A Farewell to Soul-Crushing Code

Nicole Rauch

@nicolerauch

Michael Sperber

@sperbsen

Soul-Crushing Code

```
package innocent.plugin.adapter;

import org.eclipse.core.runtime.IAdapterFactory;
import org.eclipse.ui.views.properties.IPropertySource;
import innocent.plugin.adapter.model.Todo;

public class TodoAdapterFactory implements IAdapterFactory {
  @Override
  public Object getAdapter(Object adaptableObject, Class adapterType) {
    if (adapterType== IPropertySource.class && adaptableObject instanceof Todo){
      return new TodoPropertySource((Todo) adaptableObject);
    }
    return null;
  }
  @Override
  public Class[] getAdapterList() {
    return new Class[] { IPropertySource.class };
  }
}
```

Soul-Crushing Code

```
public class NativeQueryInterpreterInitiator implements SessionFactoryServiceInitiator<NativeQueryInterpreter> {
  public static final NativeQueryInterpreterInitiator INSTANCE = new NativeQueryInterpreterInitiator();

  @Override
  public NativeQueryInterpreter initiateService(
    SessionFactoryImplementor sessionFactory,
    SessionFactoryOptions sessionFactoryOptions,
    ServiceRegistryImplementor registry) {
      return new NativeQueryInterpreterStandardImpl( sessionFactory );
  }
  @Override
  public NativeQueryInterpreter initiateService(SessionFactoryServiceInitiatorContext context) {
    return new NativeQueryInterpreterStandardImpl( context.getSessionFactory() );
  }

  @Override
  public Class<NativeQueryInterpreter> getServiceInitiated() {
    return NativeQueryInterpreter.class;
  }
}
```

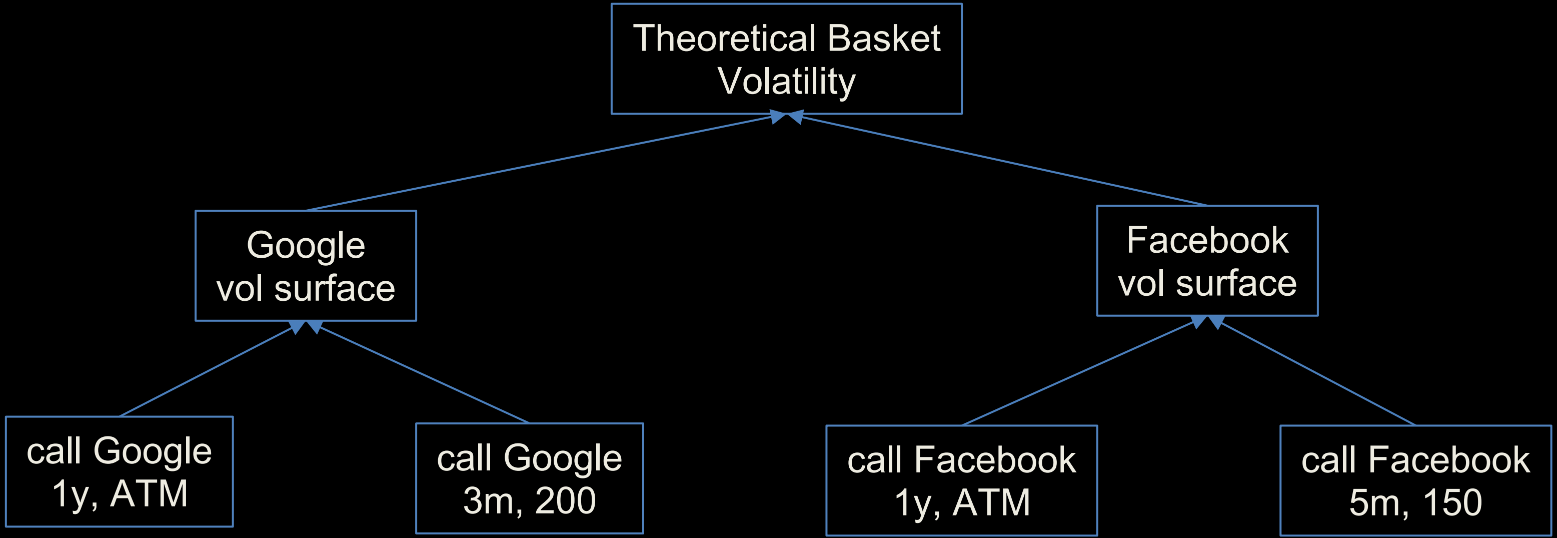
Soul-Crushing Code

```
DRecherche::DRecherche()
{
  if(DIMAServerModel::iFulltext)
    m_pcoImaObjectBuilder =
      (DImaObjectBuilder*) new DFTRechercheBuilder;
  else
    m_pcoImaObjectBuilder =
      (DImaObjectBuilder*) new DRechercheBuilder;
}
```

Soul-Crushing Code

```
void DRecherche::Expand(DData* pcoDestData, DBool bRecursive, DBool bCrossOver)
{
  if(!bRecursive) {
    //m_pcoData->CleanDataContainer();
    if(DIMAServerModel::iFulltext)
      ((DFTRechercheBuilder*)m_pcoImaObjectBuilder)
        ->VolltextRecherche(m_pcoData, pcoDestData);
    else
      ((DRechercheBuilder*)m_pcoImaObjectBuilder)
        ->Recherche(m_pcoData,pcoDestData);
    // Flag setzen, damit asynchroner Job nach dem Zuruecksenden
    // der Objekte aktiviert wird
    m_iAsyncFlag = 1;
  } // if bRecursive
}
```

Market Data



Theoretical Basket Volatility

Google vol surface

Facebook vol surface

call Google 1y, ATM

call Google 3m, 200

call Facebook 1y, ATM

call Facebook 5m, 150

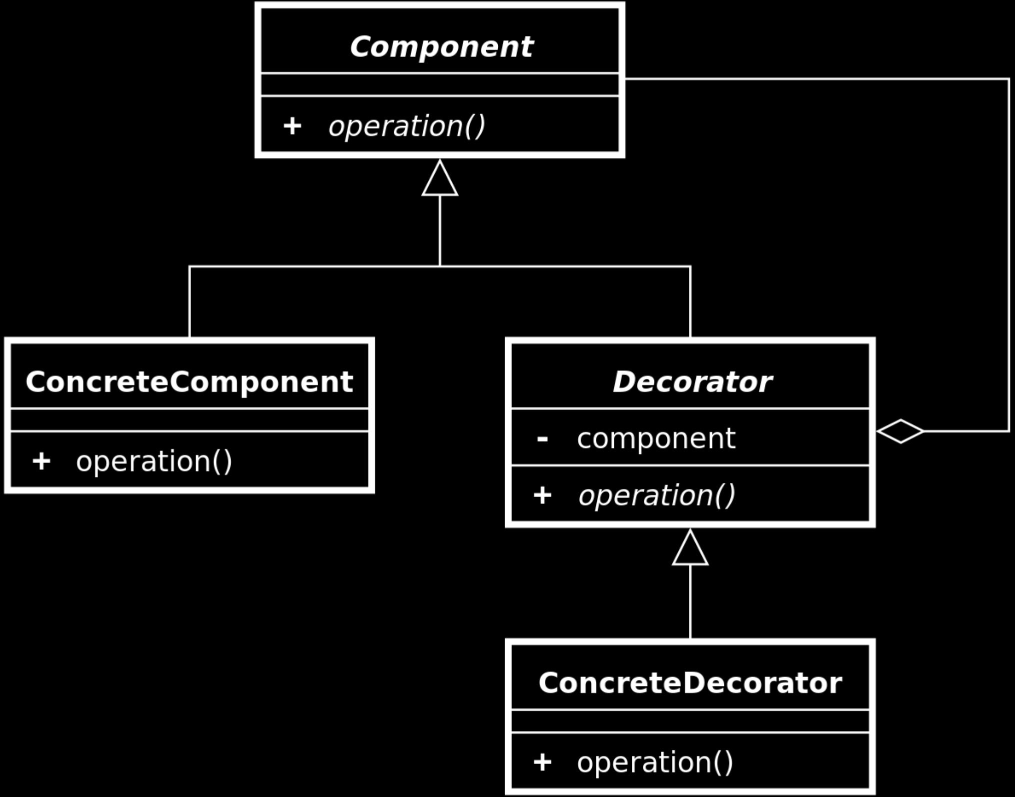
Sources of Market Data

```
class MarketData {
  double GetSpot(long sicovam);
  double GetVolat(long sicovam,
                  double maturity,
                  double strike);
}
```

Sources of Market Data

```
class SpotShiftedMarketData
  : public MarketData {
double GetSpot(long sicovam) {
  return MarketData::GetSpot(sicovam)
          * factor;
  }
}
```

Decorator Pattern



Source: https://en.wikipedia.org/wiki/Decorator_pattern

Decorated Sources of Market Data

```
class MarketDataDecorator : public MarketData {
  MarketData* fMarketData;
  MarketDataDecorator(MarketData* marketData)
    : fMarketData(marketData) {}

  double GetSpot(long sicovam) {
    return fMarketData->GetSpot(sicovam);
  }
}
```

Baskets

```
double SpotShiftedMarketData::GetSpot(long sicovam)
{
  return fMarketData->GetSpot(sicovam) * factor;
}

double MarketData::GetSpot(long sicovam) {
  return f(GetSpot(basketComponent), ...)
}
```

Soul-Crushing Code

```
double CURiskMatrixMarketData::GetVolat( long code,
                                         double startDate,
                                         double endDate,
                                         double strike,
                                         NSREnums::eVolatilityType volat,
                                         Boolean put,
                                         const CSRMarketData *context) const
{
  if (CalledFromAPricingFunctionFromLV) {
    double init_vol = 0.0;
    const CSRInstrument *instrument = GetCSRInstrument(code);
    if(instrument
        && instrument->HasAVolatilityFormula()
        && !DoesTheFirstMarketDataDeriveFromTheSecondOne(this,context))
        init_vol = fMarketData->GetVolat(code,startDate,endDate,strike,volat,put,fMarketData);
    else
      init_vol = fMarketData->GetVolat
                               (code,startDate,endDate,strike,volat,put,(context)?context:this);
    return fabsolute_volat_shift_factor + init_vol;
  }
  else {
    const CSRInstrument *instrument = GetCSRInstrument(code);
    double vol = HVBMarketDataDelegator::GetVolat(code,startDate,endDate,strike,volat,put,context);

    if(instrument && instrument->HasAVolatilityFormula())
      return vol;
    return vol + fabsolute_volat_shift_factor;
  }
}
```

Functional Programming!

1. immutable data
2. less coupling
3. verification

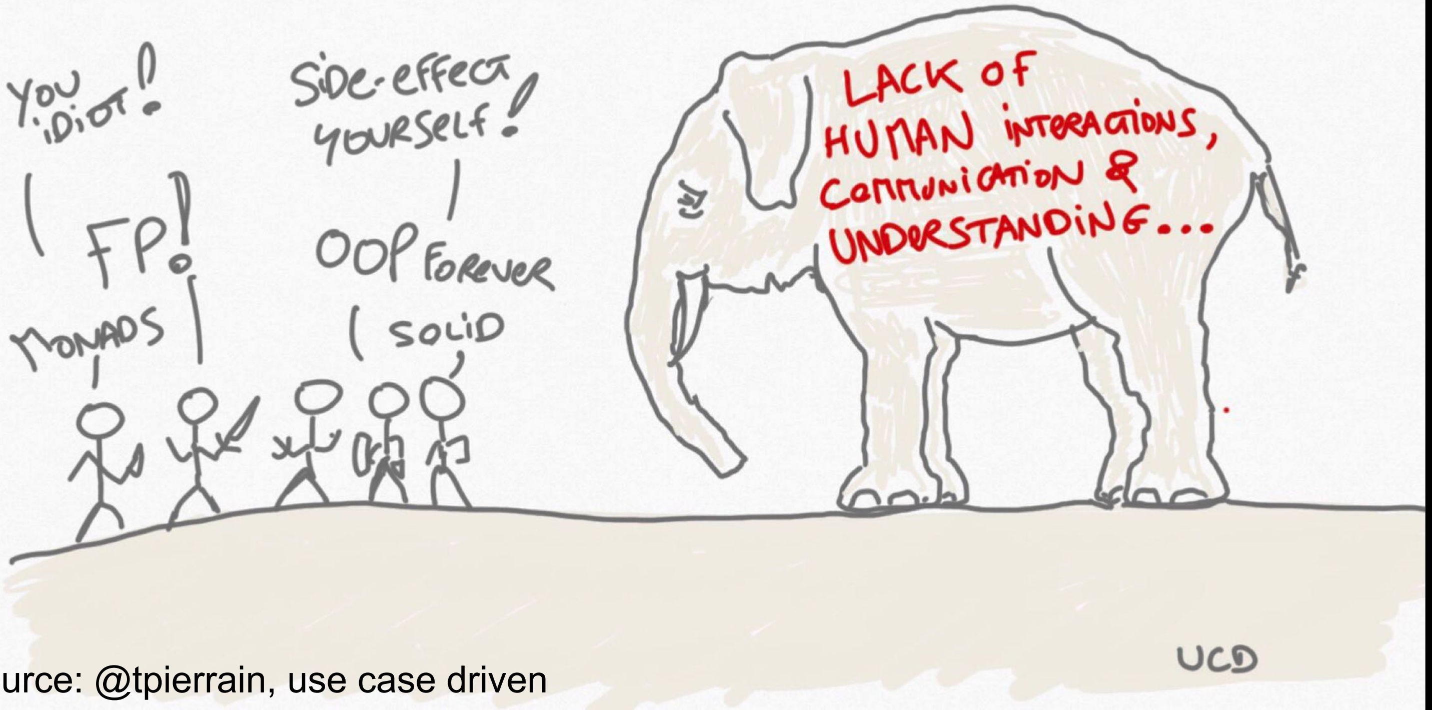
Functional Programming!

1. immutable data
2. less coupling
3. verification
4. catamorphisms
5. bifunctors

Functional Programming!

1. immutable data
2. less coupling
3. verification
4. catamorphisms
5. bifunctors
6. monads
7. monadic profunctors
8. Kleisli arrows

ELEPHANT IN THE I.T. ROOM?


YOU IDIOT!
SIDE-EFFECT YOURSELF!
FP!
OOP FOREVER
MONADS
SOLID
LACK OF HUMAN INTERACTIONS, COMMUNICATION & UNDERSTANDING...
UCD

Source: @tpierrain, use case driven

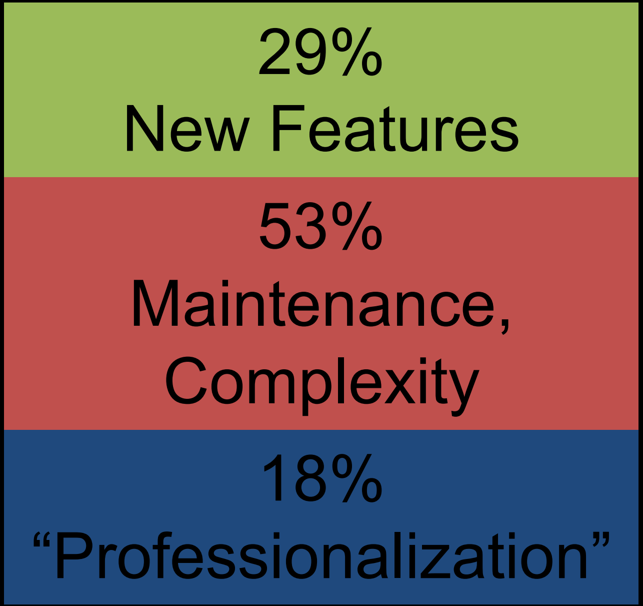
Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

After the Agile Revoluion



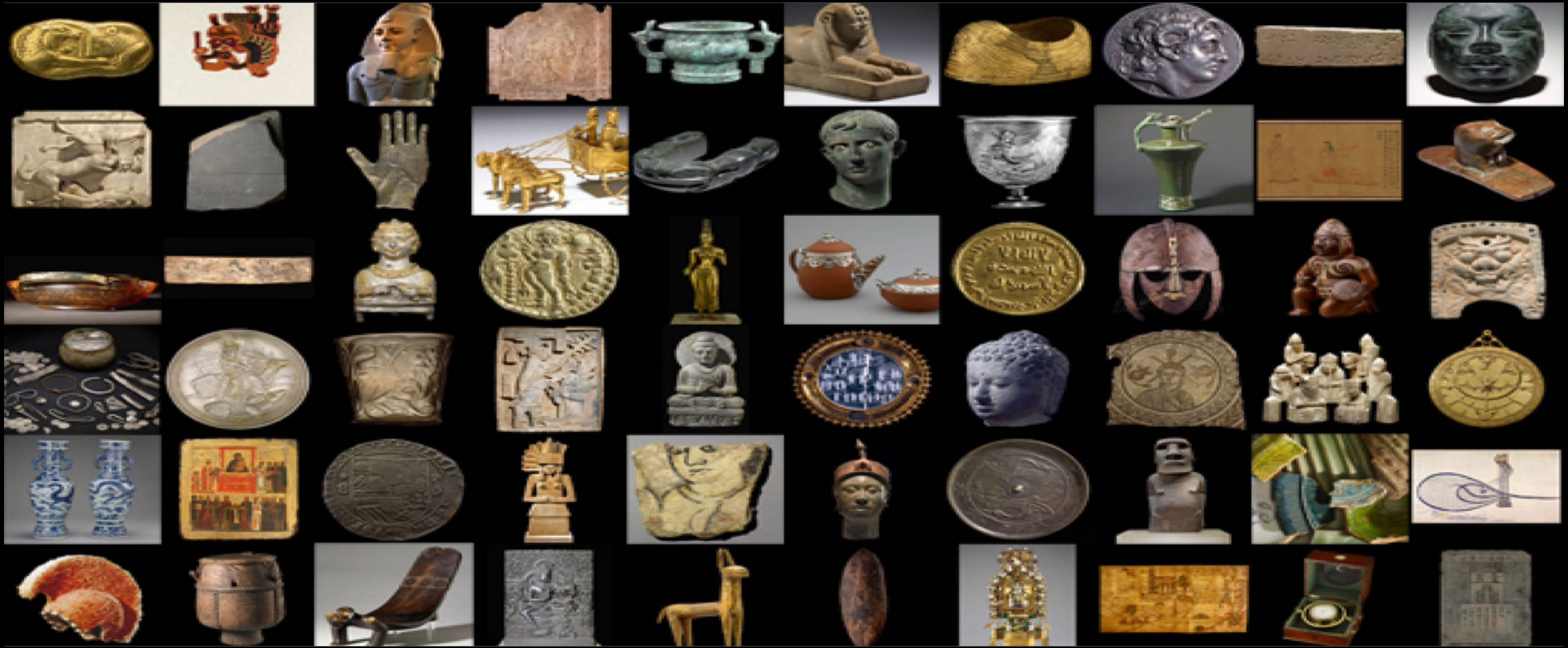
29%
New Features

53%
Maintenance, Complexity

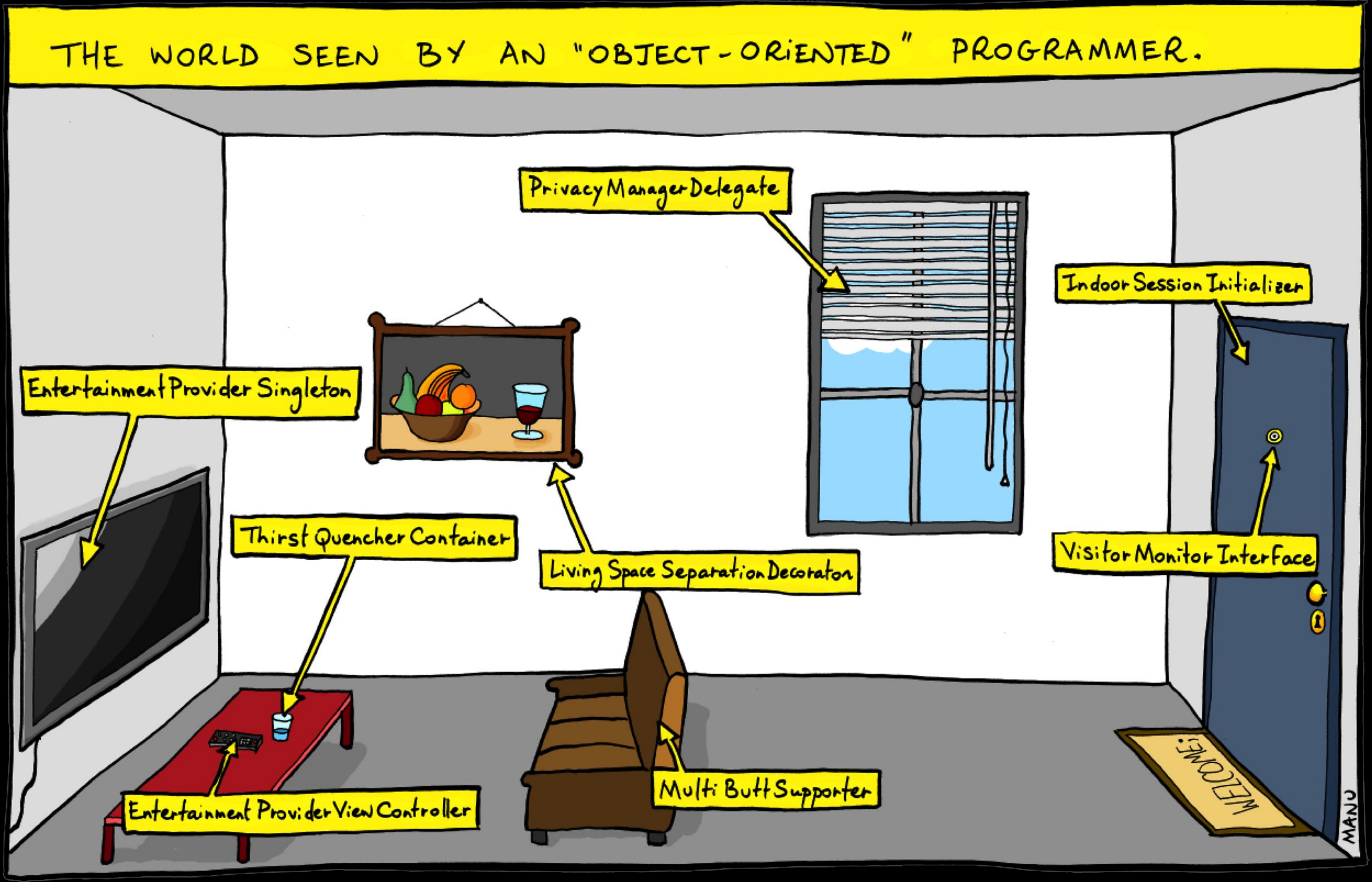
18%
“Professionalization”

Fahd Al-Fatish: Software Engineering - the roots, Karlsruher Entwicklertage 2017

A World of Objects



(© Trustees of the British Museum)


THE WORLD SEEN BY AN "OBJECT-ORIENTED" PROGRAMMER.
PrivacyManagerDelegate
Indoor Session Initializer
Entertainment Provider Singleton
Thirst Quencher Container
Living Space Separation Decorator
Visitor Monitor Interface
Entertainment Provider View Controller
Multi Butt Supporter
WELCOME!
MANU

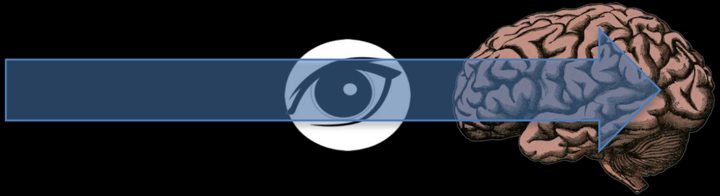
Source: http://bonkersworld.net/object-world (CC)

Imperative Programming

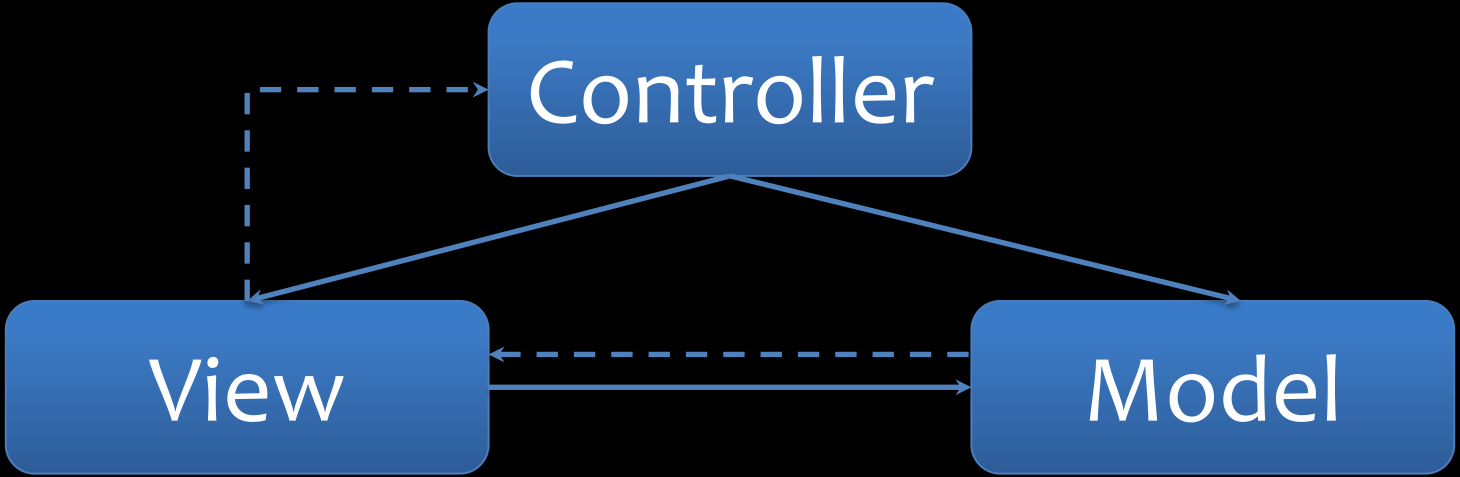
```
jungle.exit(elephant)
room.enter(elephant)
```

https://www.flickr.com/photos/tomsaint/3266056756

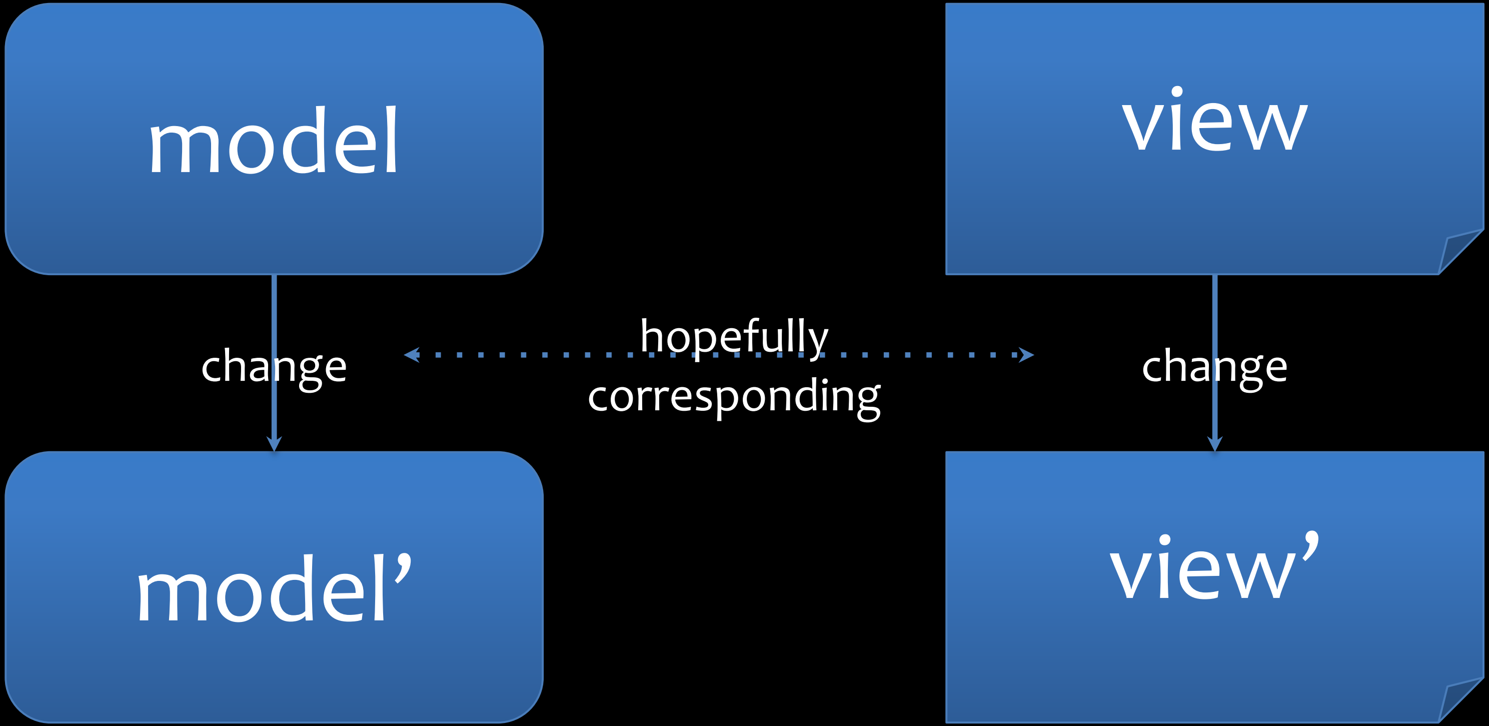
Reality and Snapshots



MVC



Problem


model

change

model’

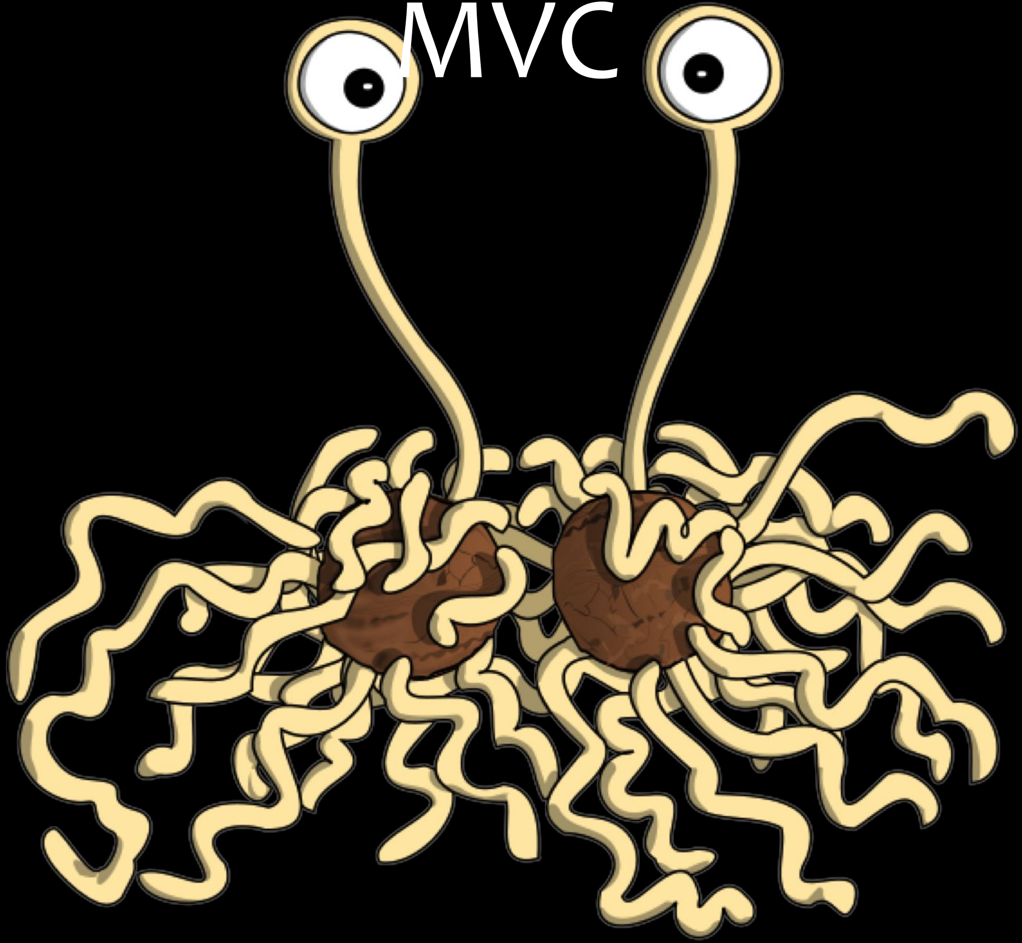
hopefully corresponding

view

change

view’

MVC

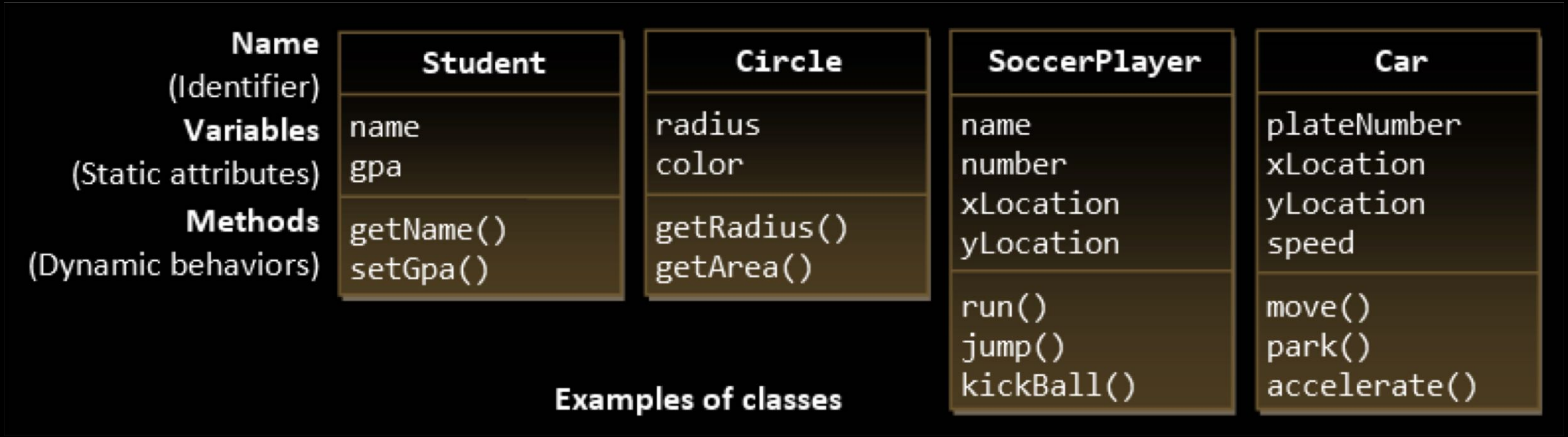


Alan Kay on OO

Though OOP came from many motivations, two were central. […] to find a more flexible version of assignment, and then to try to eliminate it altogether.

Alan Kay, *History of Smalltalk*
Communications of the ACM, 1996

OO vs. State



Name (Identifier)	**Student**	**Circle**	**SoccerPlayer**	**Car**
Variables (Static attributes)	name gpa	radius color	name number xLocation yLocation	plateNumber xLocation yLocation speed

Methods
(Dynamic behaviors)
	getName()
	setGpa()
		getRadius()
		getArea()
			run()
			jump()
			kickBall()
				move()
				park()
				accelerate()

Examples of classes

https://www.ntu.edu.sg/home/ehchua/programming/java/J3a_OOPBasics.html

FP for the Win!

- simpler languages
- less complexity
- higher productivity
- less bugs

FP for the Win!

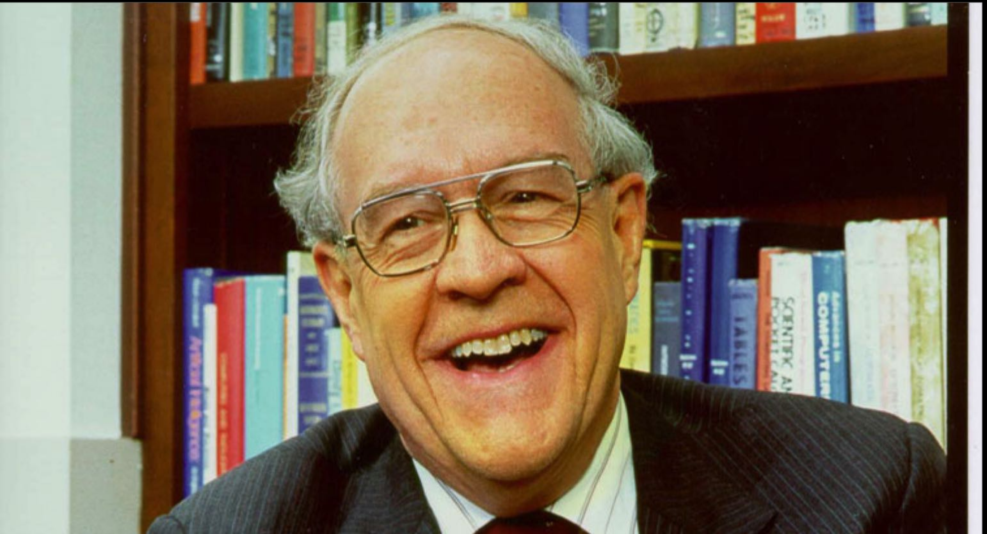
- simpler languages
- less complexity
- higher productivity
- less bugs
- powerful type systems
- property-based testing

FP for the Win!

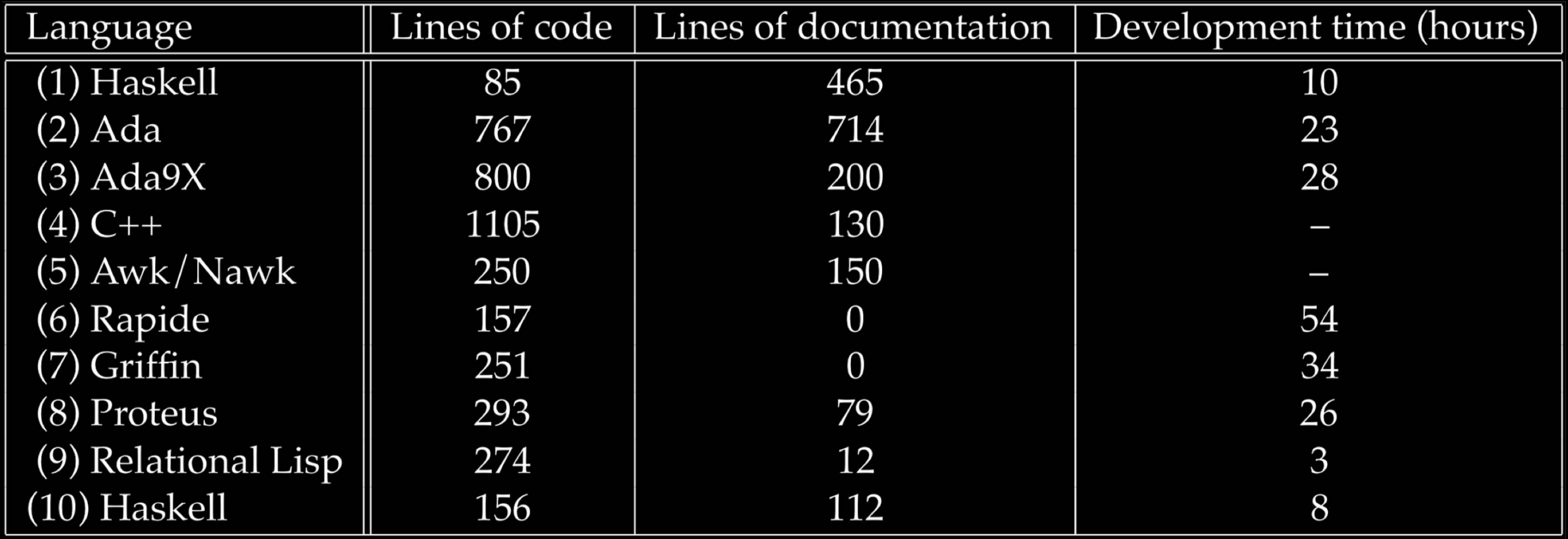
- simpler languages
- less complexity
- higher productivity
- less bugs
- powerful type systems
- property-based testing
- more predictable behavior
- easier testing
- lower coupling
- fewer dependency cycles

No Silver Bullet!

Fred Brooks *No Silver Bullet — Essence and Accidents of Software Engineering*
IEEE Computer, 1987

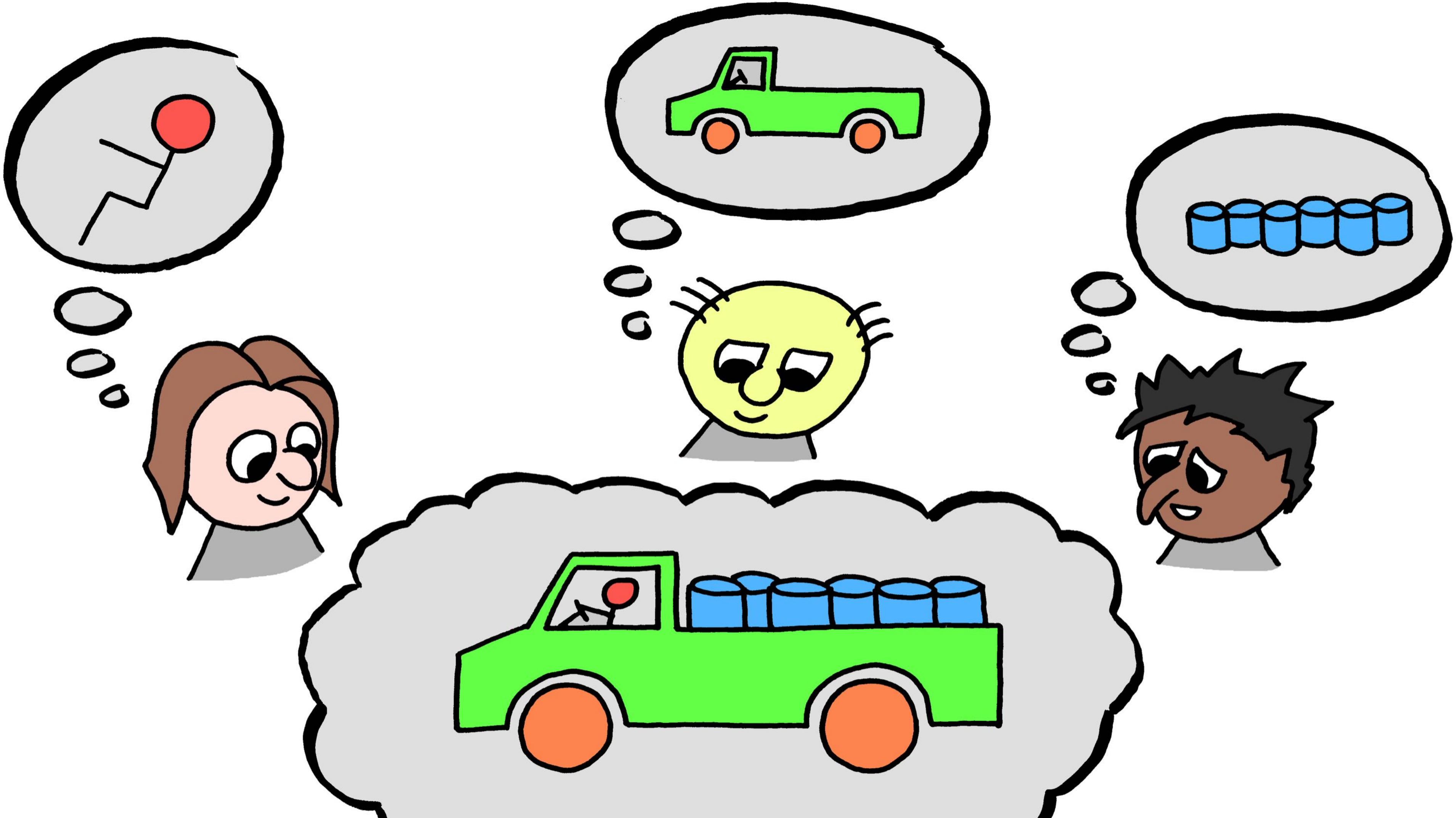


Yale Study

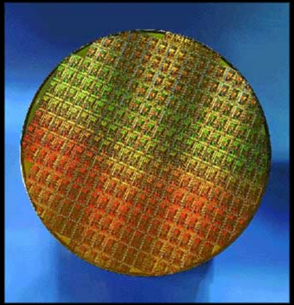


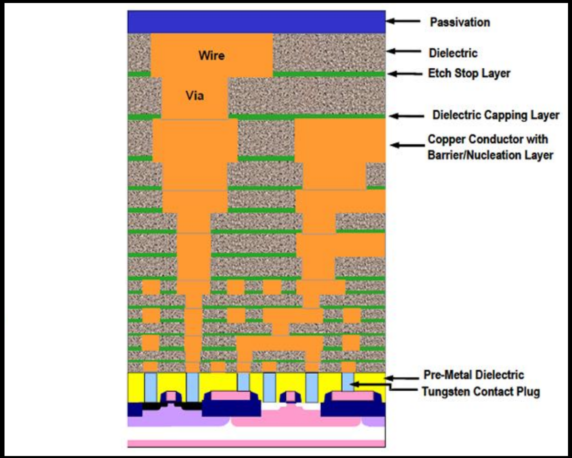
Language	Lines of code	Lines of documentation	Development time (hours)
(1) Haskell	85	465	10
(2) Ada	767	714	23
(3) Ada9X	800	200	28
(4) C++	1105	130	–
(5) Awk/Nawk	250	150	–
(6) Rapide	157	0	54
(7) Griffin	251	0	34
(8) Proteus	293	79	26
(9) Relational Lisp	274	12	3
(10) Haskell	156	112	8

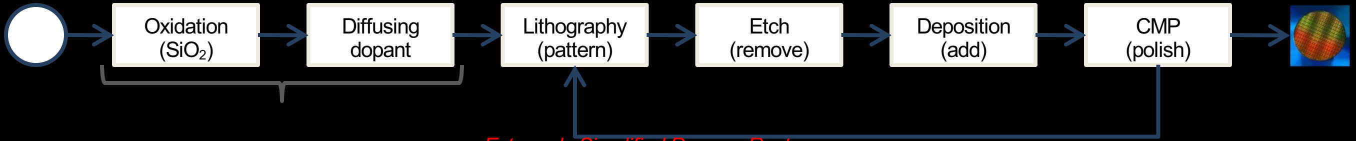
Hudak, Jones: Haskell vs. Ada vs. C++ vs. Awk vs. ...
An Experiment in Software Prototyping Productivity, Yale University, 1993



Semiconductor Manufacturing







Oxidation (SiO_2) → Diffusing dopant → Lithography (pattern) → Etch (remove) → Deposition (add) → CMP (polish) →

Extremely Simplified Process Route

Representation

```
data Operation =
  TrackIn | Process | TrackOut


type Route = [Operation]


r1 = [TrackIn, Process, Process, TrackOut]
```

Functions

```haskell
routeHead :: Route -> Operation

routeHead [] = ???
routeHead (op:_) = op
```

Optional Things

```
data Option a where
  Some :: a -> Option a
  None :: Option a
```

Operations

```
routeHead :: Route -> Option Operation

routeHead [] = None
routeHead (op:_) = Some op

routeHead r1 ⇒ Some TrackIn
```

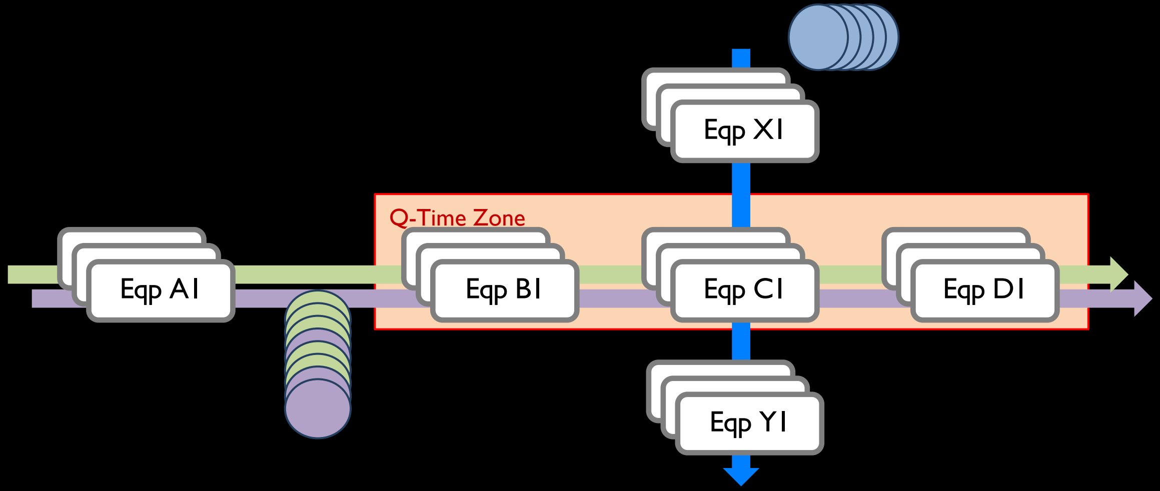
Operations

```
routeAdvance ::
   Route -> Option (Operation, Route)

routeAdvance r1
⇒  Some (TrackIn,
         [Process,Process,TrackOut])
```

Operations

```
routeAdvance ::
  Route -> Option (Operation, Route)

routeAdvance [] = None
routeAdvance (op:rest) = Some (op, rest)
```

Q-Time Zones



Per Route:

- 1000 Operations
- 50 separate Q-Time zones

Q-Time Zones

```
type Route = [RouteElement]

data RouteElement where
  RouteOp ::
    Operation -> RouteElement
  RouteQTZone ::
    Duration -> [Operation] -> RouteElement
```

Examples

```
r1 = [RouteOp TrackIn,
      RouteOp Process, RouteOp Process,
      RouteOp TrackOut]

r2 = [RouteOp TrackIn,
       RouteQTZone 5 [Process, Process],
       RouteOp TrackOut]
```

Q-Time Zones

```
type Route = [RouteElement]


data RouteElement where
  RouteOp ::
    Operation -> RouteElement
  RouteQTZone ::
    Duration -> [Operation] -> RouteElement
```

Q-Time Zones

```
type Route = [RouteElement]


data RouteElement where
   RouteOp ::
      Operation -> RouteElement
   RouteQTZone ::
      Duration -> [RouteElement] -> RouteElement
```

Examples

```
r1 = [RouteOp TrackIn,
      RouteOp Process, RouteOp Process,
      RouteOp TrackOut]

r2 = [RouteOp TrackIn,
       RouteQTZone 5
         [RouteOp Process,
          RouteOp Process],
       RouteOp TrackOut]
```

Nested Q-Time Zones

```
r3 = [RouteOp TrackIn,
       RouteQTZone 5
       [RouteOp Process,
         RouteQTZone 7
         [RouteOp Process,
          RouteOp Process]]]
```

Queue-Time Zones

```
type Route = [RouteElement]


data RouteElement where
  RouteOp ::
    Operation -> RouteElement
  RouteQTZone ::
    Duration -> [RouteElement] -> RouteElement
```

Queue-Time Zones

```
type Route = [RouteElement]


data RouteElement where
  RouteOp ::
    Operation -> RouteElement
  RouteQTZone ::
    Duration -> Route -> RouteElement
```

Functions

```
routeHead :: Route -> Option Operation
routeHead [] = None
routeHead (el:rest) =
  case routeElementHead el of
    None -> routeHead rest
    Some op -> Some op


routeElementHead :: RouteElement -> Option Operation
routeElementHead (RouteOp op) = Some op
routeElementHead (RouteQTZone _ rt) = routeHead rt
```

Advance

```
routeAdvance ::
  Route -> Time ->  Option (Operation, Route)

routeAdvance [] t = None
routeAdvance (el:rest) t =
  case el of
    RouteOp op -> Some (op, rest)
    RouteQTZone d rt -> ???
```

Queue-Time Zones in Progress

```
type Route = [RouteElement]

data RouteElement where
  RouteOp :: Operation -> RouteElement
  RouteQTZone :: Duration -> Route -> RouteElement
  RouteQTLimit :: Time -> Route -> RouteElement
```

Advance

```
routeAdvance :: Route -> Time ->  Option (Operation, Route)
routeAdvance [] t = None
routeAdvance (el:rest) t =
  case el of
    RouteOp op -> Some (op, rest)
    RouteQTZone d rt ->
      routeAdvance (RouteQTLimit (t + d) rt : rest) t
    RouteQTLimit tl rt  ->
      case routeAdvance rt t of
        None -> routeAdvance (rest) t
        Some (op, rt') ->
          Some (op, RouteQTLimit tl rt'  : rest)
```

Q-Time Zones in Progress

```
type Route = [RouteElement]

data RouteElement where
  RouteOp :: Operation -> RouteElement
  RouteQTZone :: Duration -> Route -> RouteElement
  RouteQTLimit :: Time -> Route -> RouteElement
```

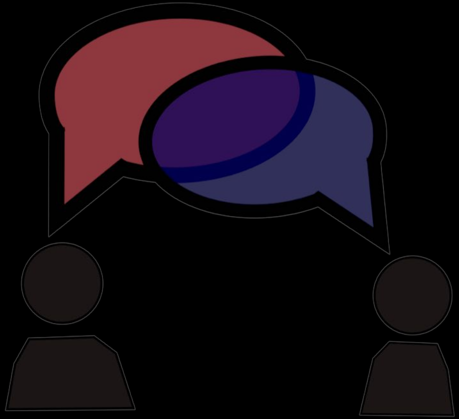
Invalid State

```
r4 = [RouteOp TrackIn,
      RouteQTLimit 12
      [RouteOp Process,
       RouteOp TrackOut]]
```

Making Invalid States Unrepresentable

```haskell
data Route where
  Route :: RouteRem -> Route
  RouteQTLimit :: Time -> Route -> RouteRem -> Route

type RouteRem = [RouteElement]

data RouteElement where
  RouteOp :: Operation -> RouteElement
  RouteQTZone :: Duration -> RouteRem -> RouteElement
```

λ +



=

https://pixabay.com/en/conversation-dialogue-interview-1262311/

https://commons.wikimedia.org/wiki/File:Bullet_train.jpg