terminal Operations
void forEach(Consumer<T>);          // Ordered only for sequential streams
void forEachOrdered(Consumer<T>);   // Ordered if encounter order exists
Object[] toArray();
<A> A[] toArray(IntFunction<A[]> arrayAllocator);
T reduce(T, BinaryOperator<T>);
Optional<T> reduce(BinaryOperator<T>);
<U> U reduce(U, BiFunction<U, T, U>, BinaryOperator<U>);
<R, A> R collect(Collector<T, A, R>); // Mutable Reduction Operation
<R> R collect(Supplier<R>, BiConsumer<R, T>, BiConsumer<R, R>);
Optional<T> min(Comparator<T>);
Optional<T> max(Comparator<T>);
long count();
boolean anyMatch(Predicate<T>);
boolean allMatch(Predicate<T>);
boolean noneMatch(Predicate<T>);
Optional<T> findFirst();
Optional<T> findAny();
```

Stream interface is a monster (3/3)

```
  // Static methods: stream sources
  public static <T> Stream.Builder<T> builder();
  public static <T> Stream<T> empty();
  public static <T> Stream<T> of(T);
  public static <T> Stream<T> of(T...);
  public static <T> Stream<T> iterate(T, UnaryOperator<T>);
  public static <T> Stream<T> generate(Supplier<T>);
  public static <T> Stream<T> concat(Stream<T>, Stream<T>);
}
```

In case your eyes aren't glazed yet

```
public interface BaseStream<T, S extends BaseStream<T, S>>
   extends AutoCloseable {
 Iterator<T> iterator();
 Spliterator<T> spliterator();
 boolean isParallel();
 S sequential();  // May have little or no effect
 S parallel();    // May have little or no effect
 S unordered();   // Note asymmetry wrt sequential/parallel
 S onClose(Runnable);
 void close();
}
```

It keeps going: `java.util.stream.Collectors`

```
... toList()
... toMap(...)
... toSet(...)
... reducingBy(...)
... groupingBy(...)
... partitioningBy(...)

        .
        .
        .
```

It keeps going: `java.util.stream.Collectors`

```
... toList()
... toMap(...)
... toSet(...)
... reducingBy(...)
... groupingBy(...)
... partitioningBy(...)

        .
        .
        .
static <T,K,D,A,M extends Map<K,D>> Collector<T,?,M> groupingBy(
    Function<? super T,? extends K> classifier,
    Supplier<M> mapFactory,
    Collector<? super T,A,D> downstream)
```

Optional<T> – a third way to indicate the absence of a result

```
public final class Optional<T> {
    boolean isPresent();
    T get();

    void ifPresent(Consumer<T>);
    Optional<T> filter(Predicate<T>);
    <U> Optional<U> map(Function<T, U>);
    <U> Optional<U> flatMap(Function<T, Optional<U>>);
    T orElse(T);
    T orElseGet(Supplier<T>);
    <X extends Throwable> T orElseThrow(Supplier<X>) throws X;
}
```

Changes to existing libraries… e.g.,

```
public interface Collection<E> {
  …
  default Stream<E> stream();
  default Stream<E> parallelStream();
  default Spliterator<E> spliterator();
}
```

Overall: Streams design discussion

- Recall the fundamental API design principles…

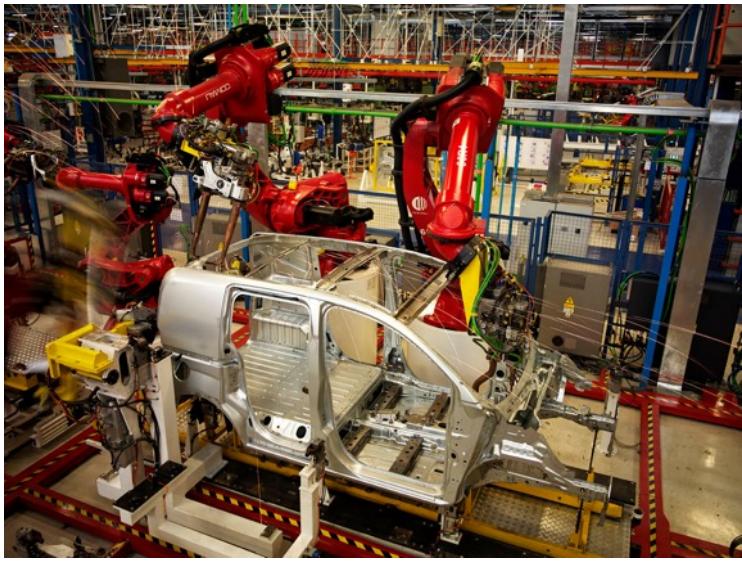
Today: Software engineering in practice

- An introduction to software engineering
- Methodologies discussion: Test-driven development

What is software engineering?

Compare to other forms of engineering

- e.g., Producing a car or bridge
  - Estimable costs and risks
  - Well-defined expected results
  - High quality
- Separation between plan and production
- Simulation before construction
- Quality assurance through measurement
- Potential for automation



Software engineering in the real world

- e.g., HealthCare.gov
  - Estimable costs and risks
  - Well-defined expected results
  - High quality
- Separation between plan and production
- Simulation before construction
- Quality assurance through measurement
- Potential for automation

1968 NATO Conference on Software Engineering

Sociotechnical systems

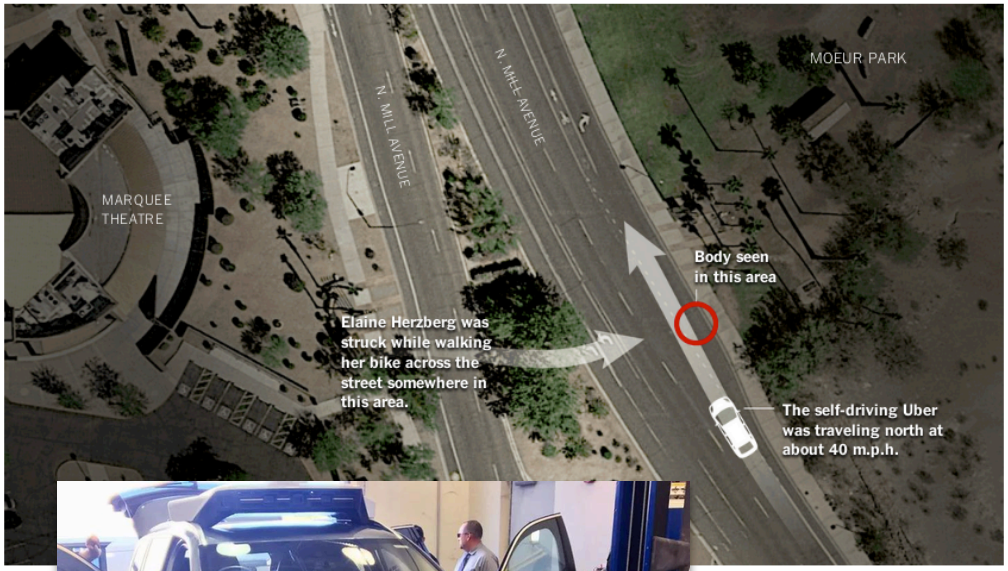
- A *sociotechnical system* is, roughly, any interlinked system of people, technology, and their environment

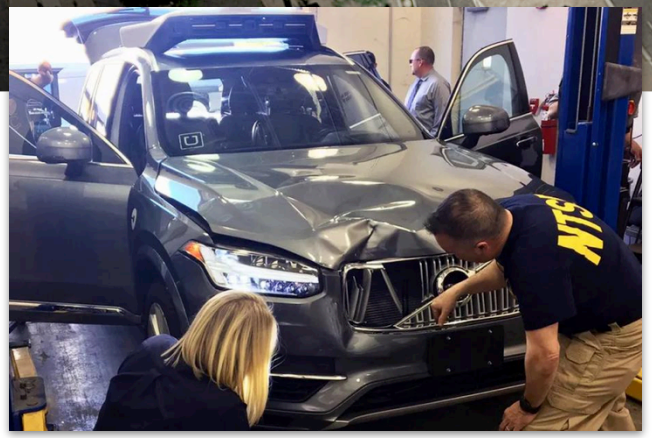
How a Self-Driving Uber Killed a Pedestrian in Arizona

By TROY GRIGGS and DAISUKE WAKABAYASHI UPDATED MARCH 21, 2018

A woman was struck and killed on Sunday night by an autonomous car operated by Uber in Tempe, Ariz. It was believed to be the first pedestrian death associated with self-driving technology.

What We Know About the Accident





BBC Sign in | News | Sport | Reel | Worklife | Travel | Future | Mo

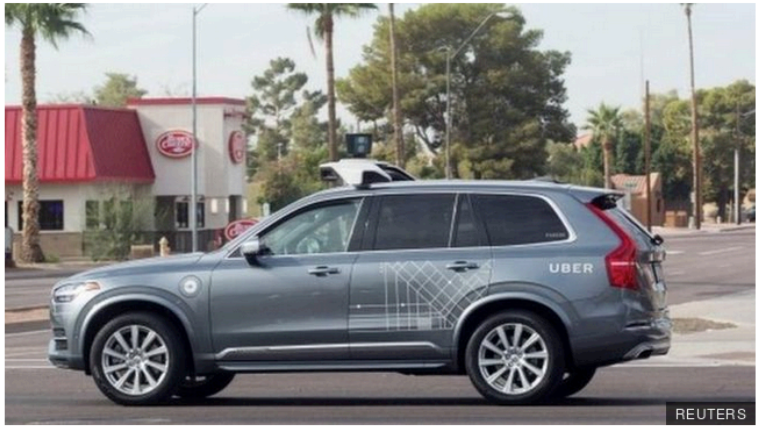
NEWS

Home | Video | World | US & Canada | UK | Business | Tech | Science | Stories | Entertai

Business | Market Data | Global Trade | Companies | Entrepreneurship | Technology of Busir

Uber in fatal crash had safety flaws say US investigators

6 November 2019 Share



An Uber self-driving test vehicle that hit and killed a woman in 2018 had software problems, according to US safety investigators.

Elaine Herzberg, 49, was hit by the car as she was crossing a road in Tempe, Arizona.

The US National Transportation Safety Board (NTSB) found the car failed to identify her properly as a pedestrian.

The detailed findings raised a series of safety issues but did not determine the probable cause of the accident.

https://www.nytimes.com/interactive/2018/03/20/us/self-driving-uber-pedestrian-killed.html?mtrref=www.google.com&assetType=REGIWALL
https://www.bbc.com/news/business-50312340
https://www.bbc.com/news/technology-44243118

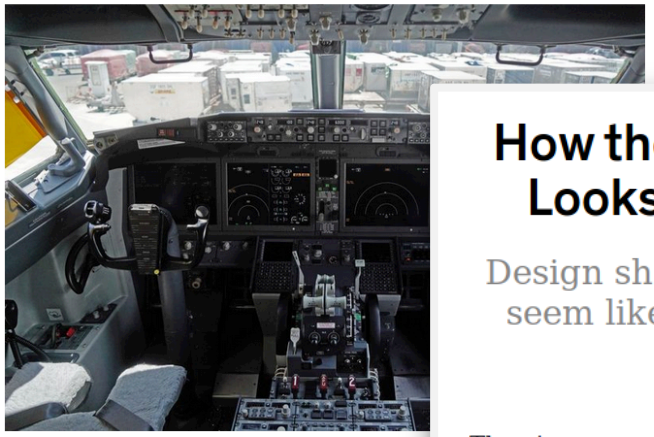
Search Bloomberg Sig

Technology

Boeing's 737 Max Software Outsourced to $9-an-Hour Engineers

By Peter Robison
June 28, 2019, 4:46 PM EDT

- Planemaker and suppliers used lower-paid temporary workers
- Engineers feared the practice meant code wasn't done right



The cockpit of a grounded 737 Max 8 aircraft. *Photographer: Dimas*

It remains the mystery at the hea
crisis: how a company renowned
made seemingly basic software n
deadly crashes. Longtime Boeing
was complicated by a push to ou
contractors.

The Max software -- plagued by is
planes grounded months longer
week revealed a new flaw -- was
was laying off experienced engin
suppliers to cut costs.

https://spectrum.ieee.org/aerospace/aviation/h
developer

How the Boeing 737 Max Disaster Looks to a Software Developer

Design shortcuts meant to make a new plane seem like an old, familiar one are to blame

By **Gregory Travis**

The views expressed here are solely those of the author and do not represent positions of IEEE Spectrum *or the IEEE.*

Photo: Jemal Countess/Getty Images

This is part of the wreckage of Ethiopian Airlines Flight ET302, a Boeing 737 Max

A year after the first 737 Max crash, it's unclear when the plane will fly again

Two crashes of Boeing's 737 Max 8 killed 346 people, and authorities are blaming Boeing's design, a faulty sensor and airline staff. Plus: Everything you need to know about the plane.

Kent German November 1, 2019 9:01 AM PDT

ed killing 346 people.

ts 737 Max 8 that killed 346 people, Boeing is facing
its newest and most critical aircraft models. The
und the world, and the Federal Aviation

institute for SOFTWARE RESEARCH

Major topics in 17-313 (Foundations of SE)

- Process considerations for software development
- Requirements elicitation, documentation, and evaluation
- Design for quality attributes
- Strategies for quality assurance
- Empirical methods in software engineering
- Time and team management
- Economics of software development

The foundations of our Software Engineering program

- Core computer science fundamentals
- Building good software, organizing software projects
  - Development teams, customers, and users
  - Process, requirements, estimation, management, and methods
- The larger context of software
  - Business, society, policy
- Engineering experience
- Communication skills
  - Written and oral

Summary

- Software engineering requires consideration of many issues, social and technical, above code-level considerations
- Interested? Take 17-313
- Shameless plug: Take API Design, 17-480