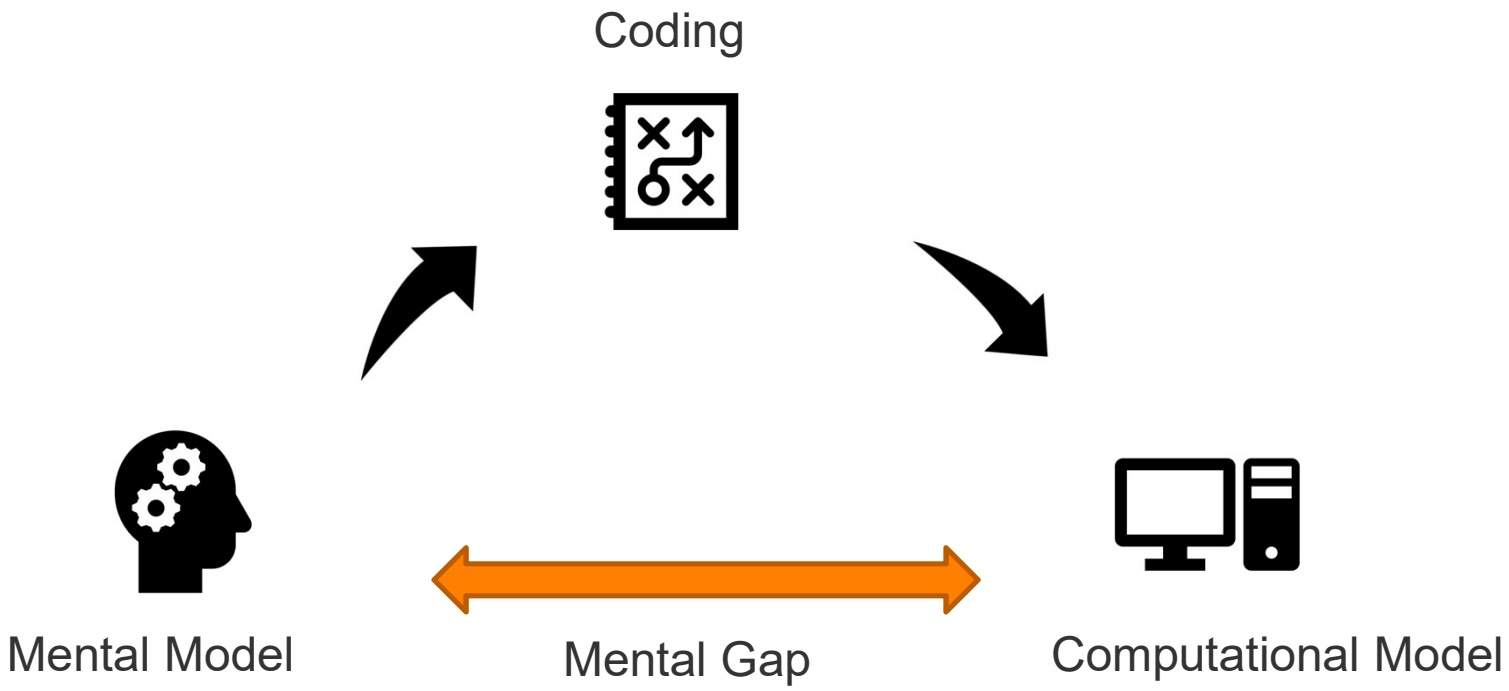
Active Expressions
Basic Building Blocks for Reactive Programming

Stefan Ramson and Robert Hirschfeld

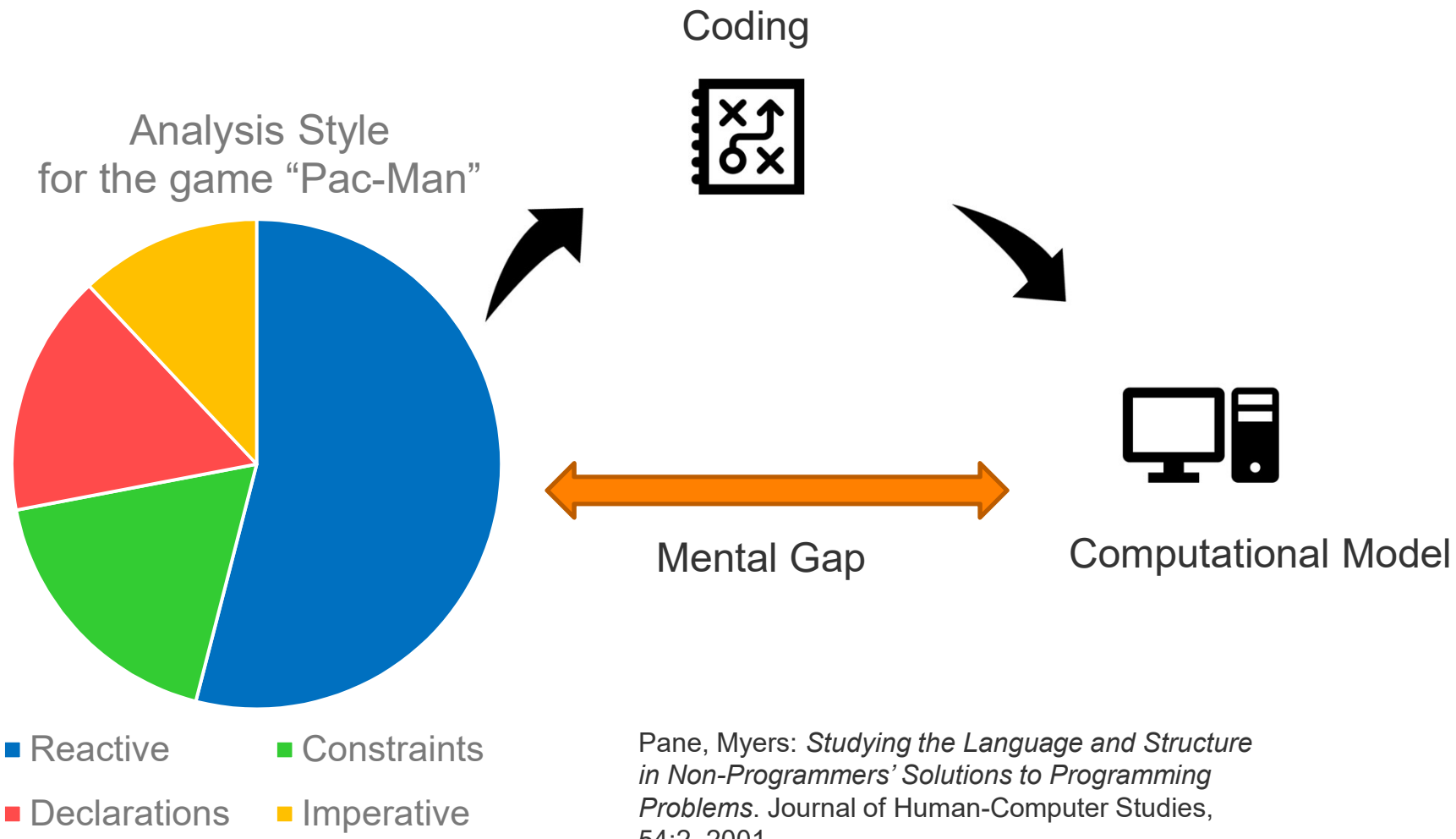
Hasso Plattner Institute Potsdam
Software Architecture Group

http://www.hpi.uni-potsdam.de/swa/

Programming: Manifesting a Mental Model as Executable Code

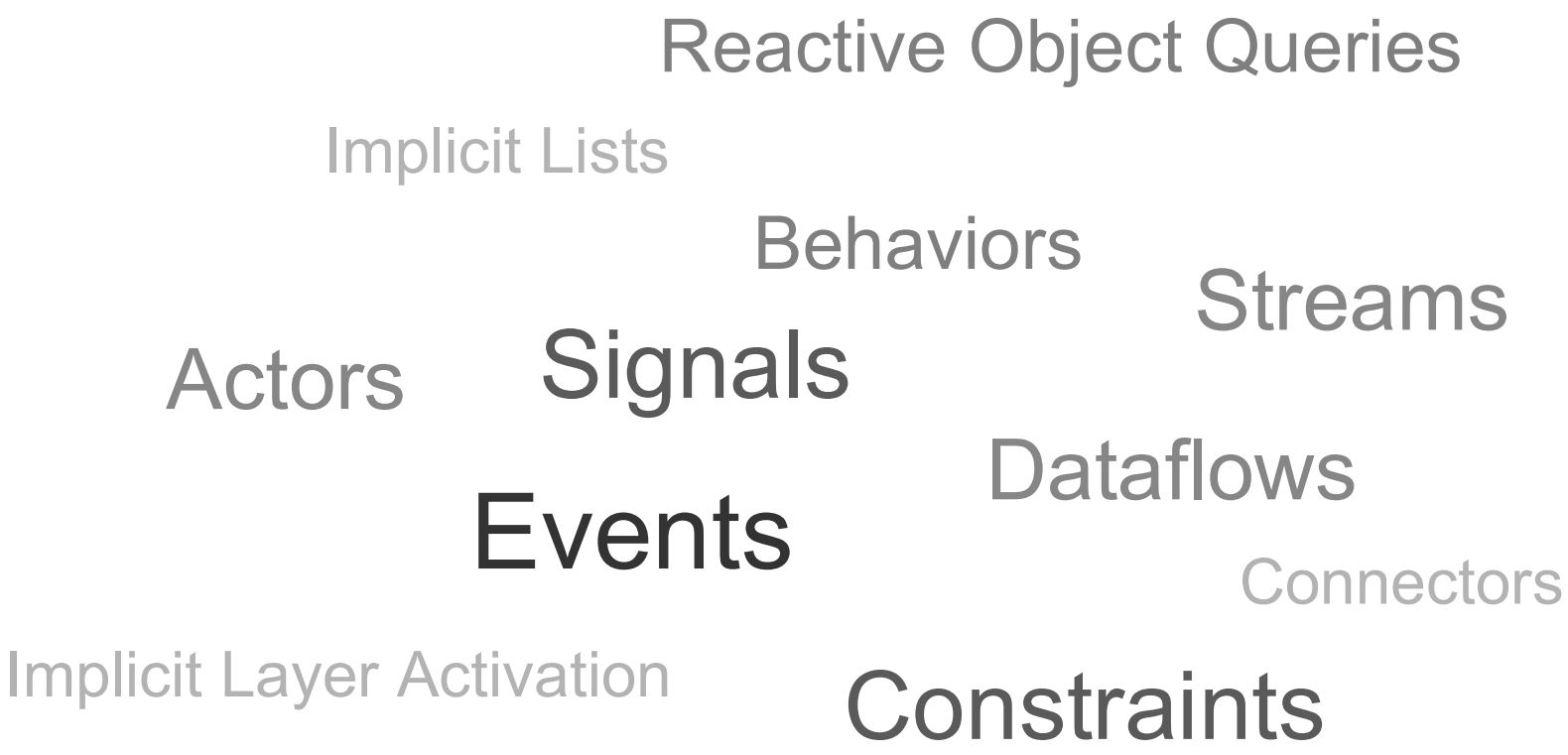


Programming: Manifesting a Mental Model as Executable Code



Pane, Myers: *Studying the Language and Structure in Non-Programmers’ Solutions to Programming Problems*. Journal of Human-Computer Studies, 54:2, 2001

The Variety of Reactive Concepts



The Structure of Reactive Programming Concepts

The Structure of Reactive Programming Concepts

- Concept-specific
- Visible for application developer

The Structure of Reactive Programming Concepts

- Conceptually exchangeable
- Hidden to application developer

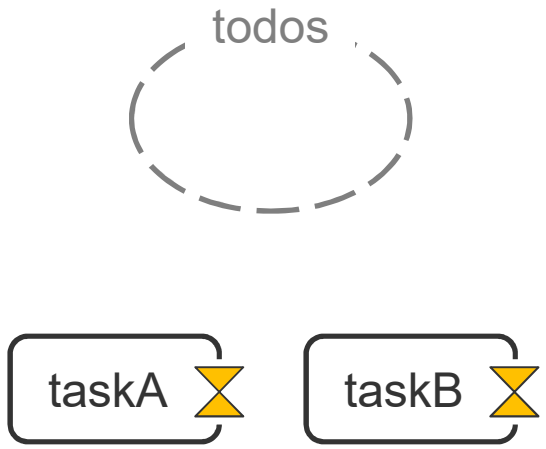
- Concept-specific
- Visible for application developer

Reactive Object Queries
Working Principle

```
let todos = select(Task,
    t => !t.done()
);
taskA in todos; // true
// ...
taskA.finish();
taskA in todos; // false
```

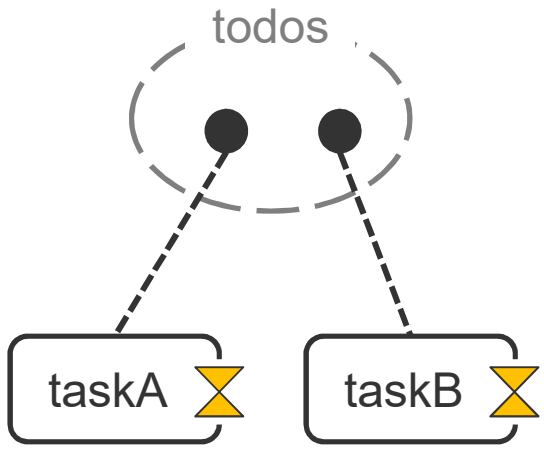
Reactive Object Queries
Working Principle

```
let todos = select(Task,
    t => !t.done()
);
taskA in todos; // true
// ...
taskA.finish();
taskA in todos; // false
```



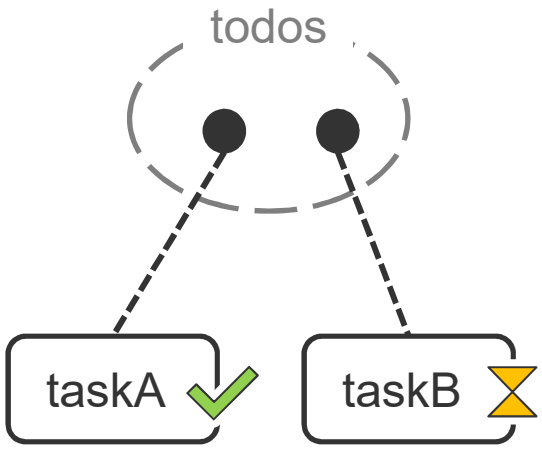
Reactive Object Queries
Working Principle

```
let todos = select(Task,
    t => !t.done()
);
taskA in todos; // true
// ...
taskA.finish();
taskA in todos; // false
```



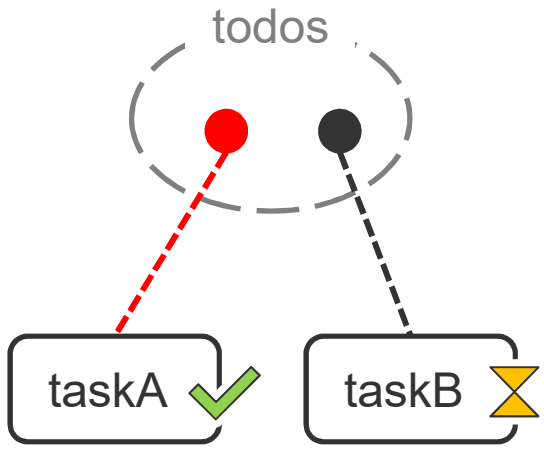
Reactive Object Queries
Working Principle

```
let todos = select(Task,
    t => !t.done()
);
taskA in todos; // true
// ...
taskA.finish();
taskA in todos; // false
```



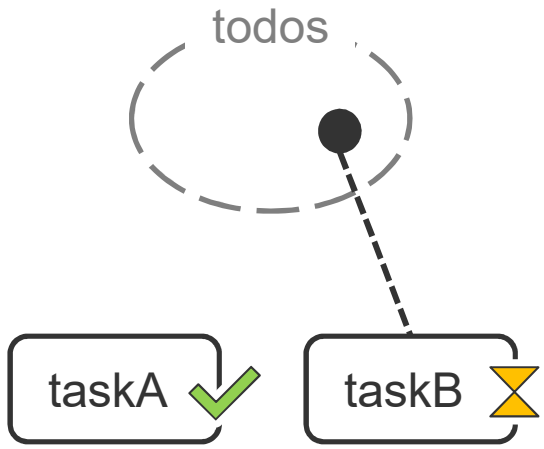
Reactive Object Queries
Working Principle

```
let todos = select(Task,
    t => !t.done()
);
taskA in todos; // true
// ...
taskA.finish();
taskA in todos; // false
```



Reactive Object Queries
Working Principle

```
let todos = select(Task,
    t => !t.done()
);
taskA in todos; // true
// ...
taskA.finish();
taskA in todos; // false
```



Implementing Reactive Object Queries
Reactive Behavior

```
if (condition(item))
  group.add(item);
else
  group.remove(item);
```

https://github.com/onsetsu/active-collection-prototype

Implementing Reactive Object Queries
Change Detection


```
installListeners() {
  expressionObserverStack.withElement(this,
    () => ExpressionInterpreter.runAndReturn(
      this.condition,
      this.context,
      this.item
    )
  );
}
```

https://github.com/onsetsu/active-collection-prototype

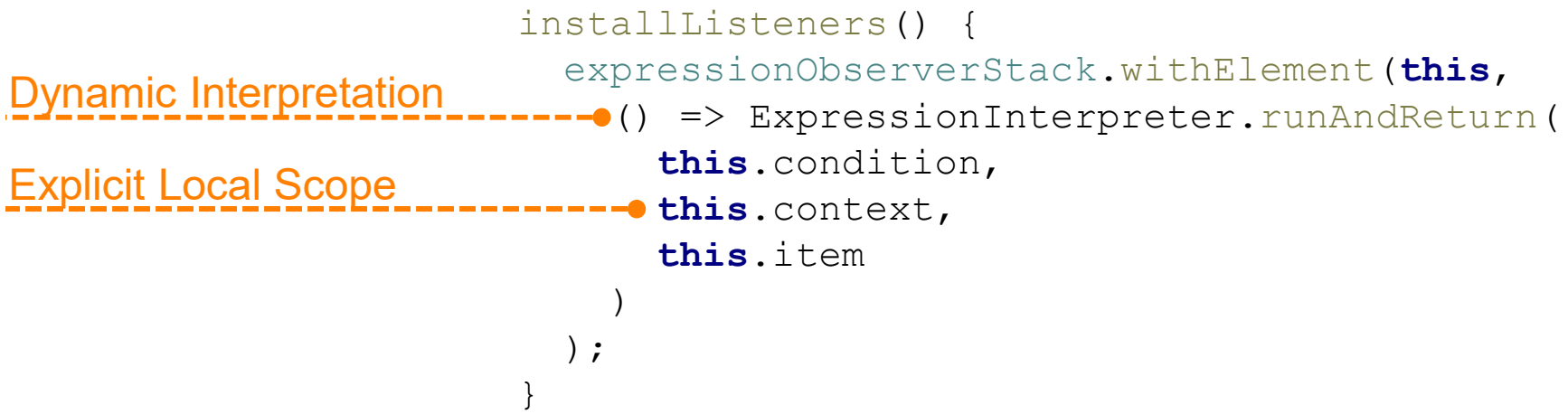
Implementing Reactive Object Queries
Change Detection

Dynamic Interpretation

```
installListeners() {
  expressionObserverStack.withElement(this,
    () => ExpressionInterpreter.runAndReturn(
      this.condition,
      this.context,
      this.item
    )
  );
}
```

https://github.com/onsetsu/active-collection-prototype

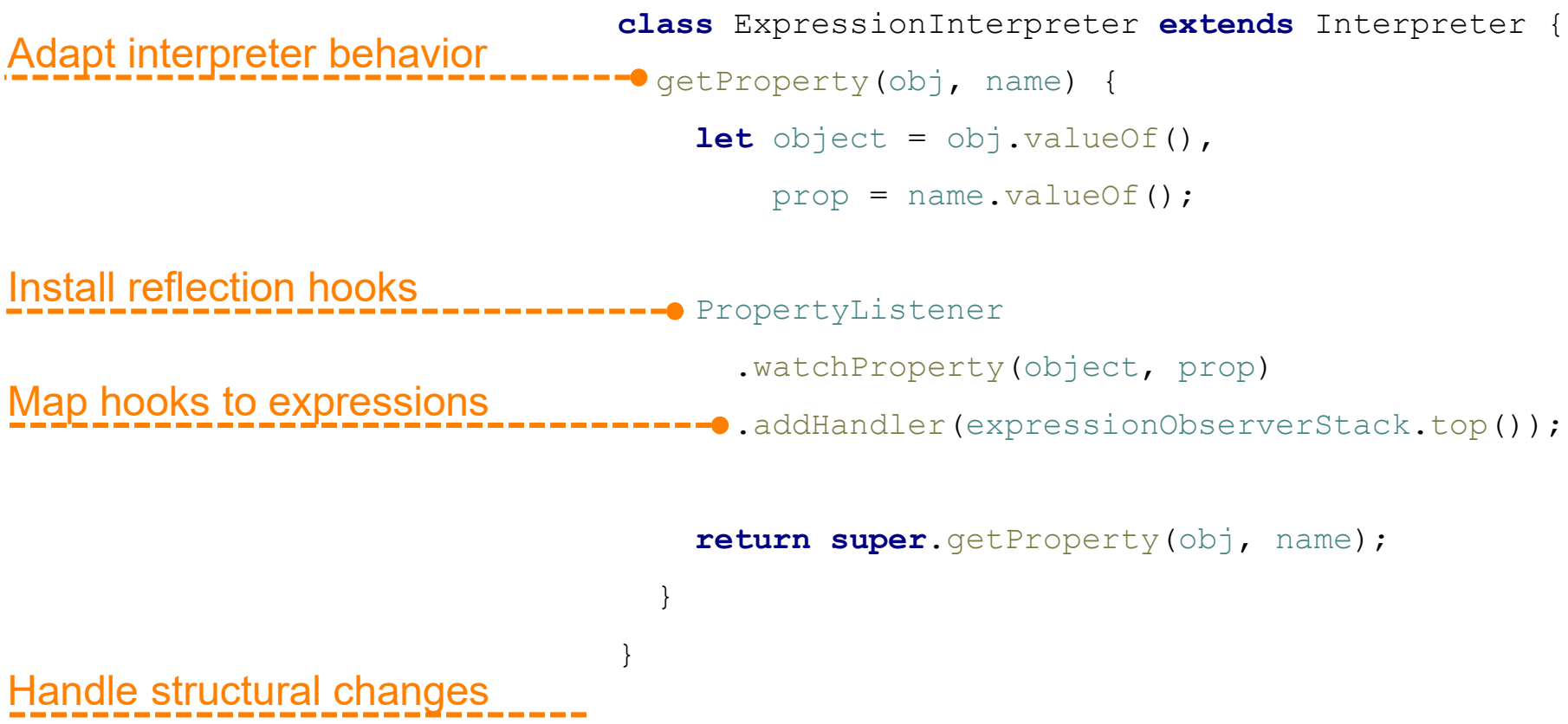
Implementing Reactive Object Queries
Change Detection




```
installListeners() {
  expressionObserverStack.withElement(this,
    () => ExpressionInterpreter.runAndReturn(
      this.condition,
      this.context,
      this.item
    )
  );
}
```

https://github.com/onsetsu/active-collection-prototype

Implementing Reactive Object Queries
Change Detection cont.




```
class ExpressionInterpreter extends Interpreter {
  getProperty(obj, name) {
    let object = obj.valueOf(),
        prop = name.valueOf();

    PropertyListener
      .watchProperty(object, prop)
      .addHandler(expressionObserverStack.top());

    return super.getProperty(obj, name);
  }
}
```

https://github.com/onsetsu/active-collection-prototype

Implementing Reactive Object Queries
Change Detection cont.

Interact with other concepts

Browser-specifics

```
this.safeOldAccessors(obj, propName);

try {
  obj.__defineGetter__(propName, () => this.propName);
} catch (e) { /* Firefox raises on Array.length */ }
let newGetter = obj.__lookupGetter__(propName);
if (!newGetter) { // Chrome silently ignores Array.length
  return;
}
```

https://github.com/onsetsu/active-collection-prototype

Implementing Reactive Object Queries
Change Detection

Difficulties:

- Vertical slice through multiple layers of abstraction
- Detection mechanism is not designed for reuse
- Detection limited to object properties

Synopsis

Problem: Change detection as a tedious but inevitable necessity for practical implementations

Goal: Ease the development of novel reactive programming concepts

Approach: Identify and exploit commonalities in existing reactive concepts

State-based Reactive Concepts

A subset of reactive programming concepts

Criteria: Dependencies specified implicitly as expressions over program state

Working Principle: The reactive framework identifies and monitors relevant state

State-based Reactive Concepts

		Monitors	Reaction
Signals	`signal s = expr`	Expression (Signal definition)	Update signal network
Constraints	`always: { expr }`	Expression (Constraint expression)	Solve constraints
Reactive Object Queries	`select(Class, expr)`	Expression (Group condition)	Update group membership
Implicit Layer Activation	`layer.when(expr)`	Expression (Layer condition)	Update layer composition
Two-way Data Bindings	`<tag value={expr}>`	Expression (Model)	Update view
Implicit Lists	`list.map(expr)`	Expression (Base list, iterators)	Update derived lists

Active Expressions
Approach

Reify identified commonality into a reusable concept

	Implementation	Monitors	Reaction
Signals	Implicit lifting	Expression (Signal definition)	Update signal network
Constraints	Alternate VM	Expression (Constraint expression)	Solve constraints
Reactive Object Queries	Reflection	Expression (Group condition)	Update group membership
…	…	…	…

Active Expressions
Approach

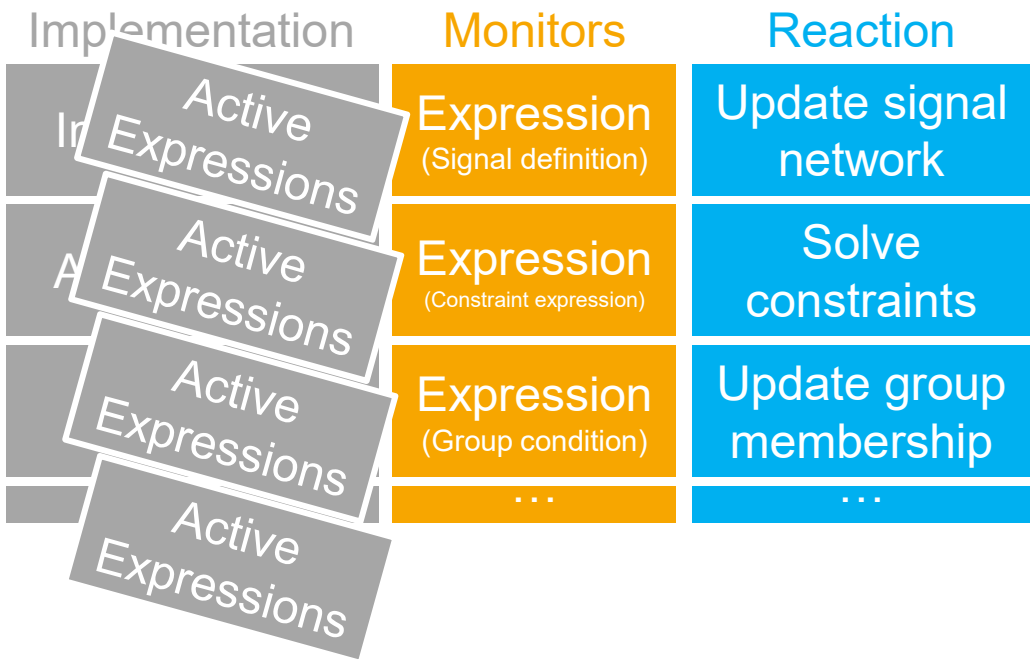
Reify identified commonality into a reusable concept

Signals

Constraints

Reactive Object Queries

…



Active Expressions
Design Perimeters

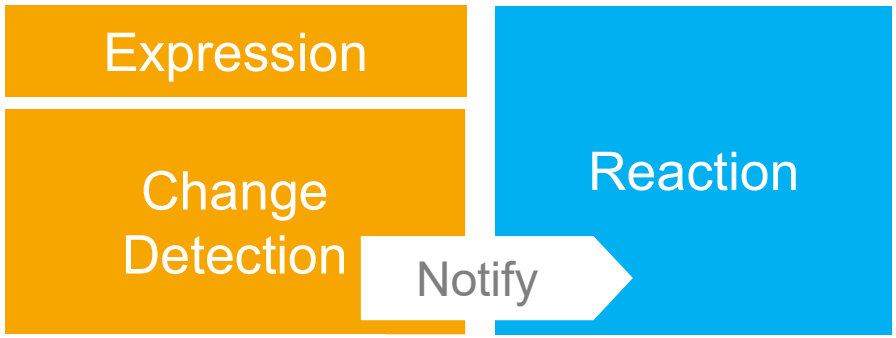
1. Ease change detection by hiding technology-specific implementation details

2. Support a variety of reactive behavior

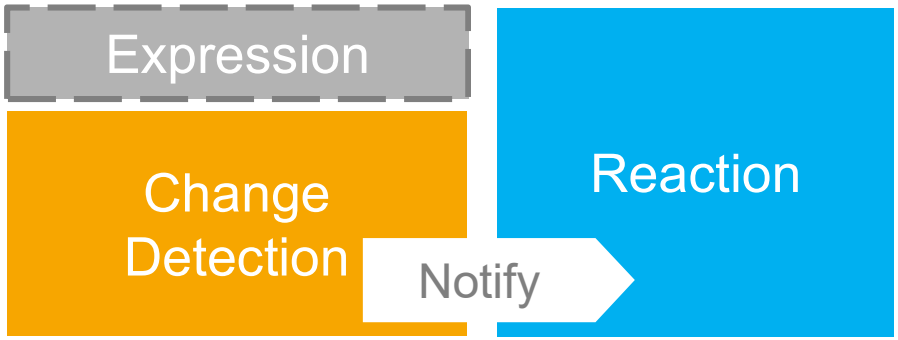
3. Integrate with object-oriented environments

Active Expressions
Design

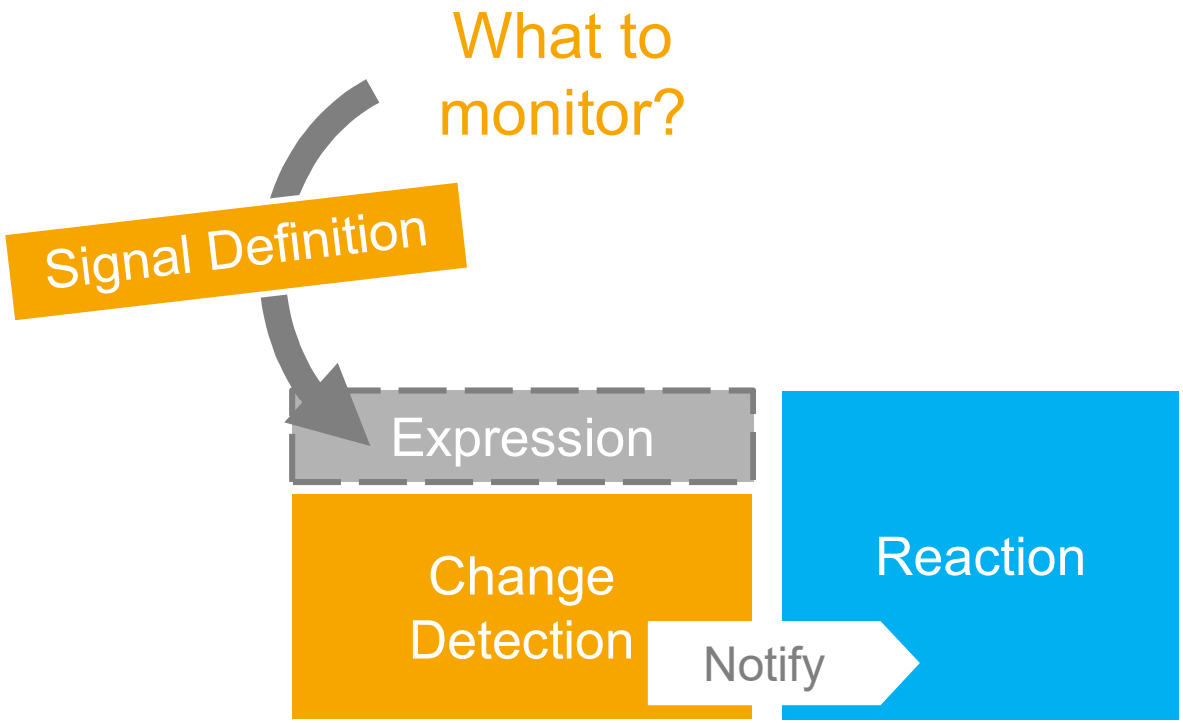
Active Expressions as a state-based reactive concept



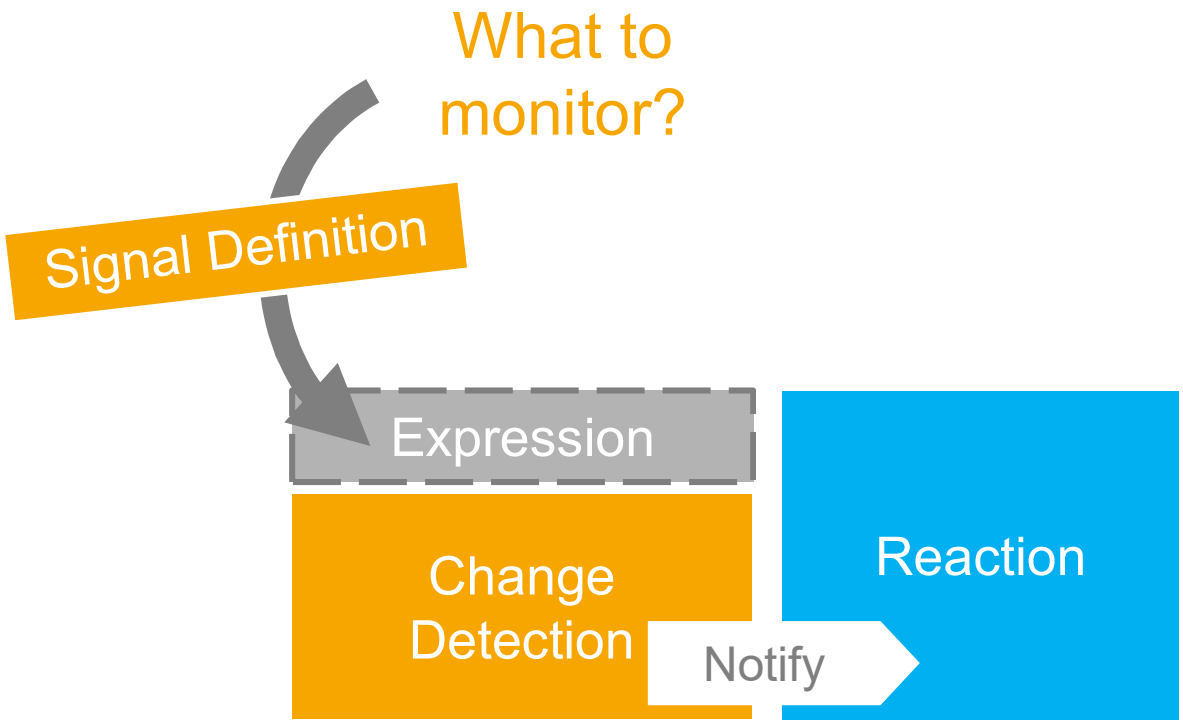
Active Expressions
Design – Change Detection



Active Expressions
Design – Change Detection



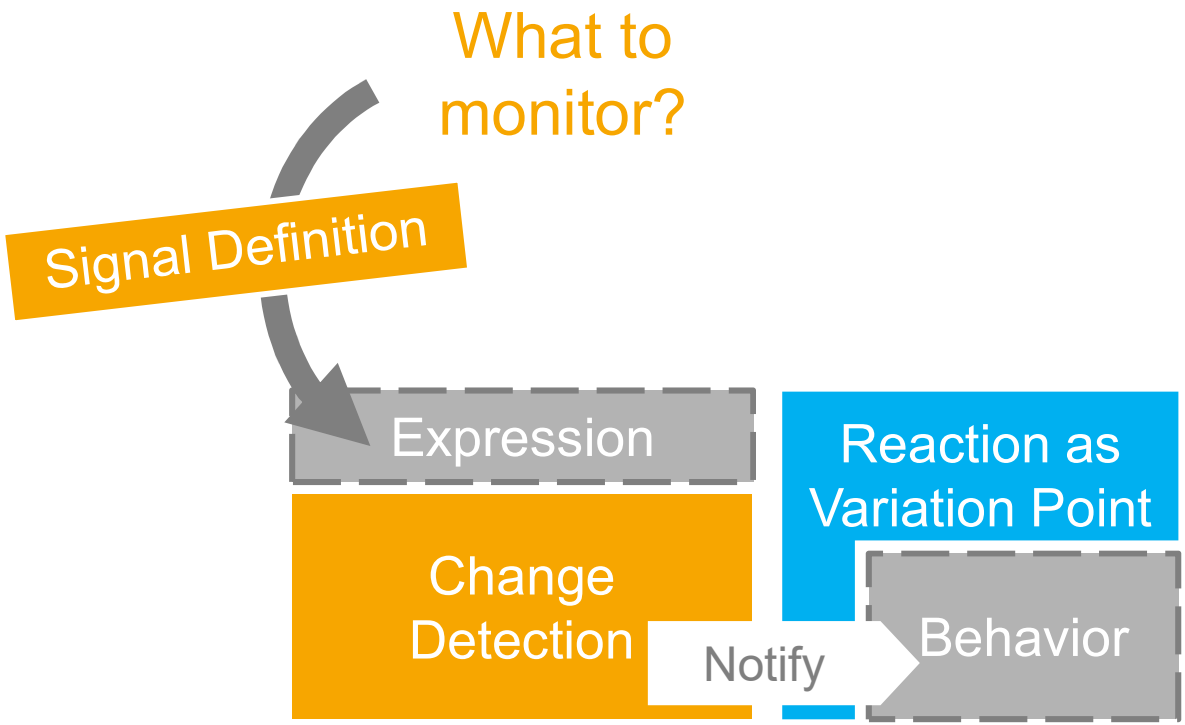
Active Expressions
Design – Change Detection




```
signal s = expr;
```

```
aexpr(() => expr)
```

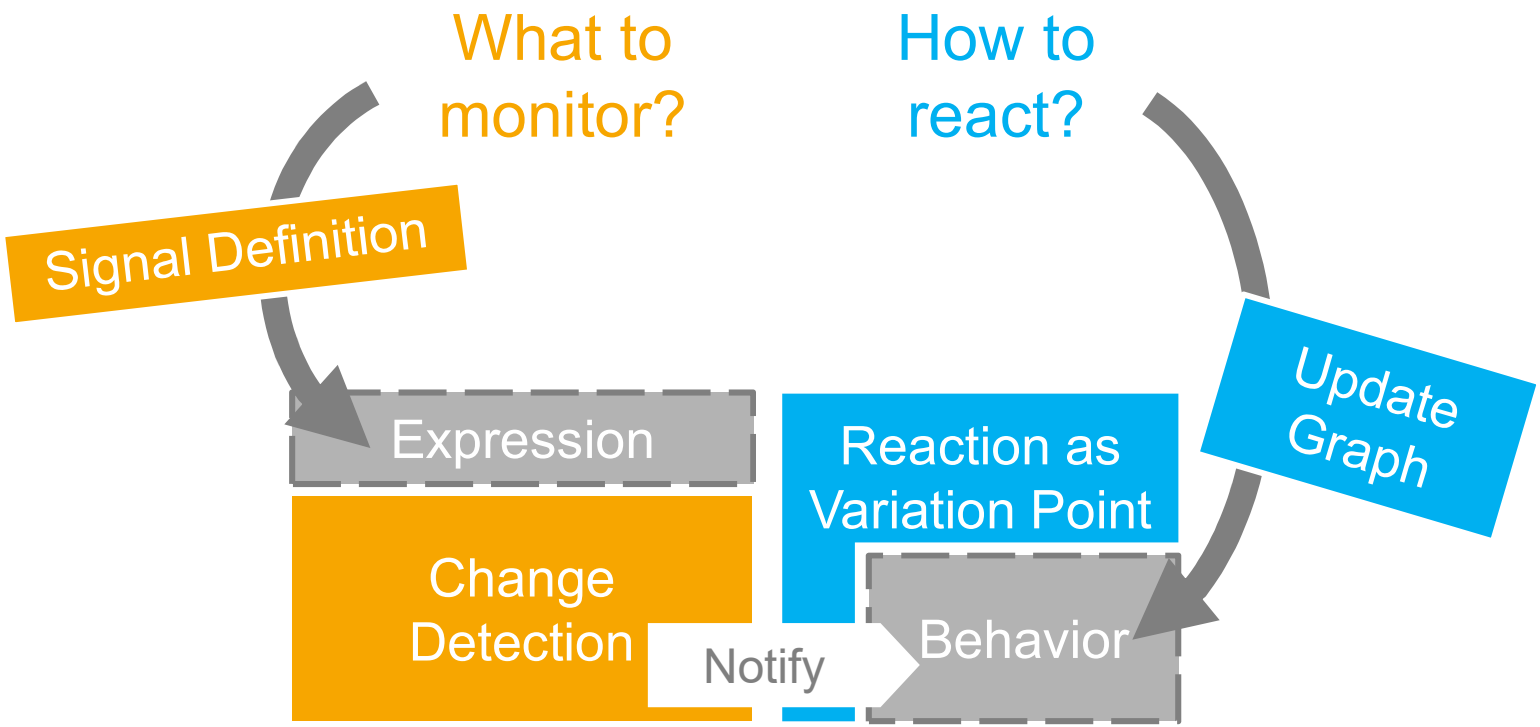
Active Expressions
Design – Reactive Behavior




```
signal s = expr;
```

```
aexpr(() => expr)
```

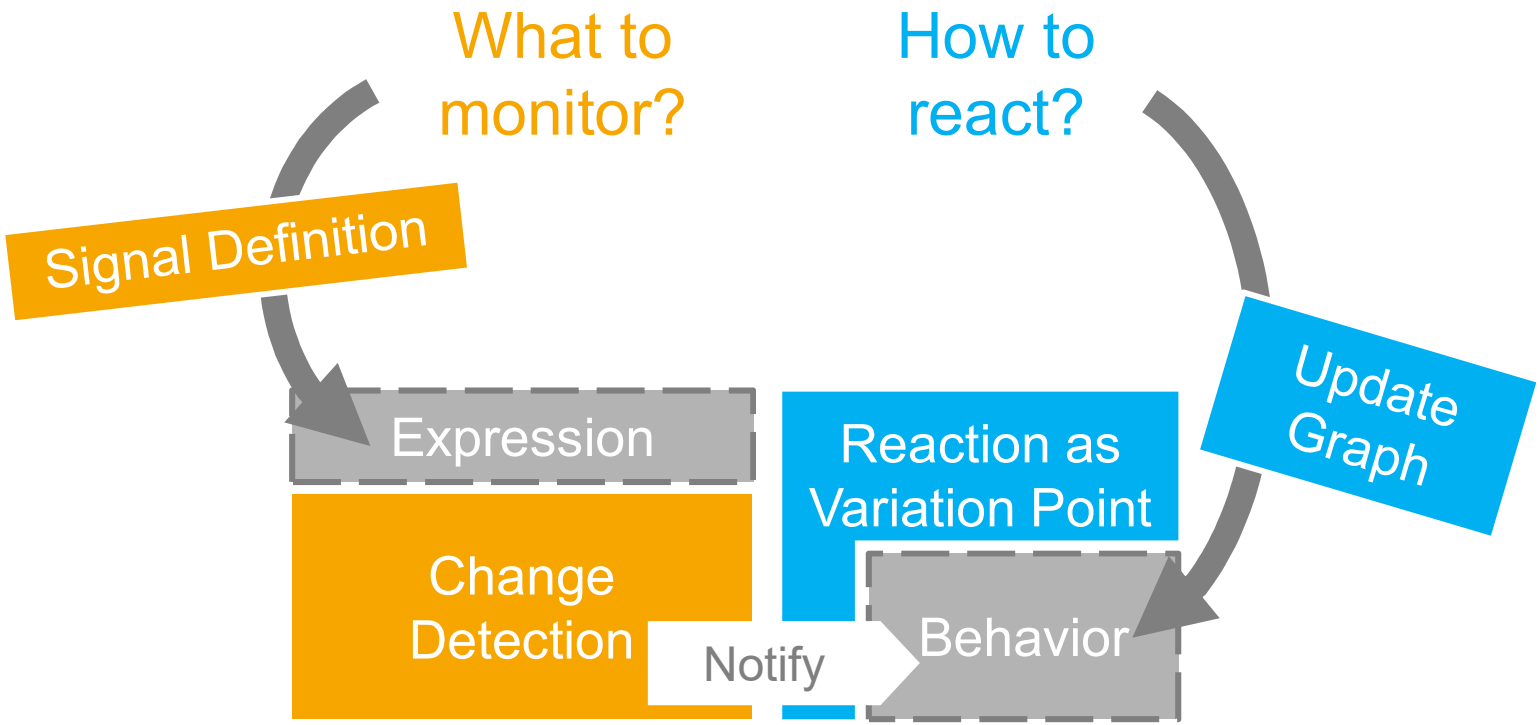
Active Expressions
Design – Reactive Behavior




```
signal s = expr;
```

```
aexpr(() => expr)
```

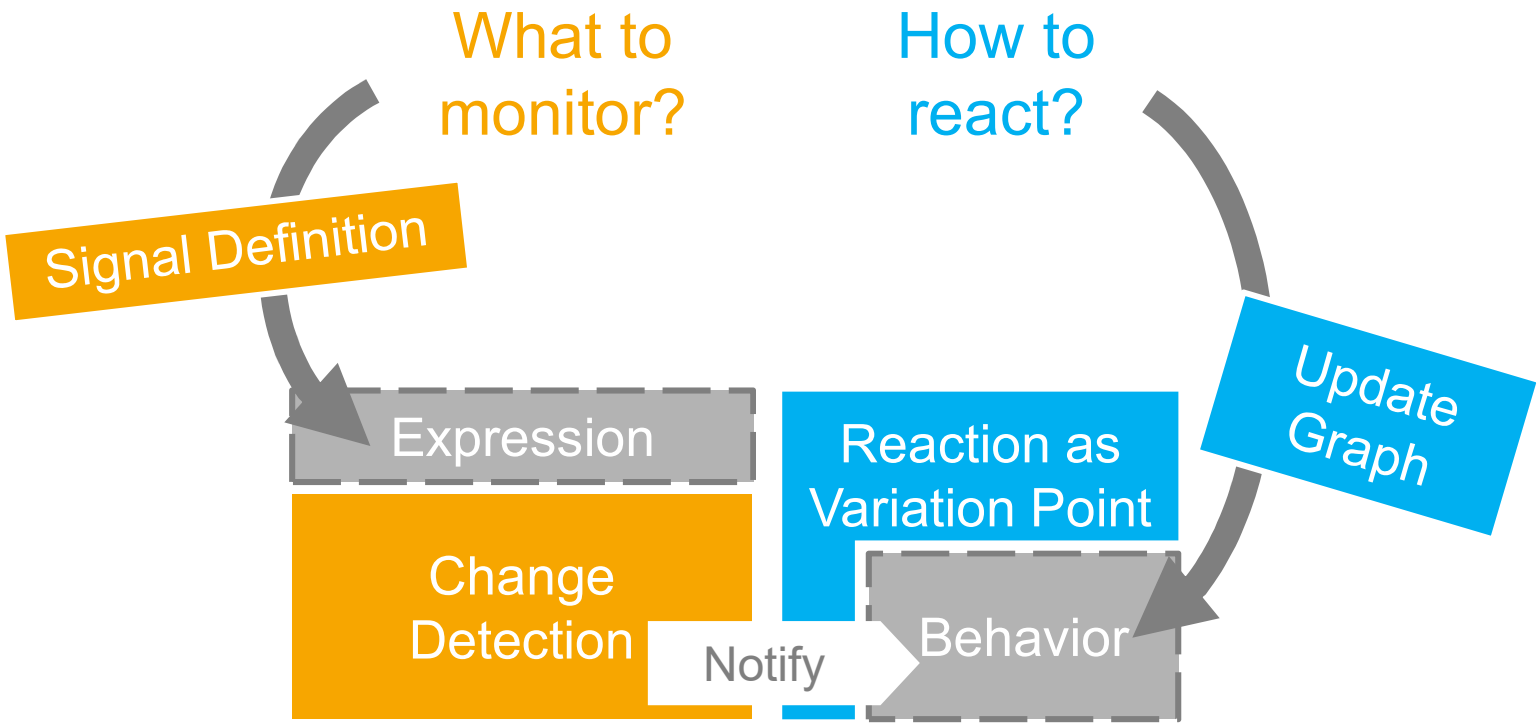
Active Expressions
Design – Reactive Behavior




```
signal s = expr;
```

```
aexpr(() => expr)
    .onChange(val => s = val)
```

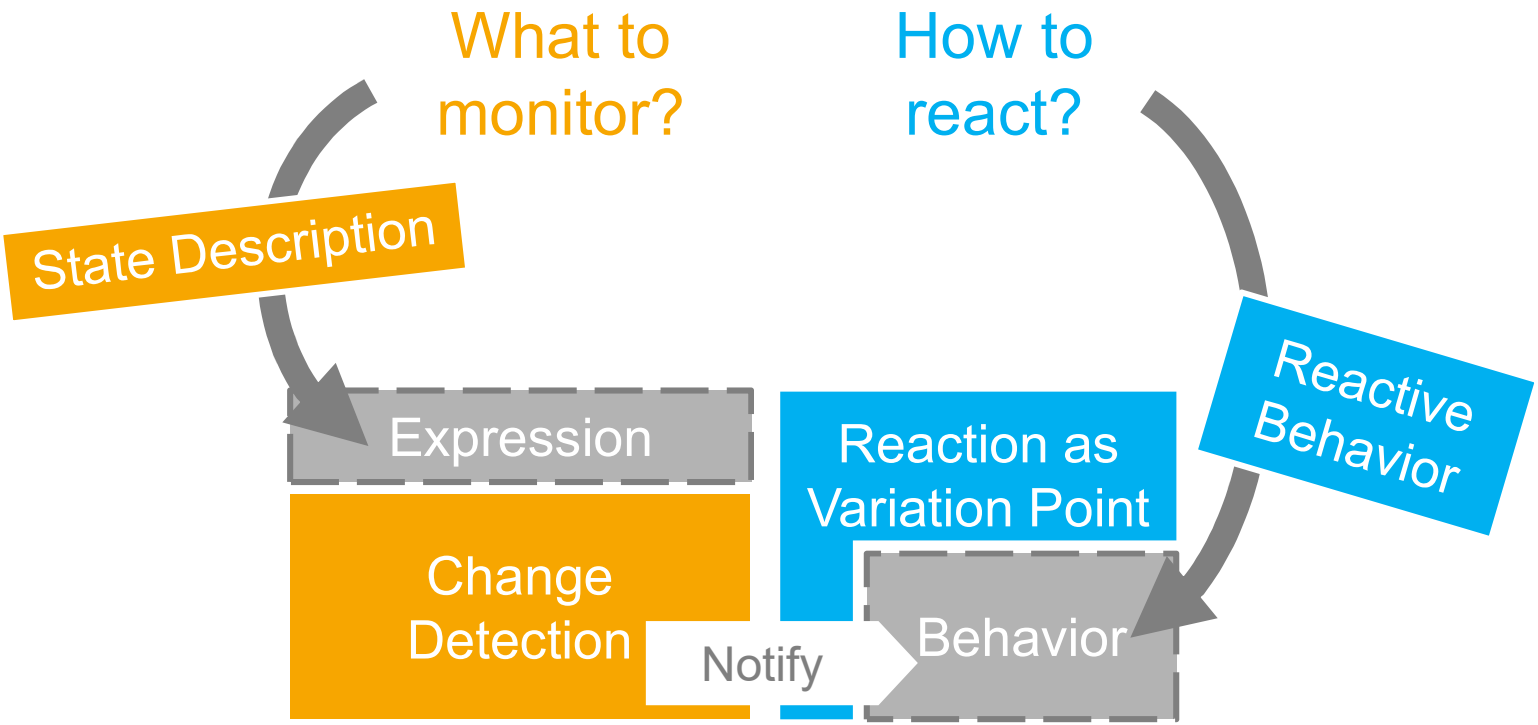
Active Expressions
Design




```
signal s = expr;
```

```
let s = aexpr(() => expr)
          .onChange(val => s = val)
          .now();
```

Active Expressions
Design




```
aexpr(expr).onChange(behavior)
```

Implementing Reactive Object Queries

```
group = select(Class, condition);
```

```
onNewInstance(item) {
  aexpr(() => condition(item))
    .onBecomeTrue(() => group.add(item))
    .onBecomeFalse(() => group.remove(item));
}
```

Applicability of Active Expressions

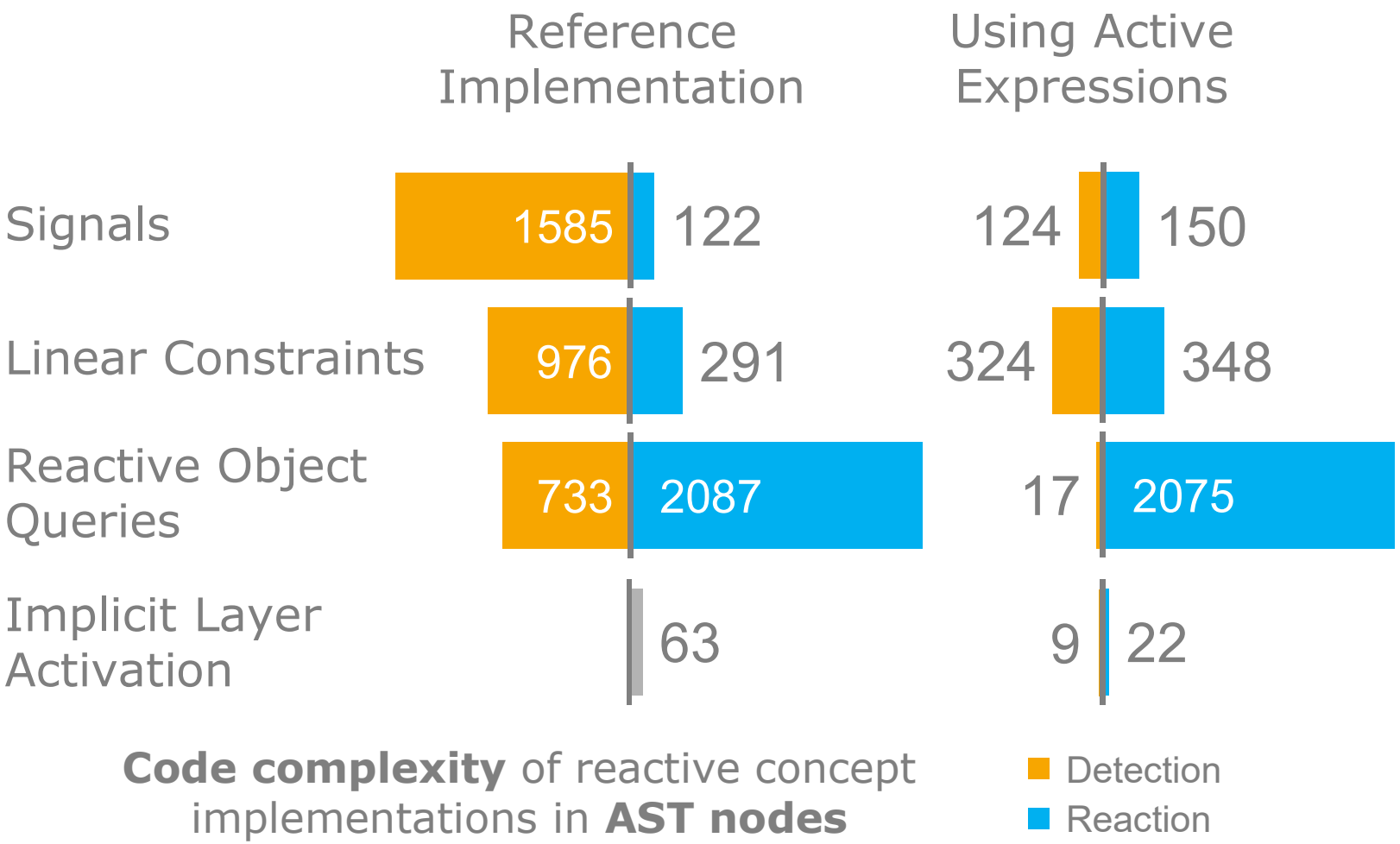
Signals

Linear Constraints

Reactive Object Queries

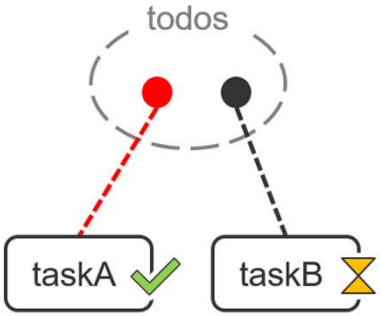
Implicit Layer Activation

Effects on Code Complexity



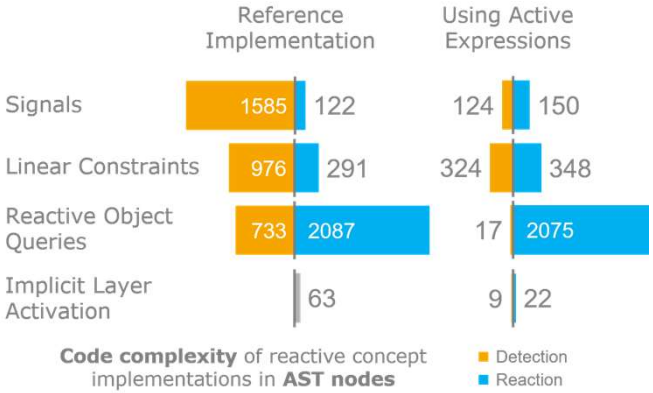
Code complexity of reactive concept implementations in **AST nodes**

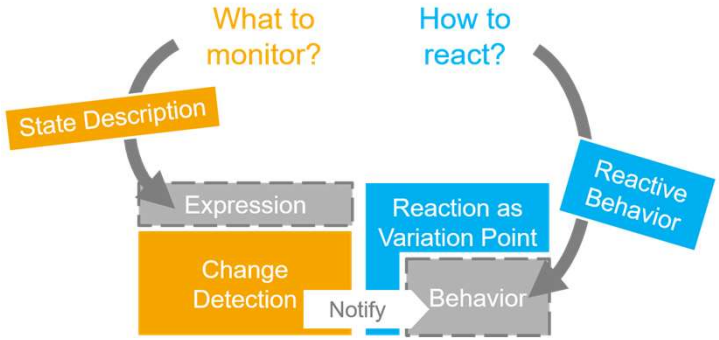
Summary



Problem: Change detection as a tedious but inevitable necessity for practical implementations

Goal: Ease the development of novel reactive programming concepts





Approach: Identify and exploit commonalities in existing reactive concepts