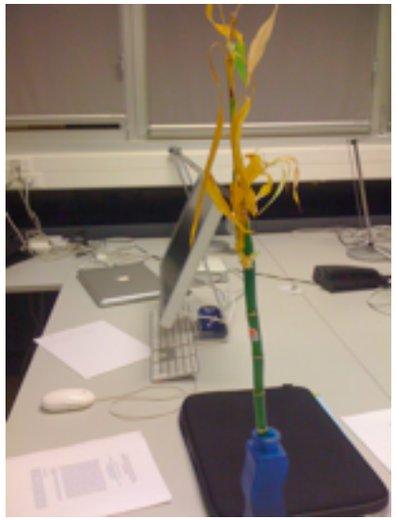
Interruptible Context-dependent Executions:
A Fresh Look at Programming Context-aware Applications

Engineer Bainomugisha, Jorge Vallejos, Coen De Roover,
Andoni Lombide Carreton and Wolfgang De Meuter

Software Languages Lab.
Vrije Universiteit Brussel, Belgium

True Context-aware Applications

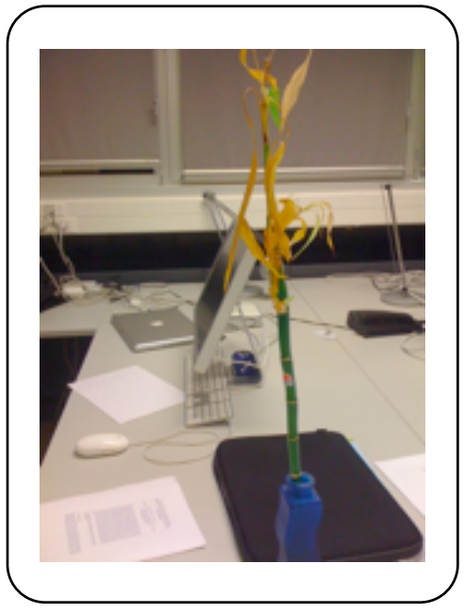


@Office

@Printer room

@Home

True Context-aware Applications

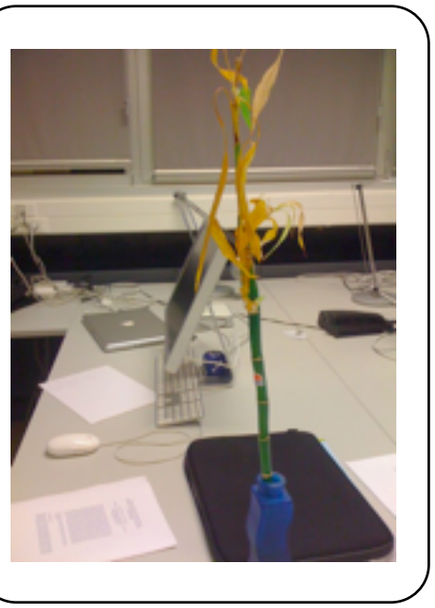


@Office

@Printer room

@Home

True Context-aware Applications

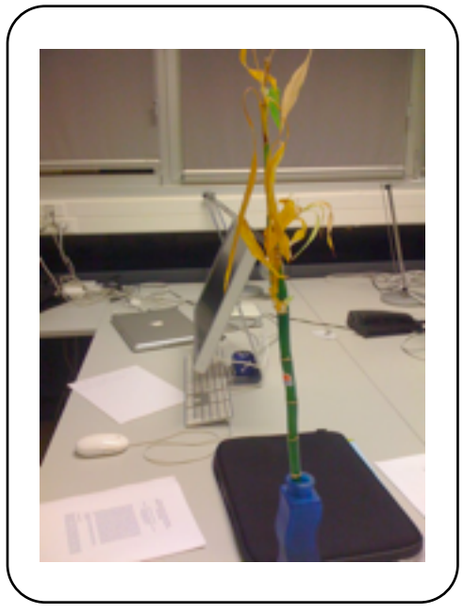


@Office

@Printer room

@Home

True Context-aware Applications

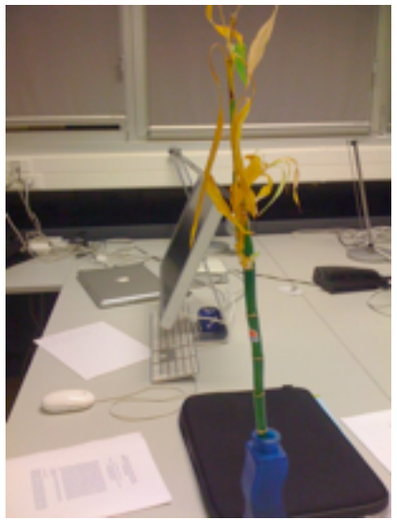


@Office

@Printer room

@Home

True Context-aware Applications

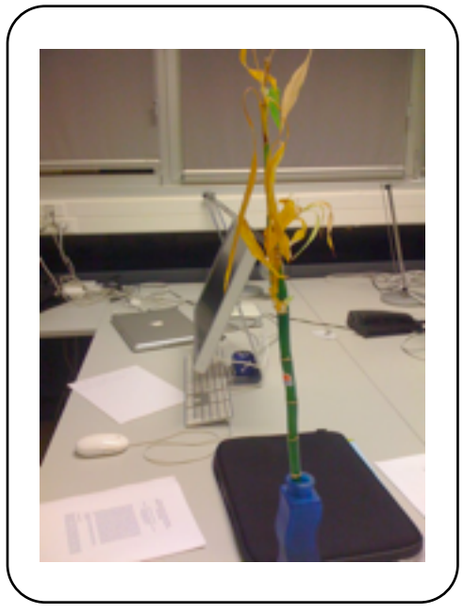


@Office

@Printer room

@Home

True Context-aware Applications



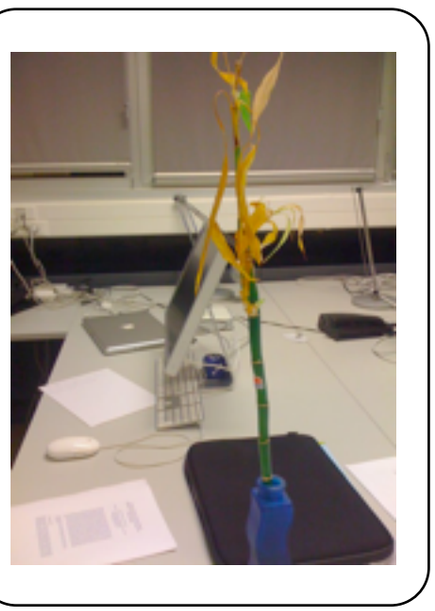
@Office

@Printer room

@Home

Behavioural variants

True Context-aware Applications



@Office

@Printer room

@Home

Characteristics:

1. **Prompt** adaptability.
2. **Context-constrained** executions.
3. **Sudden** interruptions.

Programming **True** Context-aware Applications

```
1 (define making-coffee
2   (lambda (person-name)
3     (show "Welcome: " person-name)
4     (task "1.place a cup")
5     (task "2.select ingredients")
6     (task "3.press make button")
7     (task "4.pick your coffee")))
```

Context-independent

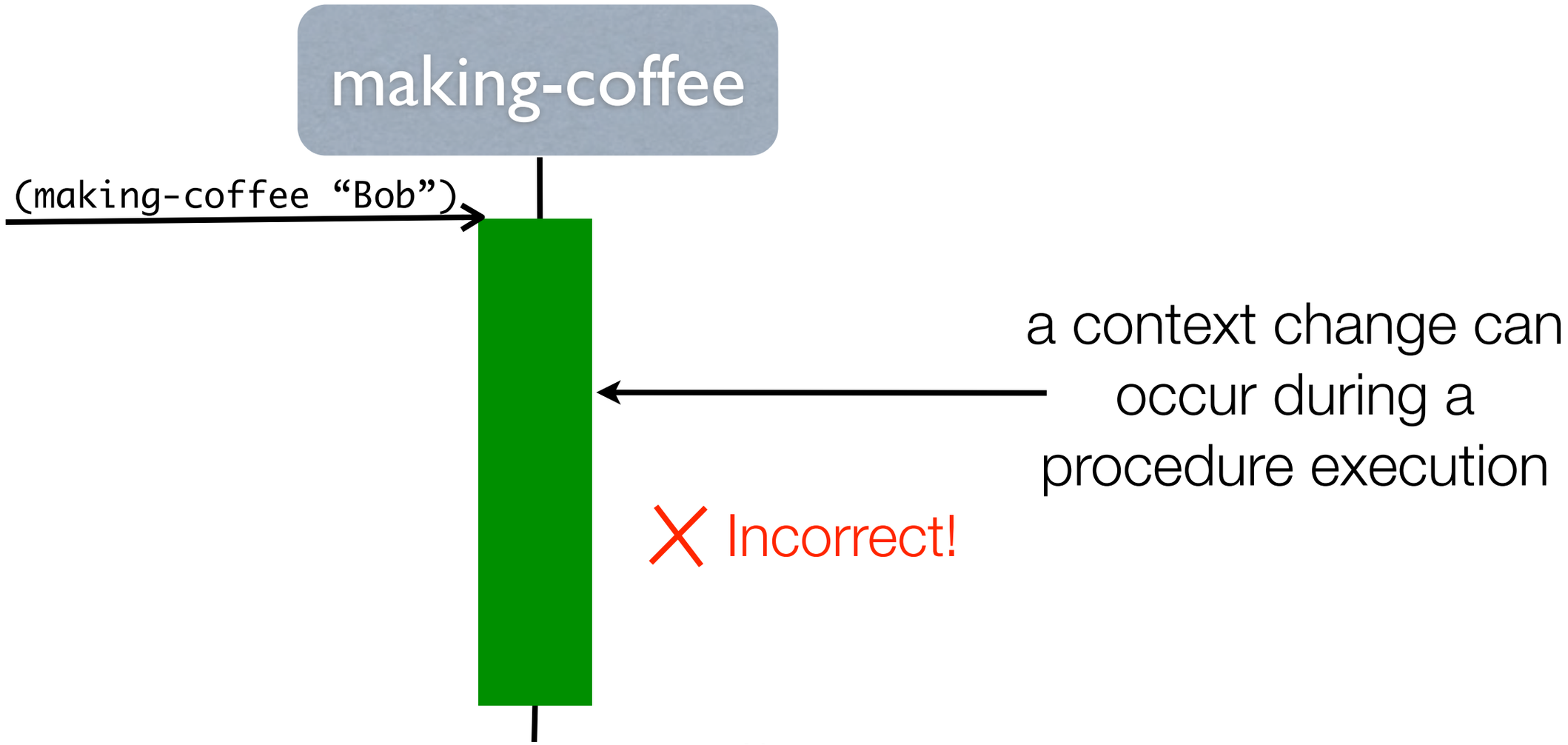
Programming **True** Context-aware Applications

```
(if (nearby-coffee-machine?)
    (making-coffee “Bob”))
```

Programming **True** Context-aware Applications

```
(if (nearby-coffee-machine?)
    (making-coffee “Bob”))
```

Not enough!



Programming **True** Context-aware Applications

```
1 (define making-coffee
2   (lambda (person-name)
3     (show "Welcome: " person-name)
4     (task "1.place a cup")
5     (task "2.select ingredients")
6     (task "3.press make button")
7     (task "4.pick your coffee")))
```

- How to constrain an entire procedure execution to the right context?
- What to do when a context change occurs in the middle of an ongoing execution?

Manual Checks, Coroutines, Continuations, ...

```
1 (define making-coffee
2   (lambda (person-name)
3     (if (nearby-coffee-machine?)
4         (show "Welcome: " person-name)
5         (save/suspend))
6     (if (nearby-coffee-machine?)
7         (task "1.place a cup")
8         (save/suspend))
9     (if (nearby-coffee-machine?)
10        (task "2.select ingredients")
11        (save/suspend))
12    (if (nearby-coffee-machine?)
13        (task "3.press make button")
14        (save/suspend))
15    (if (nearby-coffee-machine?)
16        (task "4.pick your coffee")
17        (save/suspend))))
```

Repetitive context checks

Error-prone!

Resumptions?

What the Developer Really Wants ...

(nearby-coffee-machine?)

```
1 (define making-beverage
2   (lambda (person-name)
3     (show "Welcome: " person-name)
4     (task "1.place a cup")
5     (task "2.select ingredients")
6     (task "3.press make button")
7     (task "4.pick your coffee")))
```

(nearby-soup-maker?)

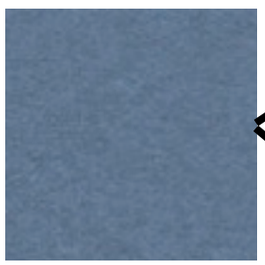
```
1 (define making-beverage
2   (lambda (person-name)
3     (show "Welcome: " person-name)
4     (task "1. select soup can")
5     (task "2. get a pan")
6     (task "3. pour soup in the pan")
7     (task "4. turn on hotplate")
```

Requirements:

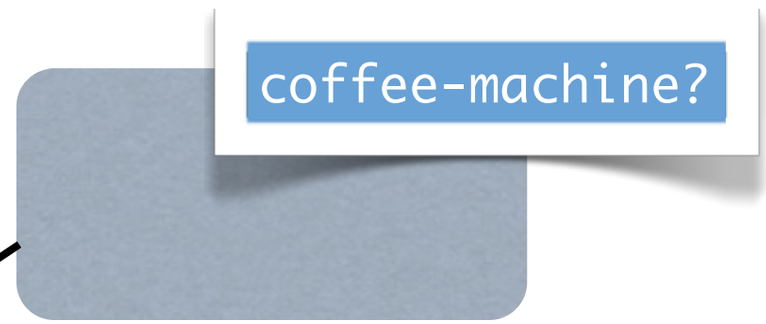
1. Contextual dispatch
2. Reactive dispatch
3. Context-dependent interruptions
4. Context-dependent resumptions
5. Reactive scope management

Interruptible Context-dependent Executions (ICoDE)

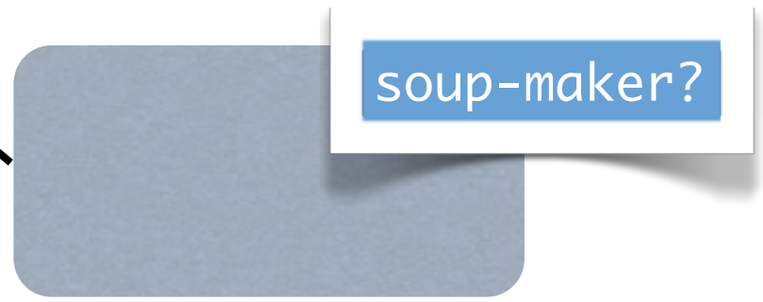
Context sources
as reactive values.



making-beverage
modal



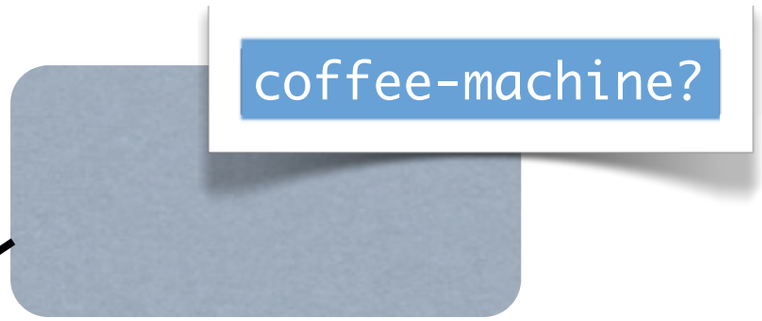
making-coffee **mode**



making-soup **mode**

Contextual and Reactive Dispatching

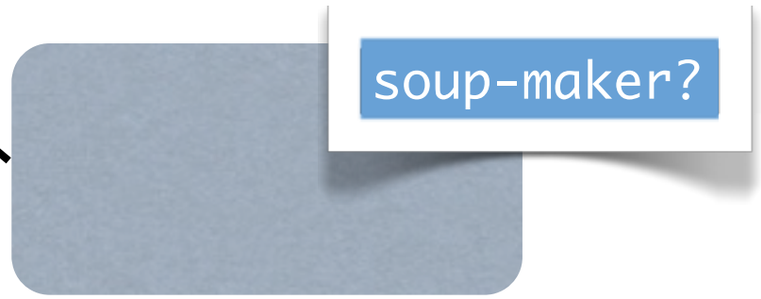
Context sources
as reactive values.



making-coffee **mode**

(making-beverage “Bob”)

making-beverage
modal



making-soup **mode**

Contextual and Reactive Dispatching

Context sources
as reactive values.

(making-beverage “Bob”)

making-beverage
modal

coffee-machine?

making-coffee **mode**

soup-maker?

making-soup **mode**

Contextual and Reactive Dispatching

Context sources
as reactive values.

making-beverage
modal

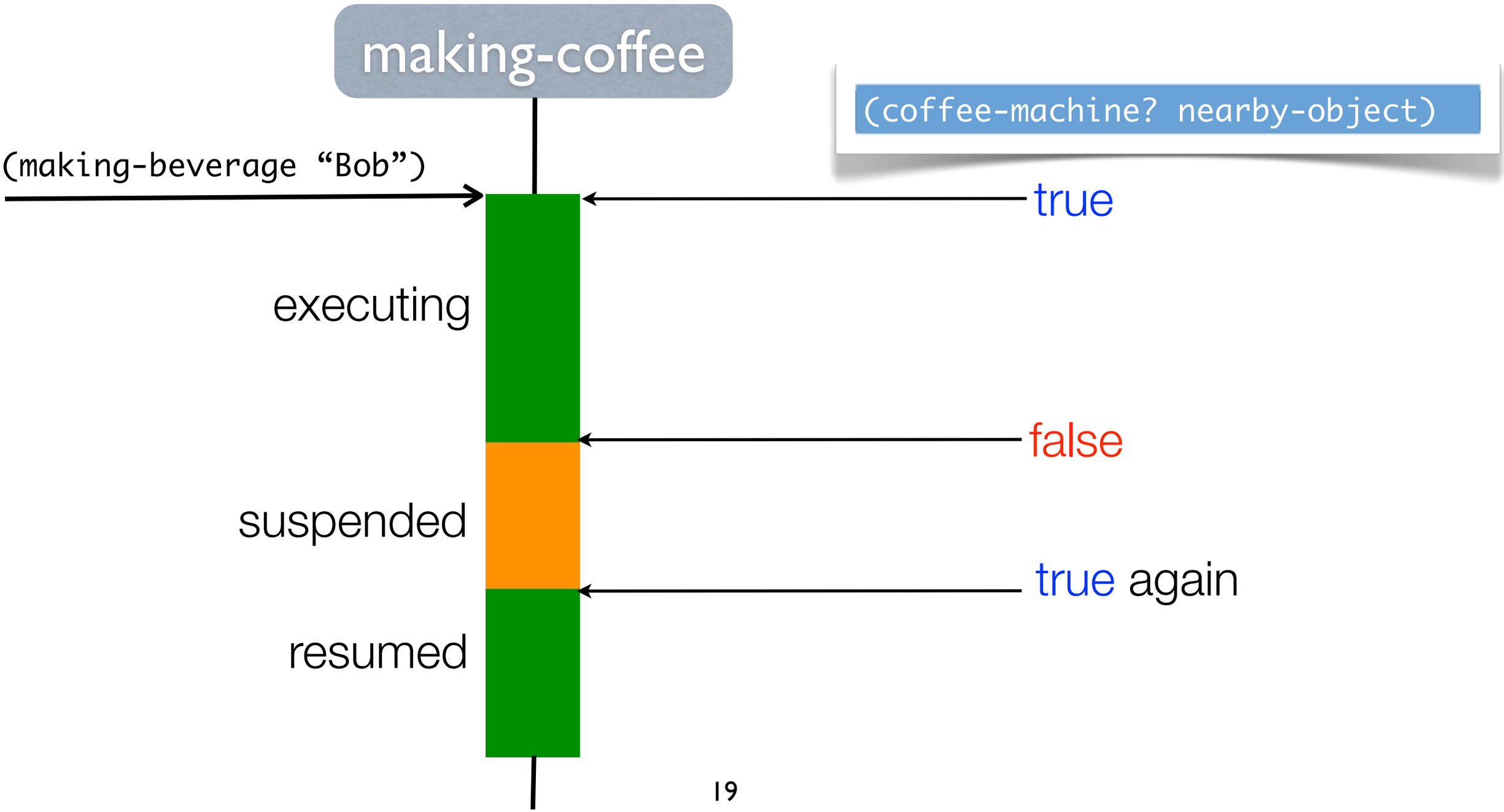
coffee-machine?

making-coffee **mode**

soup-maker?

making-soup **mode**

Interruptible and Resumable Executions



Interruptible Context-dependent Executions in Flute

Context source:

```
(define nearby-object (ctx-source))
```

Modal:

```
(define making-beverage (modal (nearby-object)
  (define user-experience 1)))
```

context source

shared variable

Mode:

```
(define making-coffee
  (mode (making-beverage)
    (coffee-machine? nearby-object)
    (suspend resume deferred)
    (lambda (person-name)
      (show "Welcome: " person-name)
      (task "1. place a cup")
      (task "2. select ingredients")
      ...)))
```

modal

context predicate

interruption, resumption, and scoping strategies

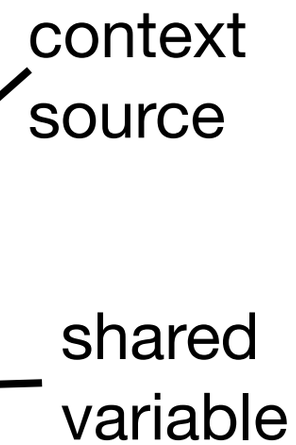
Interruptible Context-dependent Executions in Flute

Context source:

```
(define nearby-object (ctx-source))
```

Modal:

```
(define making-beverage (modal (nearby-object)
  (define user-experience 1)))
```

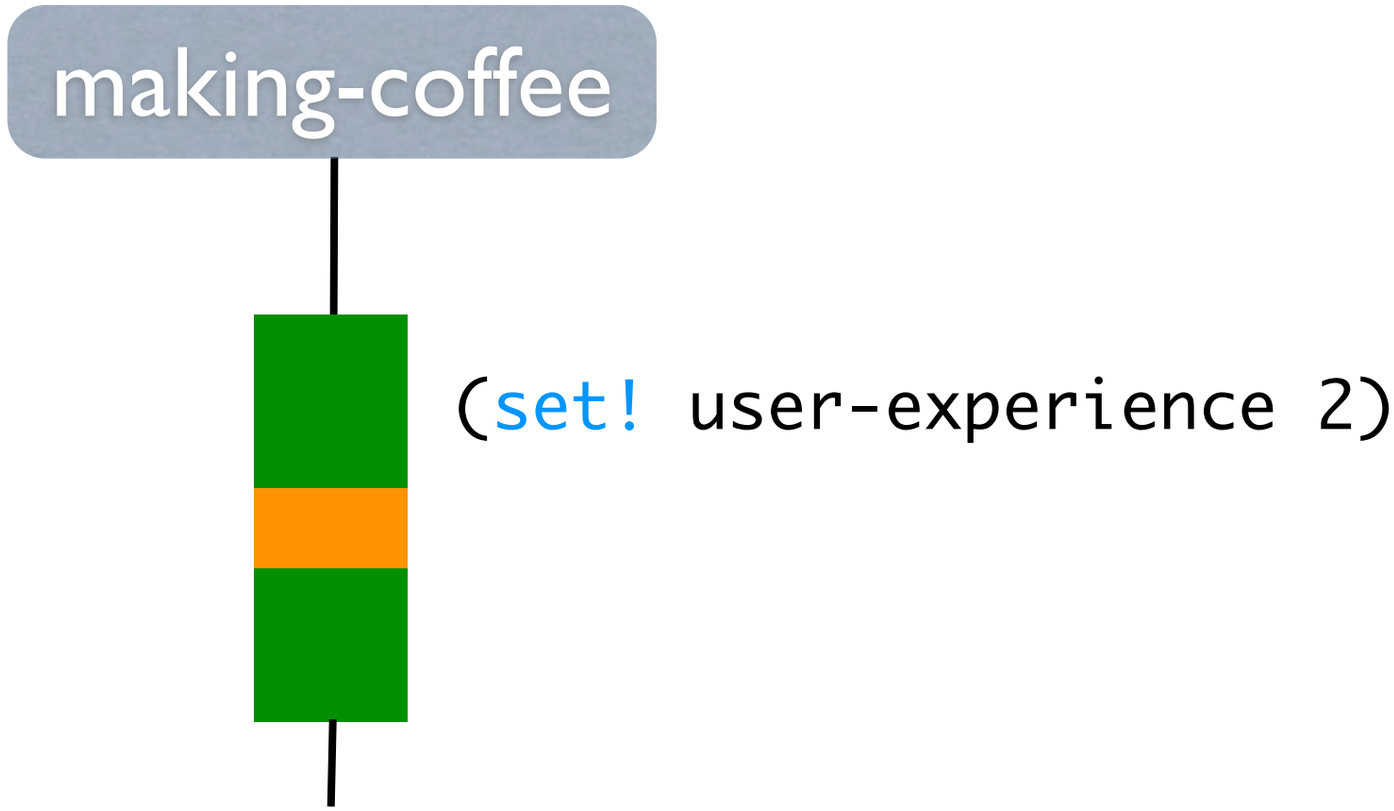


Mode:

```
(define making-coffee
  (mode (making-beverage)
    (coffee-machine? nearby-object)
    (suspend resume deferred)
    (lambda (person-name)
      (show "Welcome: " person-name)
      (task "1. place a cup")
      (task "2. select ingredients")
      ...)))
```

```
(define making-soup
  (mode (making-beverage)
    (soup-maker? nearby-object)
    (suspend resume deferred)
    (lambda (person-name)
      (show "Welcome: " person-name)
      (task "1. select soup can")
      (task "2. get a pan")
      ...)))
```

What to do with State Changes?



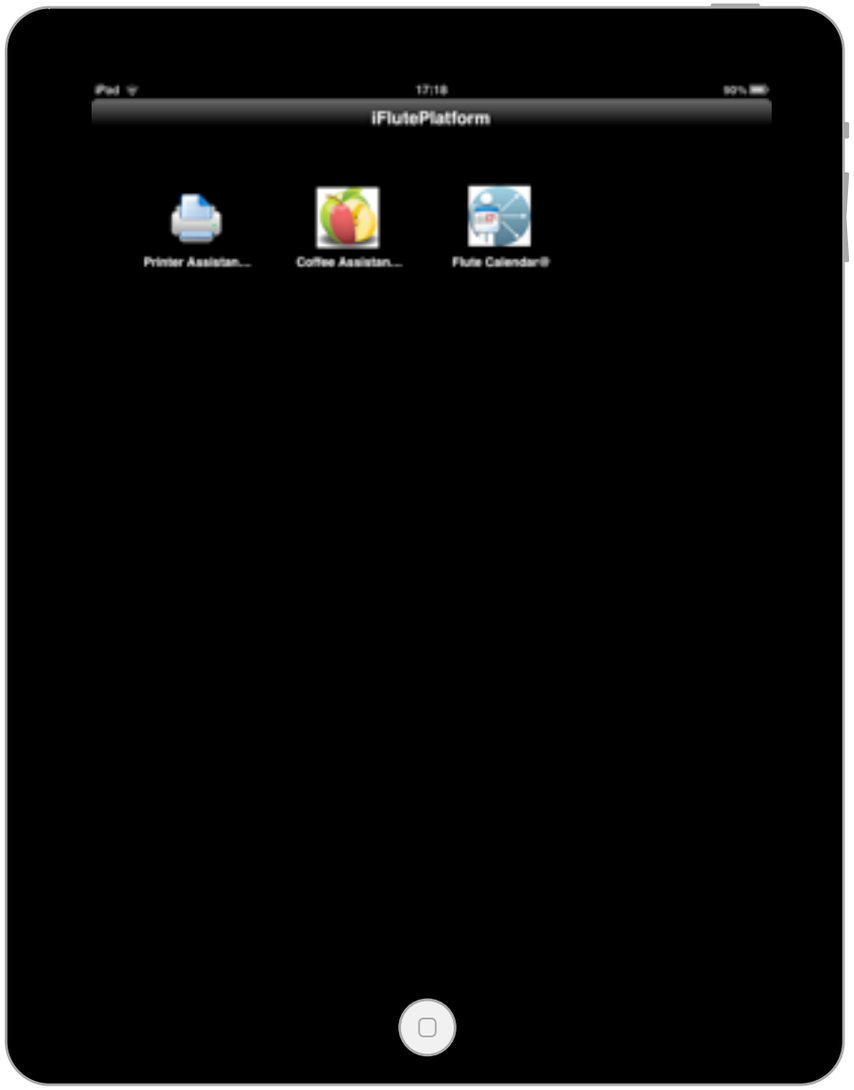
`immediate` Changes are immediately visible to other executions.

`deferred` Changes become visible to other executions on completion.

`isolated` Changes remain locally visible to the execution.

The Flute Mobile Platform

- Flute is implemented as a meta-interpreter on top of iScheme [1].
- Context sources: GPS, proximity sensor, accelerometer on the iOS.

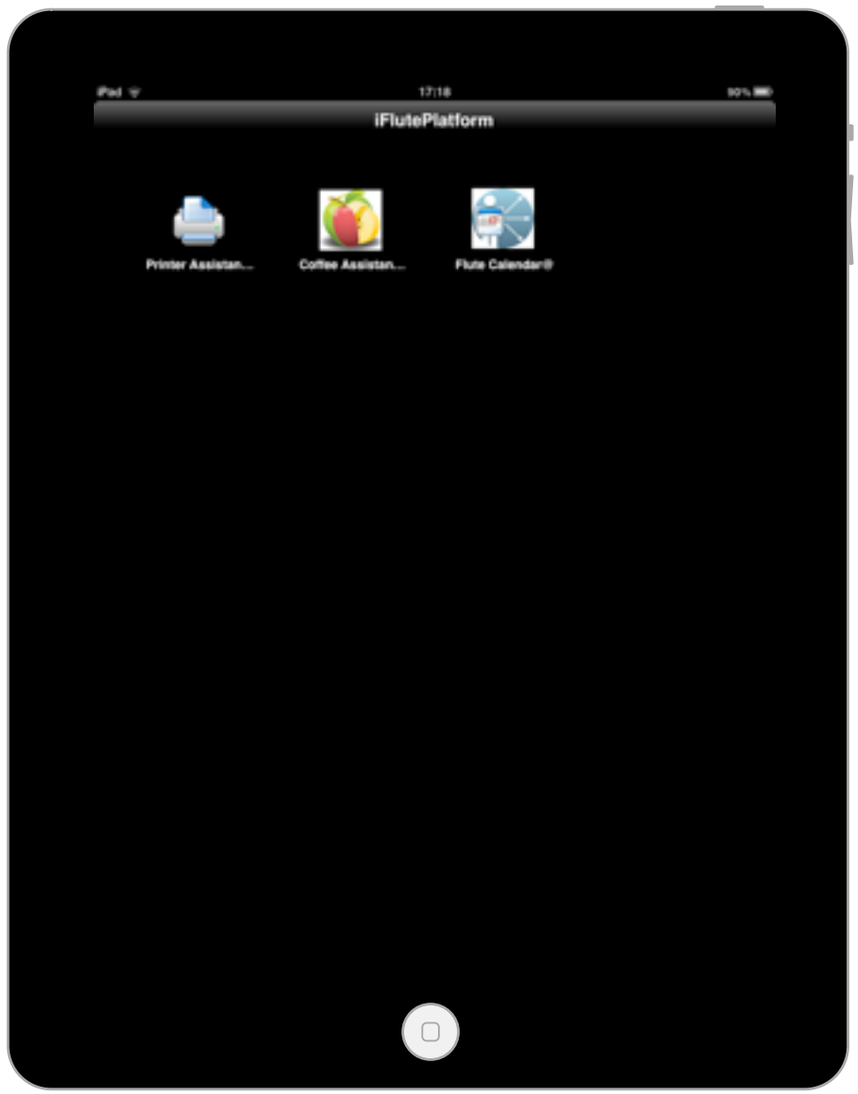


Example apps on the Flute mobile platform

1. *Kalenda:* a *true* context-aware calendar assistant.
2. *Pulinta:* a *true* context-aware printer assistant.
3. *Tasiki*: a *true* context-aware task assistant.

[1] Bainomugisha, E. *et.al*. (2012), Bringing Scheme Programming to the iPhone - Experience. *Software: Practice and Experience*, 42(3):331–356.

The Future of Mobile Platforms Lies in **True** Context-awareness



✓ Languages

✗ Middleware

In Summary

Interruptible Context-dependent Executions (ICoDE):

- Interruptible and resumable executions.
- Contextual and reactive dispatch.
- Reactive scope management.
- Flute: an ICoDE instantiation.

Challenges

- Building a fully interruptible system.
- Garbage collection of suspended executions.

Thank You.

ebainomu@vub.ac.be

http://soft.vub.ac.be/~ebainomu/Flute/