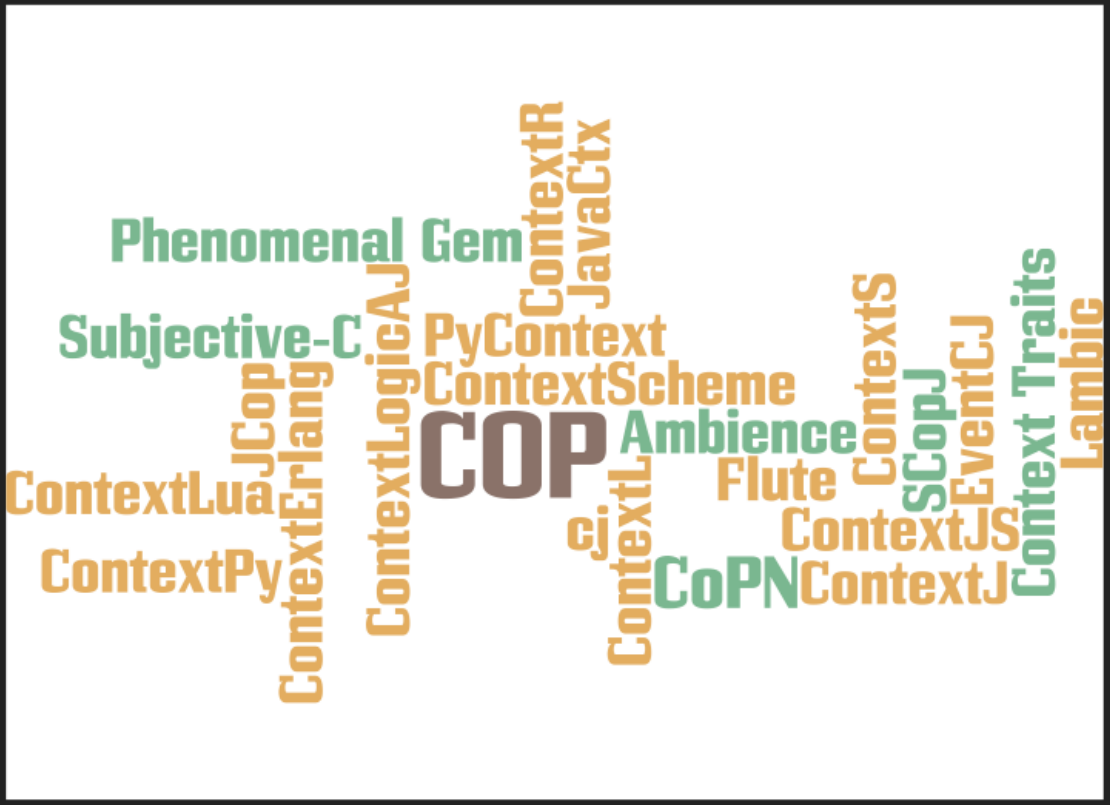
LINGI2252 – PROF. KIM MENS

SOFTWARE MAINTENANCE & EVOLUTION



LINGI2252 – PROF. KIM MENS

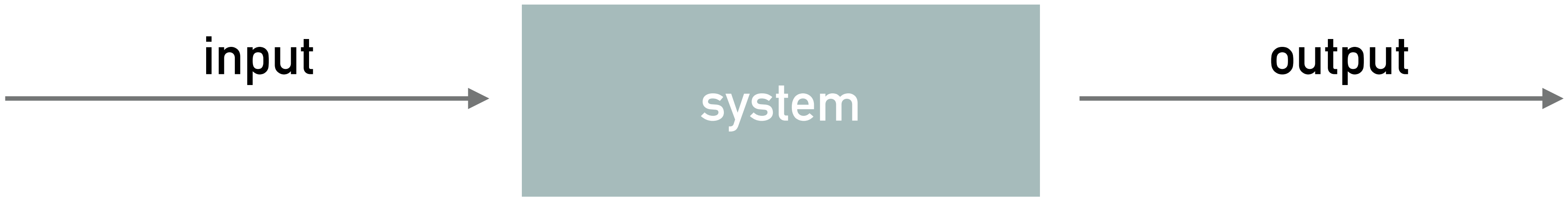
CONTEXT-ORIENTED PROGRAMMING*

* Slides based on joint research with Dr. Sebastian Gonzalez, Dr. Nicolas Cardozo & others

Why

is there a need for dynamic adaptation to context ?

Traditional Computer Systems



"Traditionally, hardware and software were input-output systems that took input explicitly given to them by a human, and acted upon that input alone to produce an explicit output.
Now, this view is seen as too restrictive. …"

Henry Lieberman & Ted Selker

Out of Context: Computer Systems That Adapt To, and Learn From, Context.
IBM Systems Journal, Vol 39, Nos 3&4, p.617-631, 2000 [Lieberman&Selker2000]

Context-Aware Systems

"… Smart computers, intelligent agent software, and digital devices of the future operate on data that is not explicitly given to them, data that they observe or gather for themselves. These operations may be dependent on time, place, weather, user preferences, or the history of interaction.
In other words: ***context****."*

Henry Lieberman & Ted Selker

Out of Context: Computer Systems That Adapt To, and Learn From, Context.
IBM Systems Journal, Vol 39, Nos 3&4, p.617-631, 2000 [Lieberman&Selker2000]

Evolution Of Hardware

fixed (1980)
mainframes
servers
desktops
consoles

portable (1990)
laptops
netbooks
subnotebooks

mobile (2000)
handhelds
tablets
smartphones

- CPU load
- available memory (RAM)
- available storage (HD)
- date and time
- connected peripherals
- network peers

+

- wi-fi signal quality
- battery power
- camera
- microphone
- light sensor

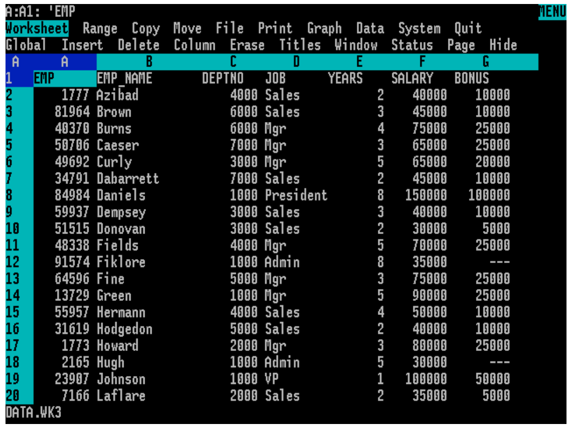
+

- touch screen
- geographical location
- GPS signal quality
- accelerometer

. . .

Evolution Of Software

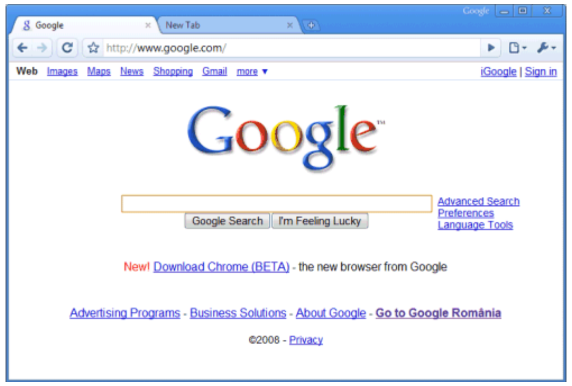
text (1970)
BSD
SunOS
MS DOS
GNU/Linux



```
A:A1: 'EMP                                                          MENU
Worksheet  Range  Copy  Move  File  Print  Graph  Data  System  Quit
Global  Insert  Delete  Column  Erase  Titles  Window  Status  Page  Hide
 A   A          B               C       D         E       F        G
1  EMP    EMP_NAME         DEPTNO    JOB       YEARS    SALARY   BONUS
2     1777 Azibad             4000 Sales            2     40000    10000
3    81964 Brown              6000 Sales            3     45000    10000
4    40370 Burns              6000 Mgr              4     75000    25000
5    50706 Caeser             7000 Mgr              3     65000    25000
6    49692 Curly              3000 Mgr              5     65000    20000
7    34791 Dabarrett          7000 Sales            2     45000    10000
8    84984 Daniels            1000 President        8    150000   100000
9    59937 Dempsey            3000 Sales            3     40000    10000
10   51515 Donovan            3000 Sales            2     30000     5000
11   48338 Fields             4000 Mgr              5     70000    25000
12   91574 Fiklore            1000 Admin            8     35000      ---
13   64596 Fine               5000 Mgr              3     75000    25000
14   13729 Green              1000 Mgr              5     90000    25000
15   55957 Hermann            4000 Sales            4     50000    10000
16   31619 Hodgedon           5000 Sales            2     40000    10000
17    1773 Howard             2000 Mgr              3     80000    25000
18    2165 Hugh               1000 Admin            5     30000      ---
19   23907 Johnson            1000 VP               1    100000    50000
20    7166 Laflare            2000 Sales            2     35000     5000
DATA.WK3
```

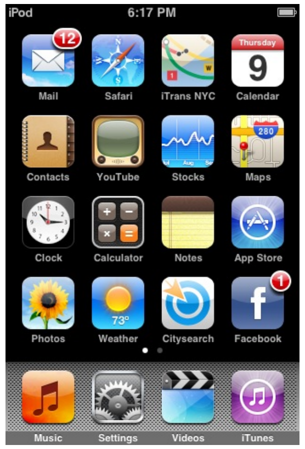
graphical (1980)
Mac OS
Amiga OS
Windows
KDE, GNOME

web (1990)
static
dynamic
web 2.0
mashups



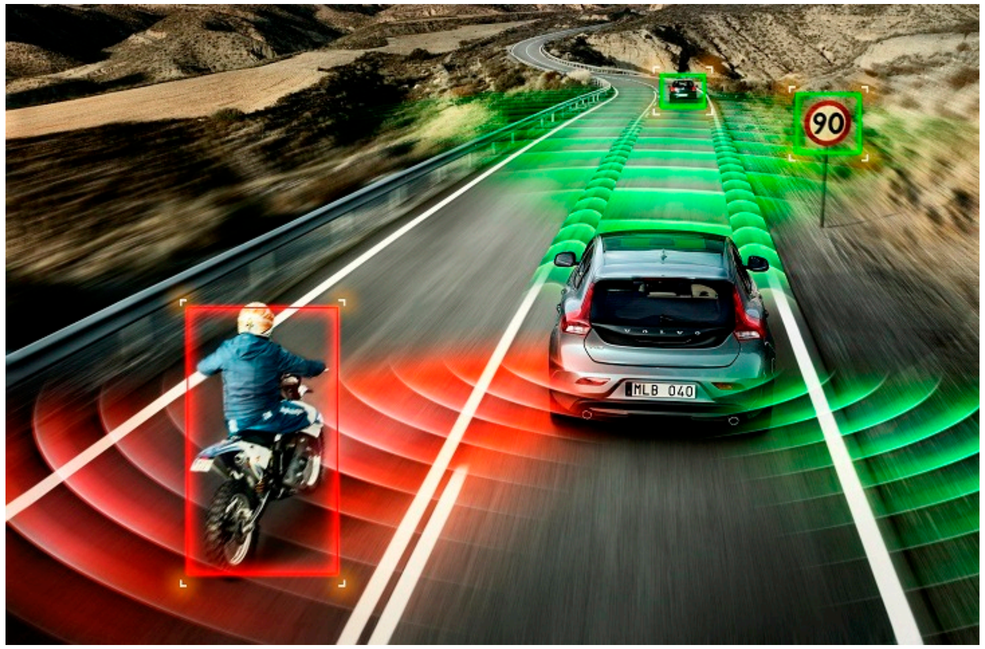
New! Download Chrome (BETA) - the new browser from Google

mobile (2000)
Symbian OS
Windows CE
iOS
Android



- available libraries
- available hardware services
- available network services
- user task
- user expertise
- user preferences
- user privileges
- task urgency
- operation modes
  - logging
  - debugging
  - degraded
  - free trial
  - partial failure
  - domain specific
    - [3D] wireframe / solid view
    - [Maps] satellite / schematic
    - ...

The Future Is Here



“smart” objects

Need For Context-Aware Computing

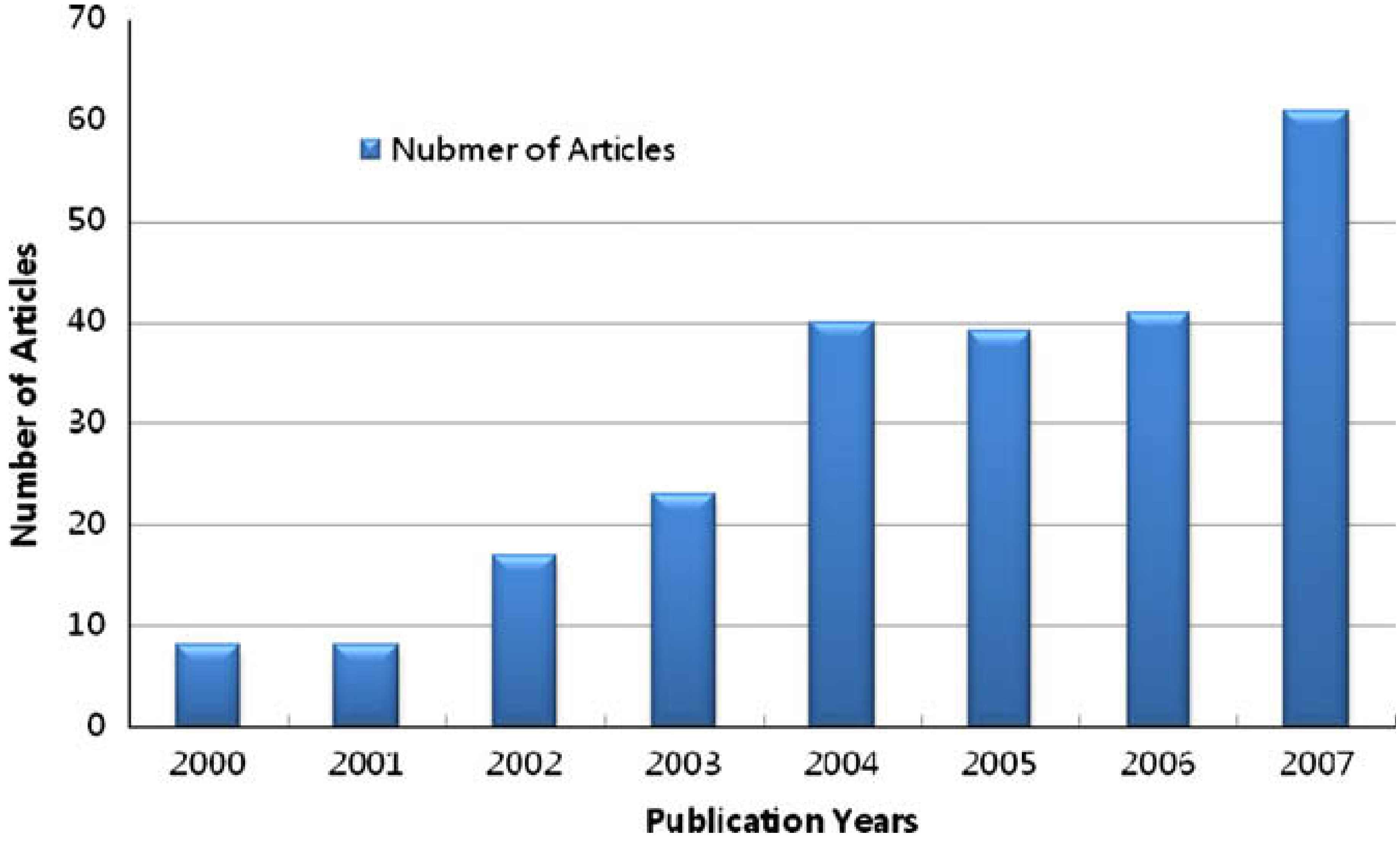
"Computer systems will increasingly need to be
sensitive to their context
to
***serve their users better**."*

Eli Rohn

Predicting Context Aware Computing Performance.
Ubiquity, p.1-17, Feb. 2003 [Rohn2003]

Context-Aware Systems

Idea appeared ~ late 1980s; increasingly studied since ~ 2000.

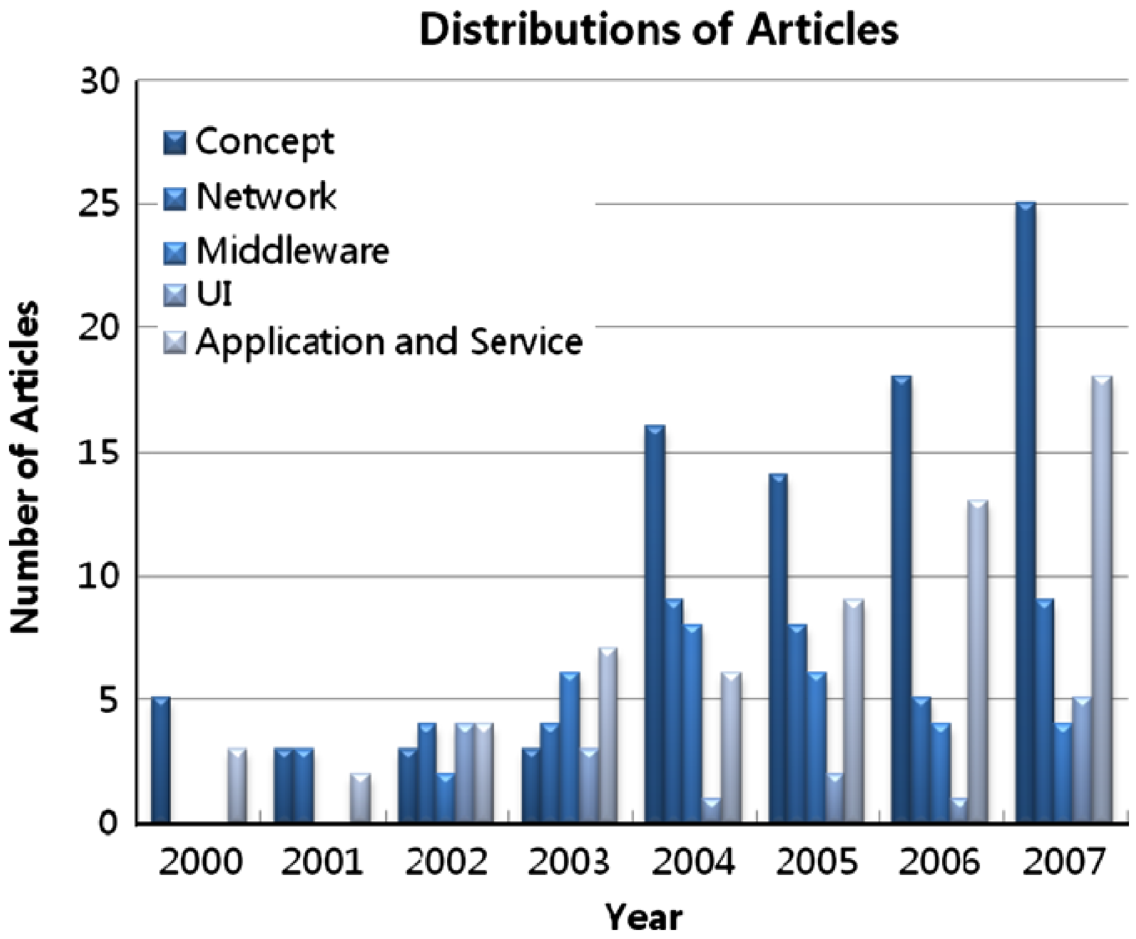


Jong-yi Hong, Eui-ho Suh, Sung-Jin Kim
Context-Aware Systems: A literature review and classification.
Expert Systems with Applications 36, 2009 [Hong&al2009]

Context-Aware Systems

Studied from a variety of research angles [Hong&al2009]:

- **conceptual**: guidelines, frameworks, algorithms, context reasoning and context data management
- **networks**: network protocols, sensor networks, …
- **middleware** for distributed context-aware applications
- **applications**: studies and development of dedicated context-aware applications (e.g., a smart tour guide)
- **user-interface** technology and usability studies



Context-Oriented Programming

Focusses on the programming angle:

Enabling context-aware software adaptability through a programming language engineering approach:

- dedicated programming languages to express context-driven behaviour adaptation
- contexts and behavioural variations to context as first class language citizens

Phenomenal Gem
ContextR
JavaCtx
Subjective-C
ContextLogicAJ
PyContext
ContextScheme
JCop
ContextErlang
COP
Ambience
ContextS
SCopJ
EventCJ
Context Traits
Lambic
ContextLua
Flute
cj
ContextL
ContextJS
ContextPy
CoPN
ContextJ

COP

enables context-driven
behaviour adaptability ...

... through a programming
language engineering approach

Some Definitions

*“A software system is **context-aware** if it can extract, interpret and use context information and adapt its functionality to the current context of use.”*

[Rohn2003]

*“**Context** is everything but the explicit input and output to a system.”*

[Lieberman&Selker2000]

*“A **context-oriented** software system is a context-aware system that has an explicit representation of context and contextual variations as first class citizens.”*

[*my definition*]

Enabling Context-Driven Behaviour Adaptability

How to build software systems that can adapt their behaviour dynamically …

… according to detected context changes in their surrounding environment ?

One possible approach :

context-oriented programming

a programming language engineering approach

Context Is Key

Applications should become more aware of their execution context, and should adapt dynamically to such context to provide services that match their clients' needs to the best extent possible.

spatial state

position, orientation, movement

location semantics

nearby objects & facilities

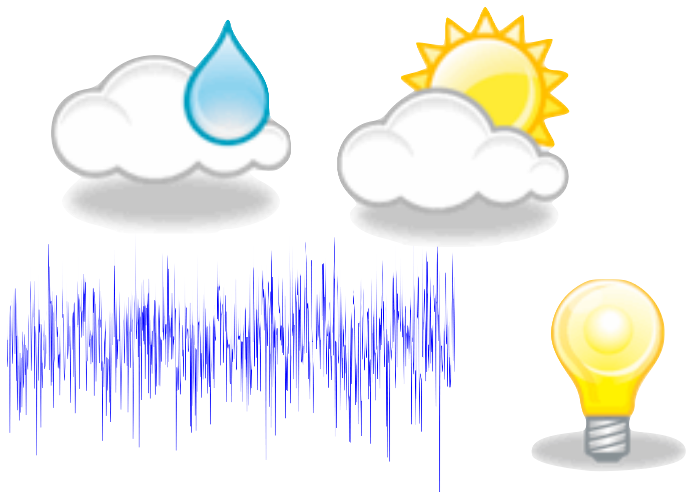
users

expertise, preferences

network peers & services

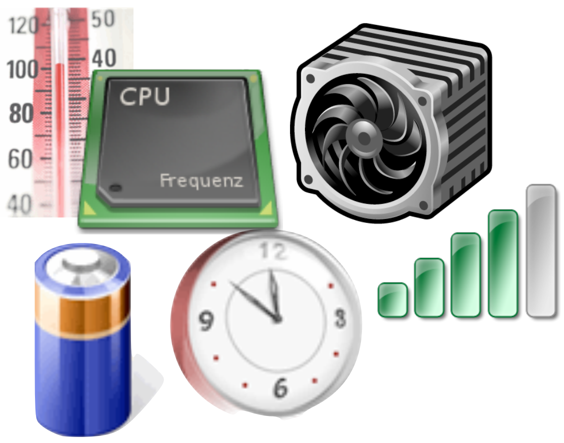
projector, GPS, storage

environmental properties



humidity, light, noise, lighting

internal state



load, time, battery

Adaptation Examples

peer service

take advantage of room projector for presentation

location semantics

disable phone ringtone in quiet places

internal state

decrease playback quality when battery power is low

user task

show parking spots and gas stations (only) when driving

environmental conditions

give more detailed indications when visibility is low

So Why Aren’t We There Yet?

Richard Gabriel, 2006

Software systems today are produced according to a manufacturing model: a finished product is constructed at the factory and shipped to its final destination where it is expected to act like any other machine —reliable but oblivious to its surroundings and its own welfare.

Mindset Mismatch

we still program this...

(2010)

using the
programming models
conceived for this....

(1980)

Current Mindset

programming in isolation

Current programming techniques and design principles invite programmers to think in a way that is mostly oblivious of the physical, technical and human environment in which the software will be used.
Many chances of delivering improved services are thus missed.

Needed Mindset

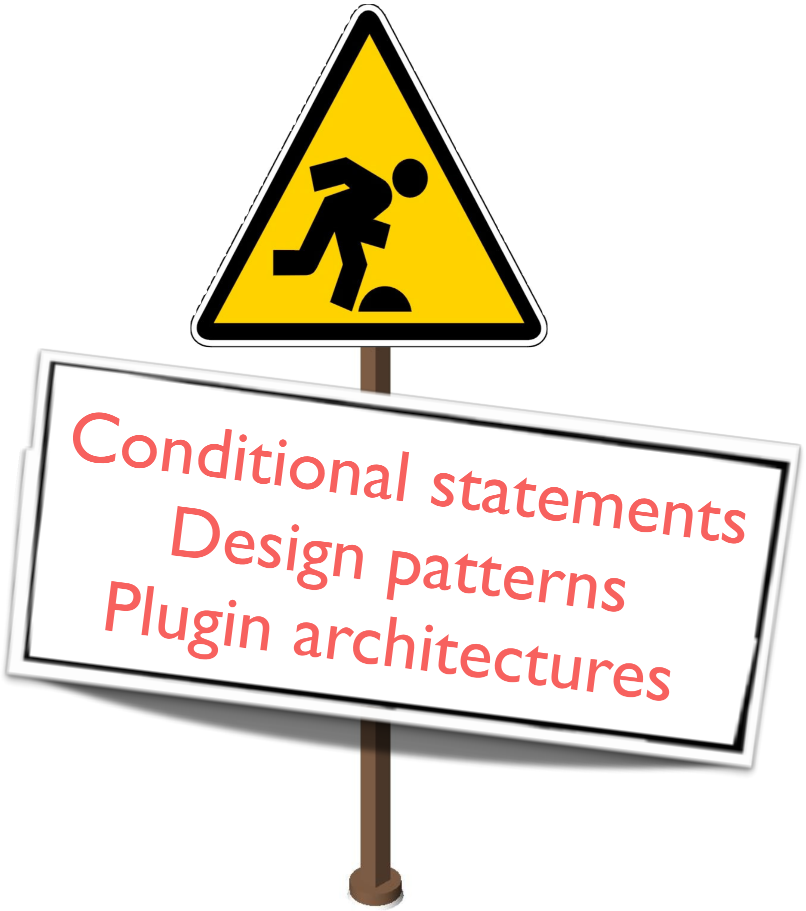
programming with context

A new paradigm is needed that helps overcoming this limiting vision by putting programmers in the **right state of mind** to build dynamically adaptable applications from the ground up.

Mindset Shift

from context-blind systems

to context-oriented systems



Conditional statements
Design patterns
Plugin architectures

Adaptation Example

call reception behaviour

context	behaviour
default	ringtone

Adaptation Example

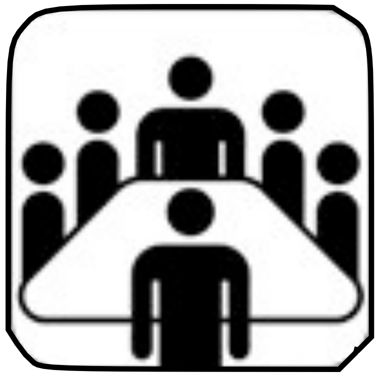
call reception behaviour

context

quiet

behaviour

vibration



Adaptation Example

call reception behaviour

context

off-hook

behaviour

call waiting signal

Adaptation Example

call reception behaviour

context

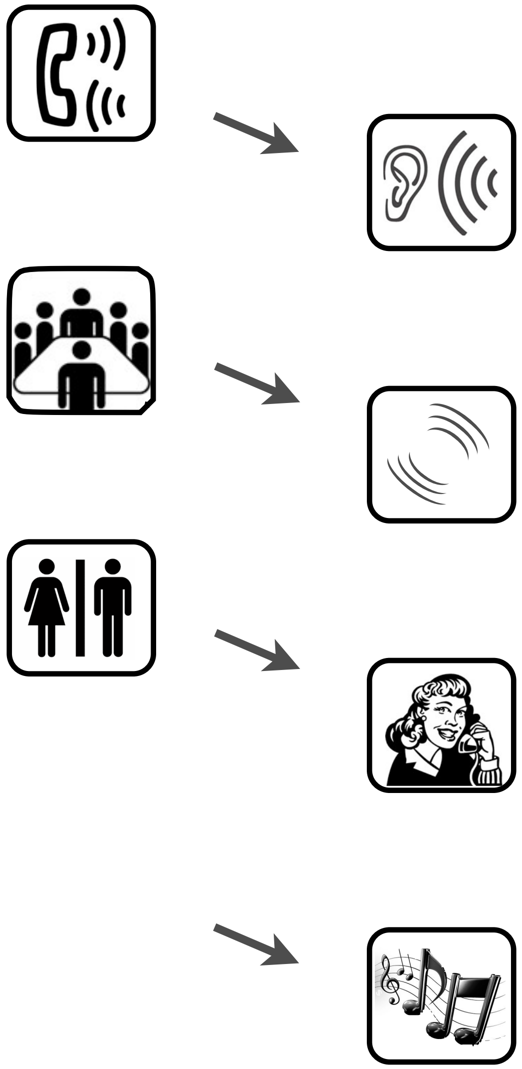
behaviour

3G
9:41 AM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Notes
Utilities
iTunes
App Store
Settings
Phone
Mail
Safari
iPod

Paradigmatic Shortcomings

conditional statements

```
class phone {

  method receive ( call ) {

    if ( phone.isOffHook( ) )
      play( phone.callWaitingSignal( ), 2 );

    else if ( phone.environment( ).acoustics( ).isQuiet( ) )
      phone.vibrate( 5 );

    else if ( phone.user( ).isUnavailable( ) )
      forwardCall( call, phone.forwardNumber( ) );

    else
      play( phone.ringTone( ), 10 );
}
```

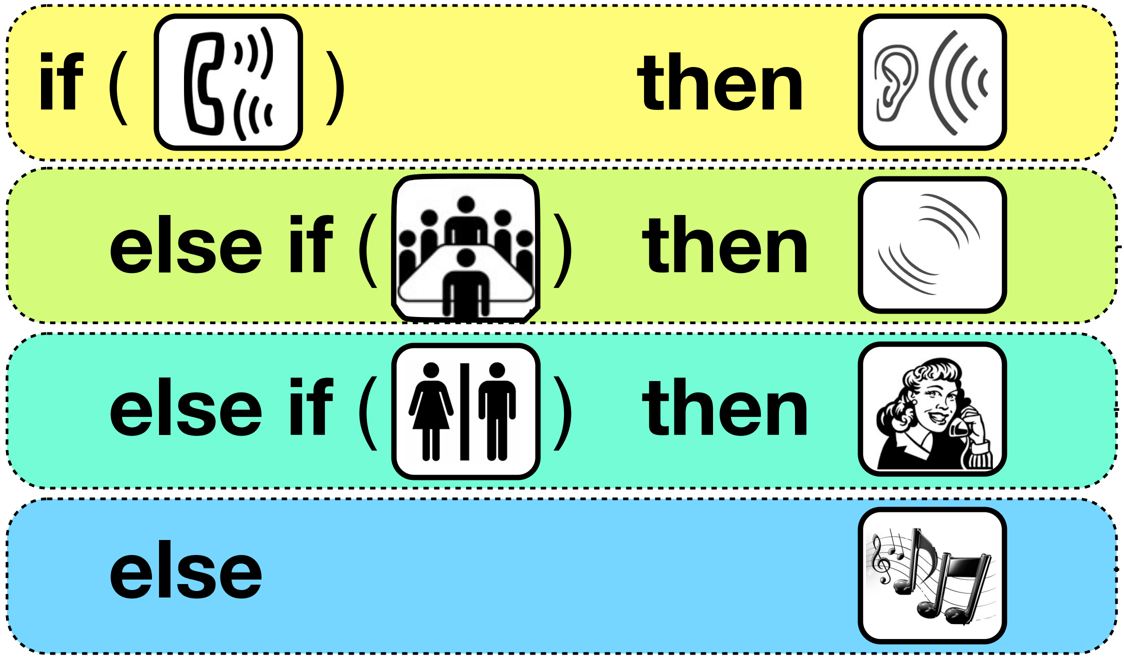


Paradigmatic Shortcomings

conditional statements

class phone {

method receive (call) {



```
if (  )            then
    else if (  )   then
    else if (  )   then
    else
```

phone status

phone environment

phone user

default

}

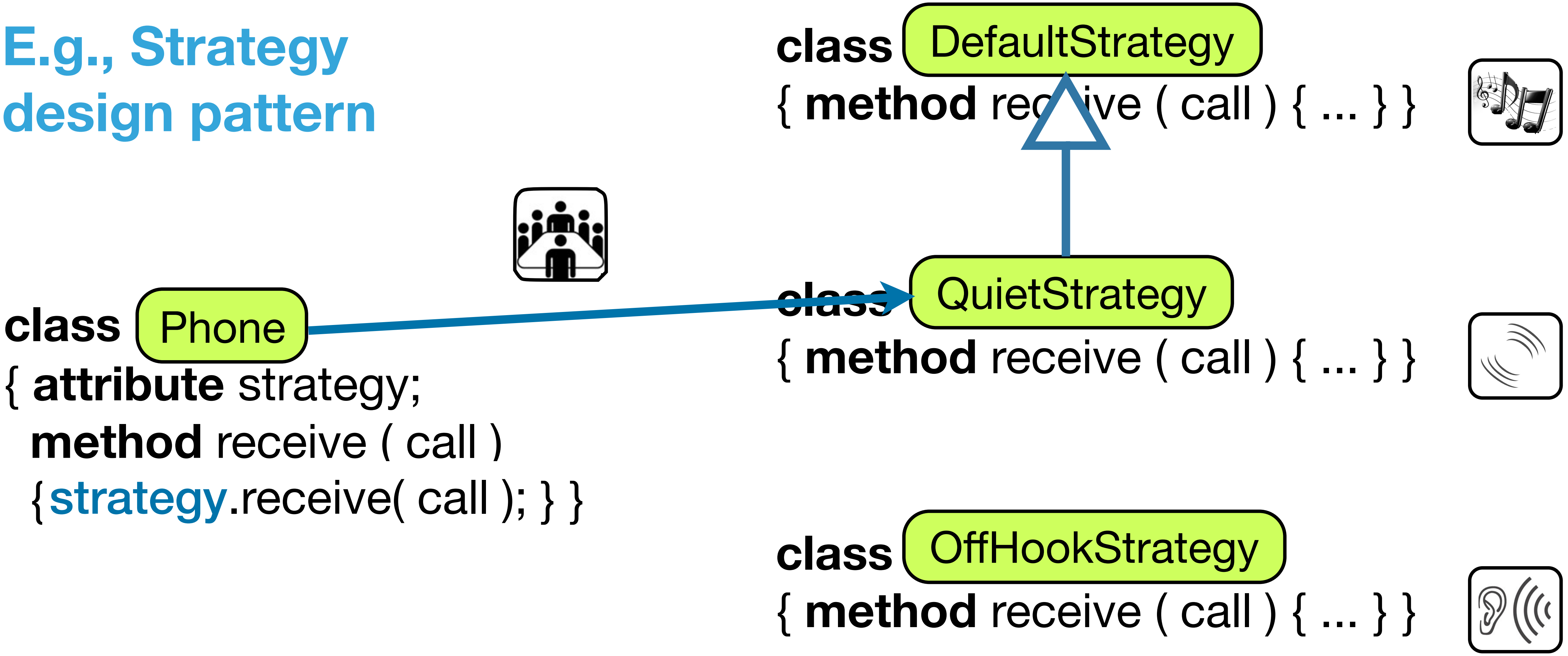
Adaptable

Tangled
Scattered
Fixed
No reuse
Complex logic

Paradigmatic Shortcomings

special software architectures

E.g., Strategy design pattern



```
class Phone
{ attribute strategy;
  method receive ( call )
  { strategy.receive( call ); } }

class DefaultStrategy
{ method receive ( call ) { ... } }

class QuietStrategy
{ method receive ( call ) { ... } }

class OffHookStrategy
{ method receive ( call ) { ... } }
```

Modular
Open

Infrastructural burden
Anticipated adaptation points

Paradigmatic Shortcomings

General Symptoms (Recap)

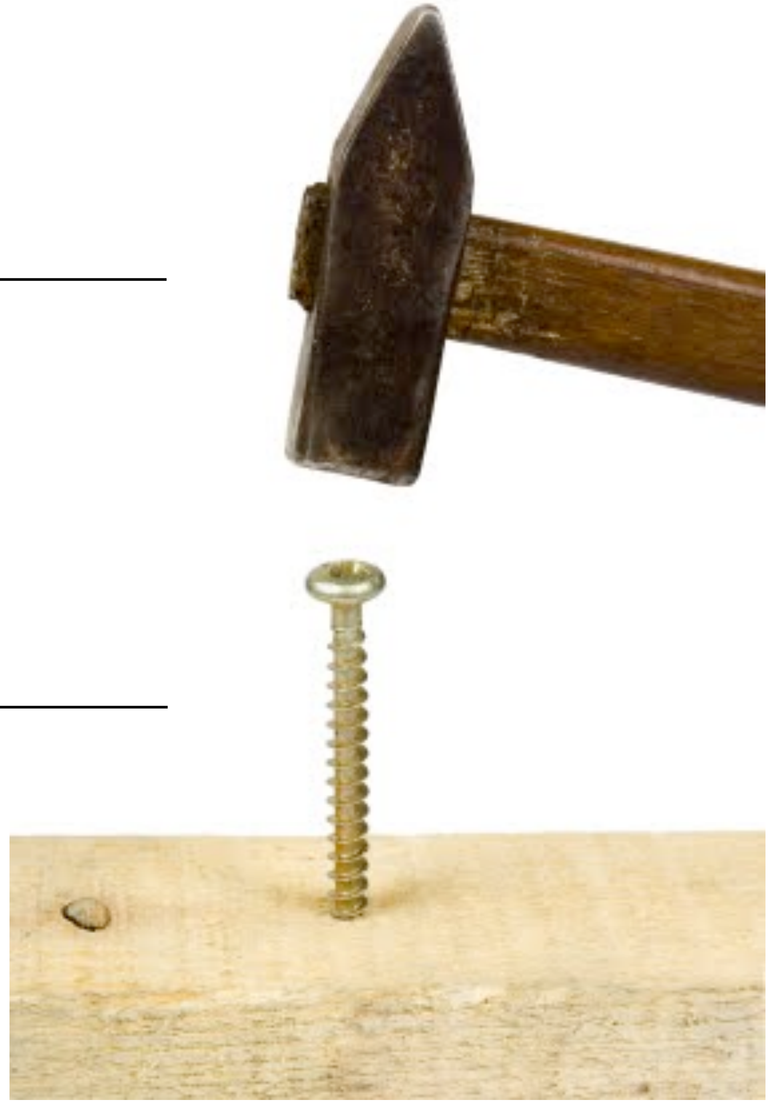
- ⚠ **Software rigidness**
  The variability points of the application are hard-coded in its architecture. It is difficult to add new variants non-invasively.

- ⚠ **Lack of modularity**
  Tight coupling between core business logic and infrastructural code to manage the variants makes the software difficult to maintain and evolve.

- ⚠ **Mindset mismatch**
  Programming tools make programmers oblivious of the context in which their applications will run. Programmers are not put in the right state of mind to build adaptable software.

Hypothesis

A major obstacle for adaptability is the unavailability of appropriate context-aware programming languages and related tool sets.



current programming tools

adaptive systems

we need to reengineer our tools

Side Comment

programming abstractions matter

domain: math

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n(n-1)! & \text{if } n > 0 \end{cases}$$

tool 1: C#

```
using System;

public class Program
{
   static long Factorial(long number)
   {
      if(number <= 1)
         return 1;
      else
         return number * Factorial(number - 1);
   }

   static int Main(string[] args) {
      Console.WriteLine(Factorial(5));
      return 0;
   }
}
```

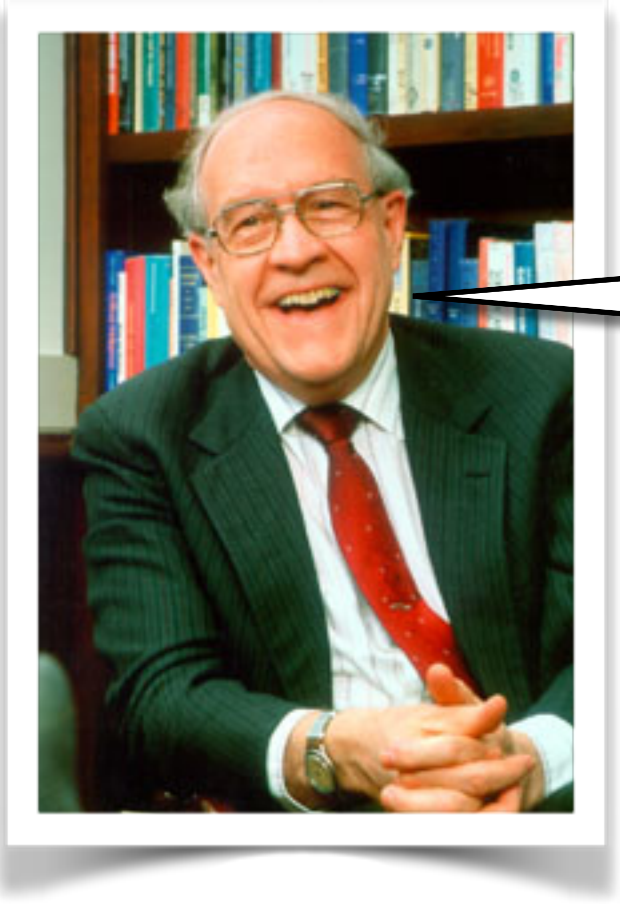
tool 2: Ruby

```
def fact(n)
  n <= 1 ? 1 : n * fact(n - 1)
end

fact(5)
```

maintainability

Side Comment

programming language engineering



Frederick Brooks, 1987

essential complexity ≠ accidental complexity

A high-level language frees a program from much of its accidental complexity; it eliminates a whole level of complexity that was never inherent in the program at all.

Develop programming tools that reduce accidental complexity in the expression of context-driven behaviour adaptation

What?

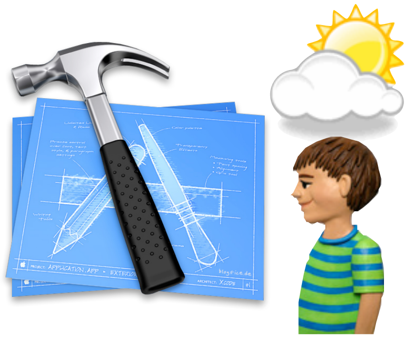
context-driven
software adaptability through

dedicated language abstractions

and composition mechanisms

Ambience

Subjective-C



Context Traits

2008 — 2010 — 2013

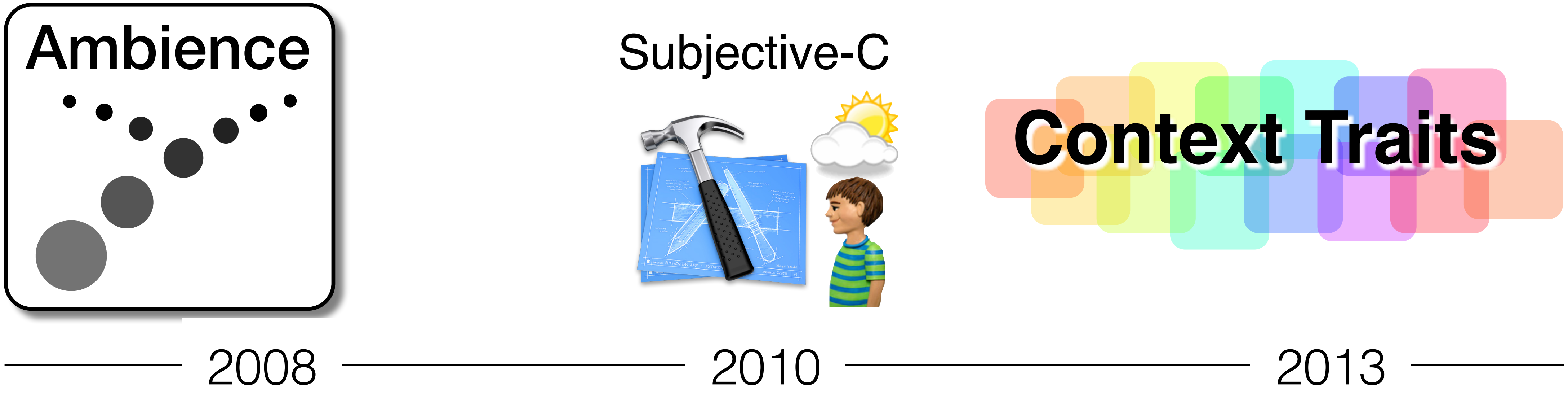
"Our ambition is to provide languages, formalisms, models and tools to support the development of software systems that can dynamically adapt their behaviour to the current execution context, to provide the most appropriate behaviour according to that context."

Some Context-Oriented Programming Languages

S. GONZALEZ, K. MENS, A. CADIZ.
Context-Oriented Programming with the Ambient Object System.
Journal of Universal Computer Science, 14(20):3307–3332, 2008.

S. GONZALEZ, N. CARDOZO, K. MENS, A. CADIZ, J-C. LIBBRECHT, J. GOFFAUX.
Subjective-C: Bringing Context to Mobile Platform Programming. International Conference on Software Language Engineering 2010.

S. GONZALEZ, K. MENS, M. COLACIOIU, W. CAZZOLA.
Context Traits: dynamic behaviour adaptation through run-time trait recomposition.
International conference on Aspect-Oriented Software Development 2013.



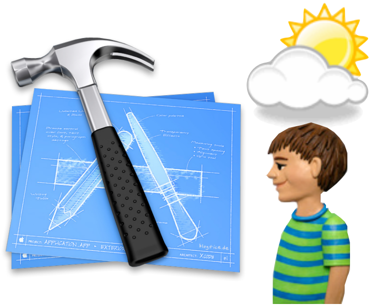
Some Context-Oriented Programming Languages

S. GONZALEZ, K. MENS, A. CADIZ.
Context-Oriented Programming with the Ambient Object System.
Journal of Universal Computer Science, 14(20):3307–3332, 2008.

S. GONZALEZ, N. CARDOZO, K. MENS, A. CADIZ, J-C. LIBBRECHT, J. GOFFAUX.
Subjective-C: Bringing Context to Mobile Platform Programming. International Conference on Software Language Engineering 2010. *

S. GONZALEZ, K. MENS, M. COLACIOIU, W. CAZZOLA.
Context Traits: dynamic behaviour adaptation through run-time trait recomposition.
International conference on Aspect-Oriented Software Development 2013.

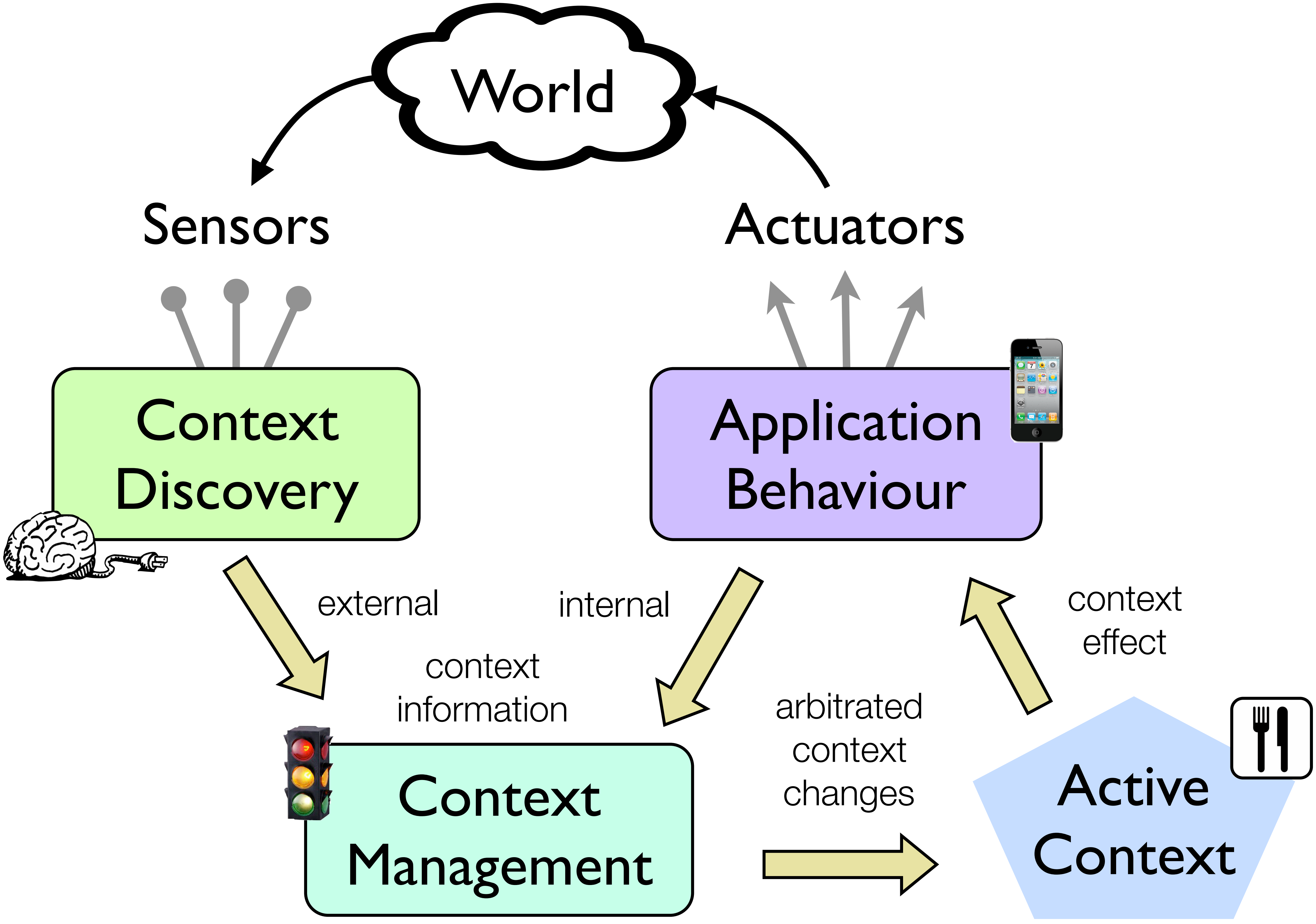
Subjective-C



2010

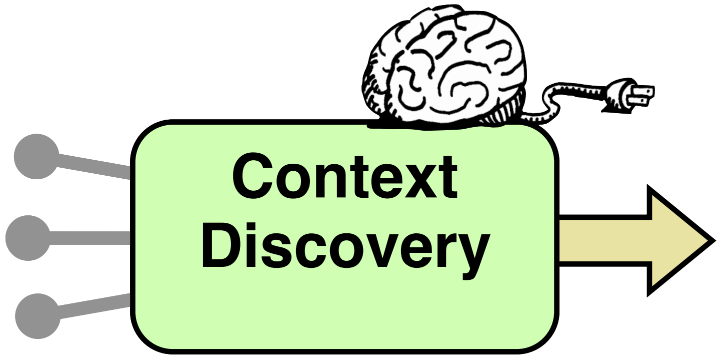
* cited by 66 according to Google Scholar on 10.12.2017

Context-Driven System Architecture



Contexts As Situation Reifiers

computationally accessible data



Context Discovery

well-defined situations

Z axis = 0.03 ··· Landscape orientation

Battery charge = 220 mAh ··· Low battery charge

Idle cycle

J load

User agent

x

no s

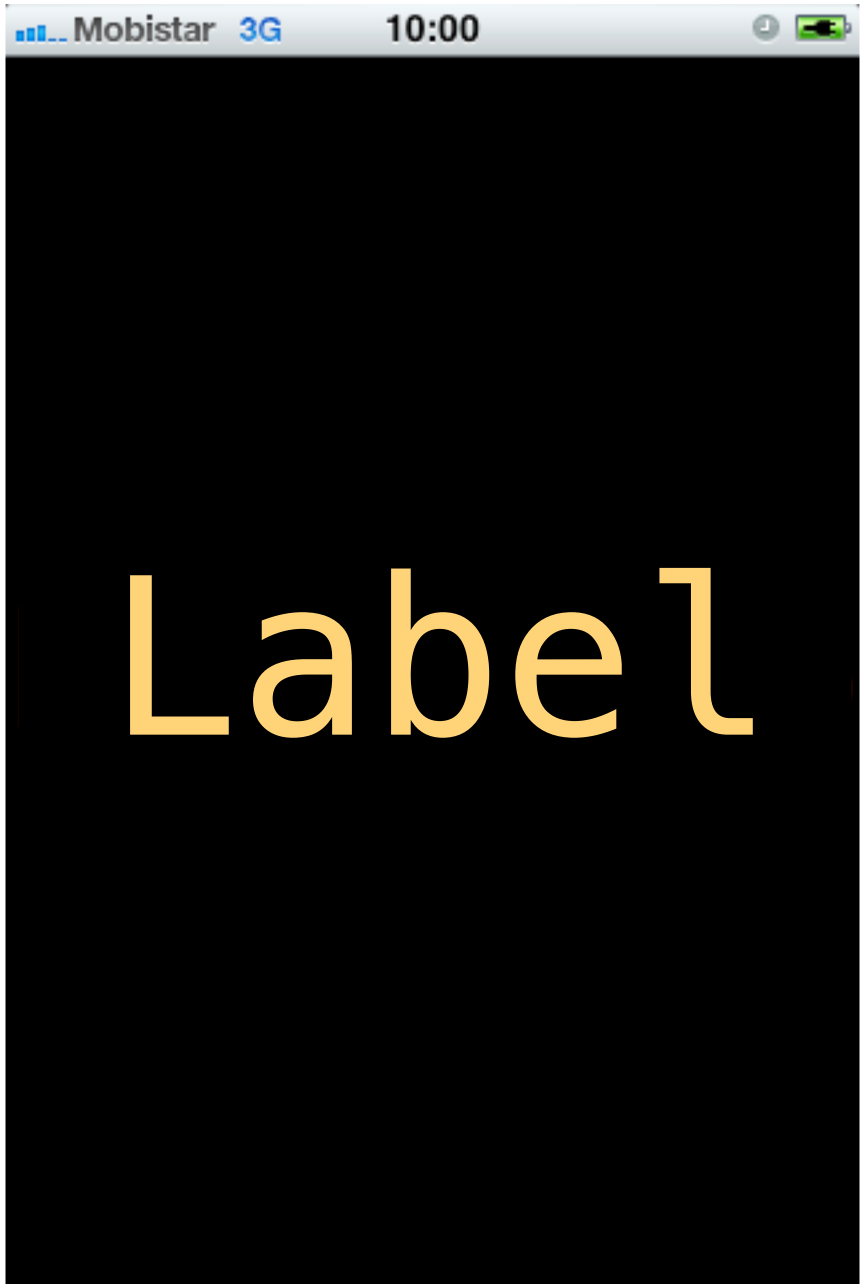
be taken

... in JavaScript

```
LowBattery = new Context();

window.addEventListener('batterystatus',
  function (battery) {
    if (battery.level < 30)
      LowBattery.activate();
    else
      LowBattery.deactivate(); });
```

for which adapted application behaviour can be defined

Minimalistic Case Study



Standard Widget Spec

UILabel class

drawTextInRect:

Draws the receiver’s text in the specified rectangle.

```
- (void)drawTextInRect:(CGRect)rect
```

Parameters

rect

The rectangle in which to draw the text.

Discussion

You should not call this method directly. This method should only be overridden by subclasses that want to modify the default drawing behavior for the label’s text.

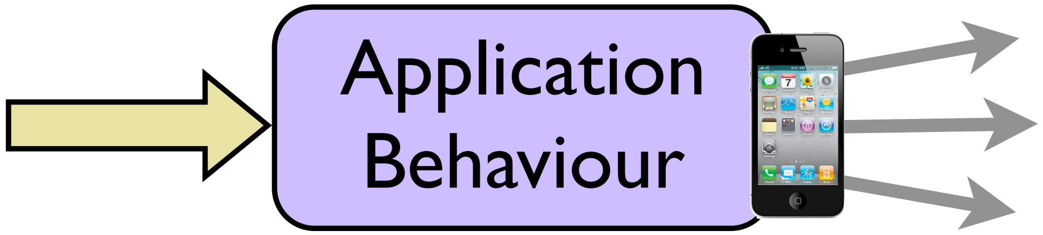
Availability

Available in iOS 2.0 and later.

Declared In

`UILabel.h`

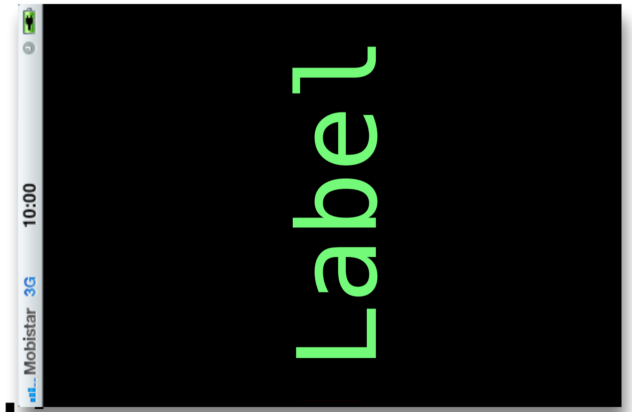
Main Idea



Open classes
Objective-C

COP
Subjective-C

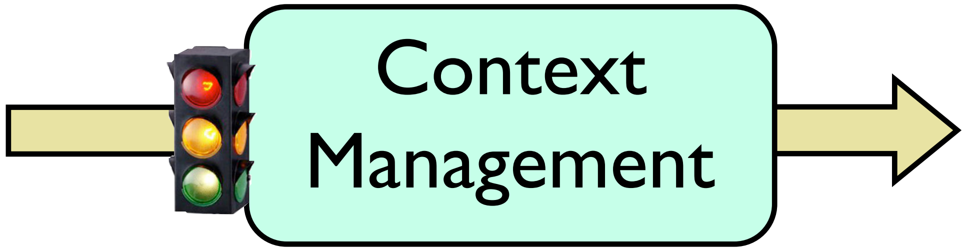
```
@implementation UILabel (color)
@contexts Landscape
- (void)drawTextInRect:(CGRect)rect {
   self.textColor = [UIColor greenColor];
   return @resend();
}
@end
```

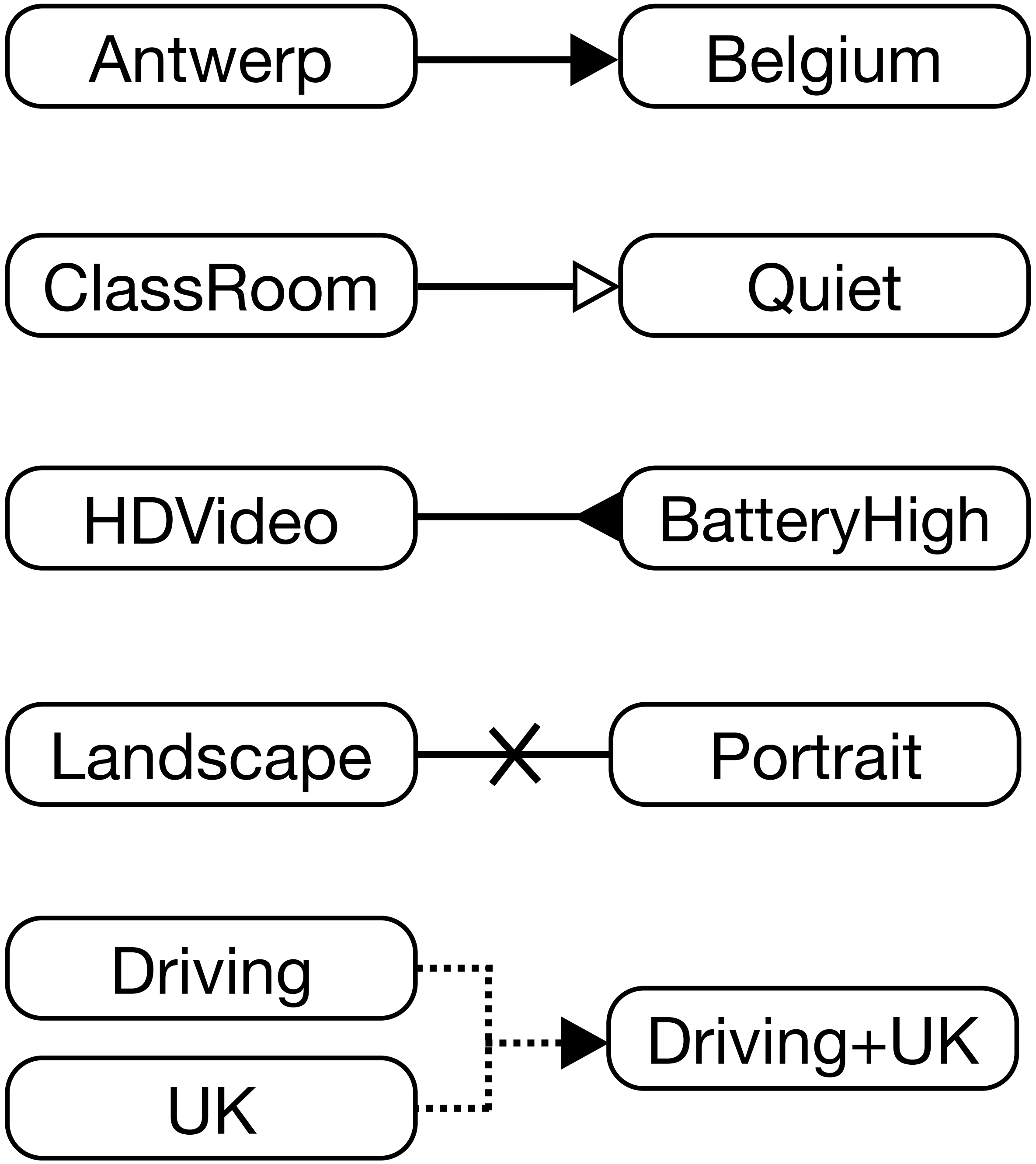


Label

- ✓ Adaptation of any existing component
- ✓ No access to original source code needed
- ✓ Adaptations can be cleanly modularised

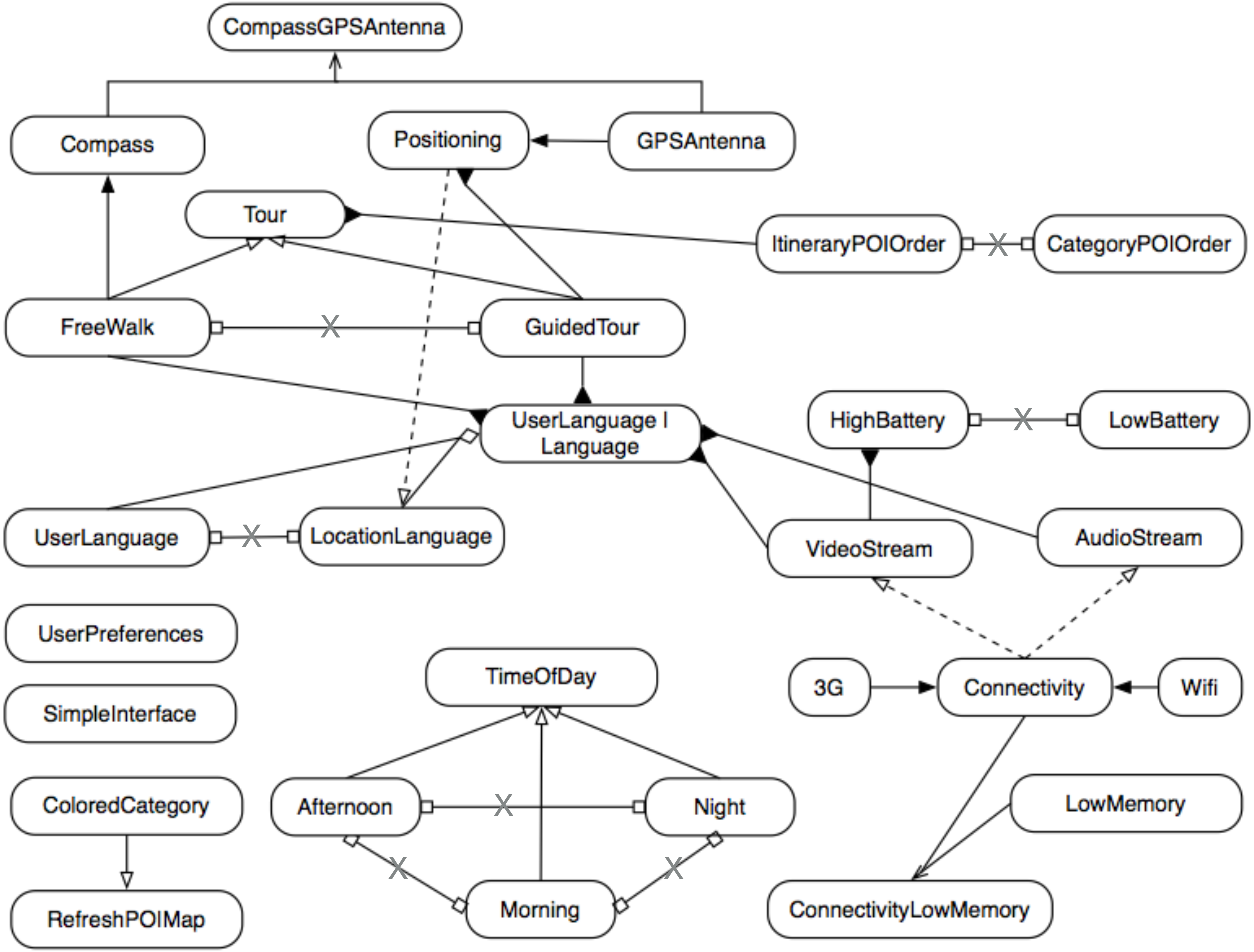
Context Dependencies





Implication	`Antwerp => Belgium`
Suggestion	`ClassRoom -> Quiet`
Requirement	`HDVideo =< BatteryHigh`
Exclusion	`Landscape >< Portrait`
Combination	`Driving + UK`

Context Dependency Graph

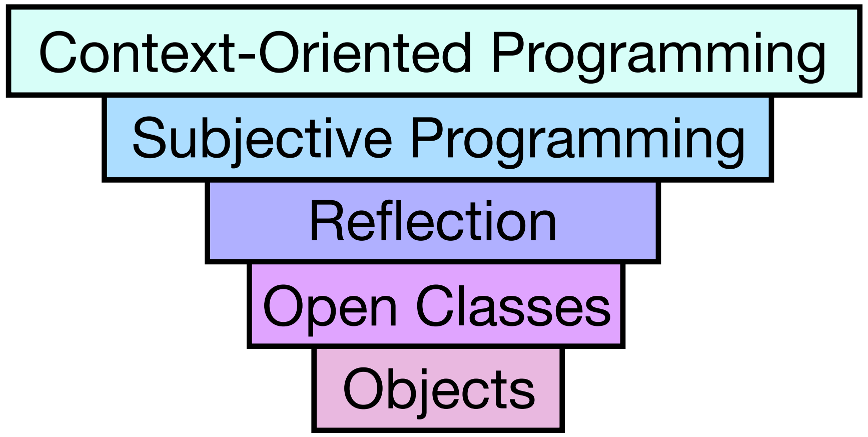


Summary

Subjective-C

language abstractions for adaptation to context … with sound technical underpinnings

```
@context Landscape
-id behaviour {
          // context-specific logic
}
```



Context-Oriented Programming

Subjective Programming

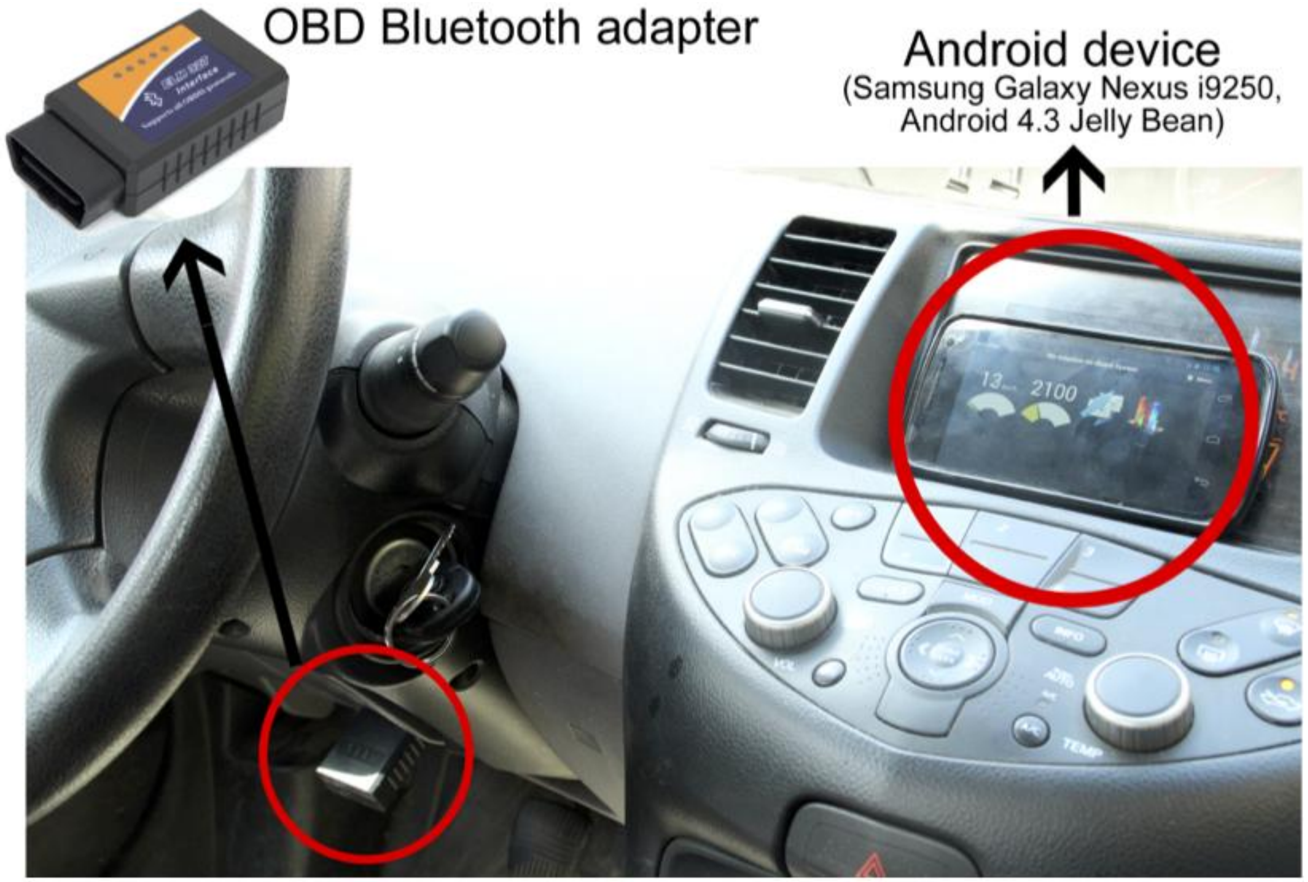
Reflection

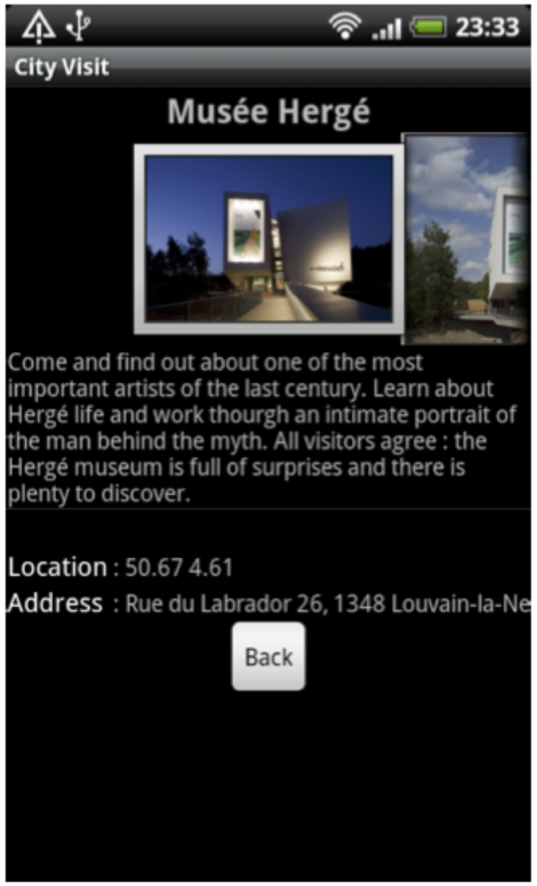
Open Classes

Objects

- ✓ Clean application logic
- ✓ Clean modularisation of adaptations
- ✓ Context reification and management
- ✓ Run-time behaviour adaptation of any component (incl. 3rd party)
  - ✓ No need for recompilation or access to original source code
- ✓ Maximises adaptation points while avoiding architectural burden
- ✓ Scoped adaptations

Some Case Studies





City Visit

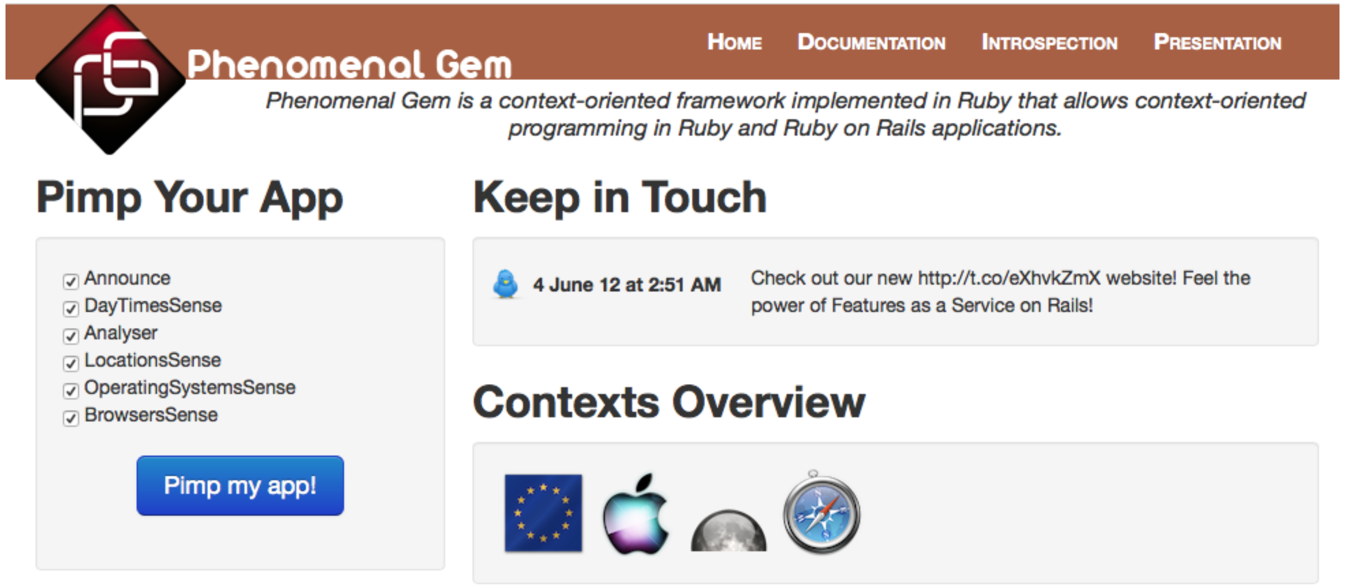
Musée Hergé

Come and find out about one of the most important artists of the last century. Learn about Hergé life and work thourgh an intimate portrait of the man behind the myth. All visitors agree : the Hergé museum is full of surprises and there is plenty to discover.

Location : 50.67 4.61

Address : Rue du Labrador 26, 1348 Louvain-la-Ne

Back



Phenomenal Gem

Phenomenal Gem is a context-oriented framework implemented in Ruby that allows context-oriented programming in Ruby and Ruby on Rails applications.

Pimp Your App

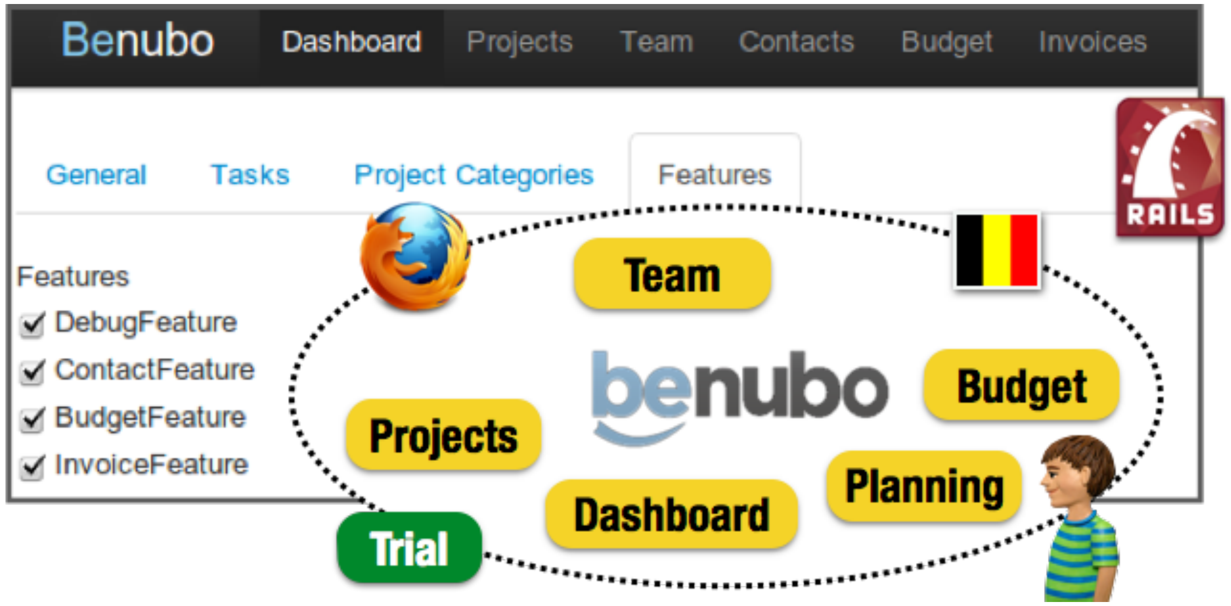
- ☑ Announce
- ☑ DayTimesSense
- ☑ Analyser
- ☑ LocationsSense
- ☑ OperatingSystemsSense
- ☑ BrowsersSense

Pimp my app!

Keep in Touch

4 June 12 at 2:51 AM Check out our new http://t.co/eXhvkZmX website! Feel the power of Features as a Service on Rails!

Contexts Overview

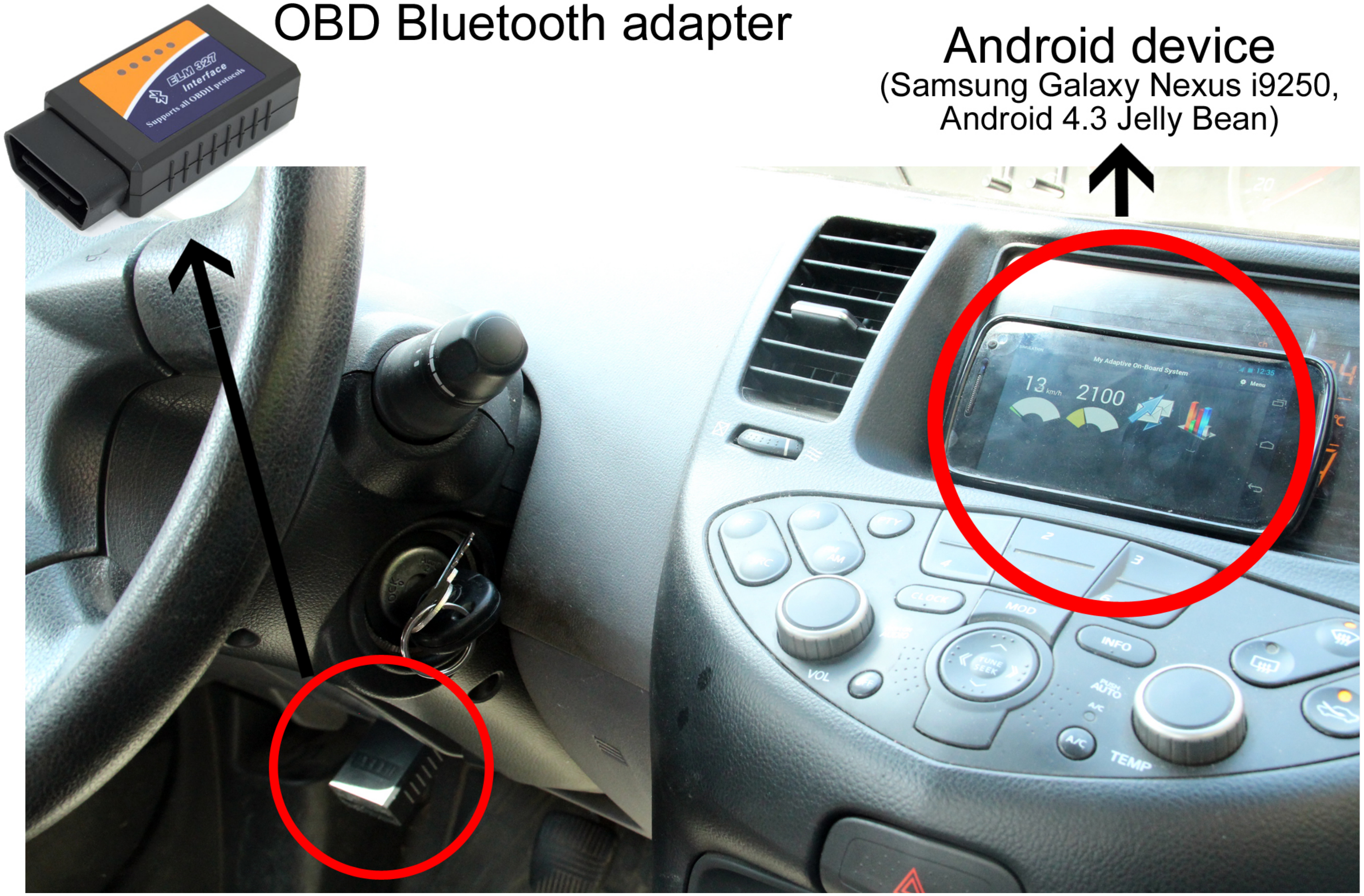


Features

- ☑ DebugFeature
- ☑ ContactFeature
- ☑ BudgetFeature
- ☑ InvoiceFeature

Case Study: On-Board Car System

Context Traits



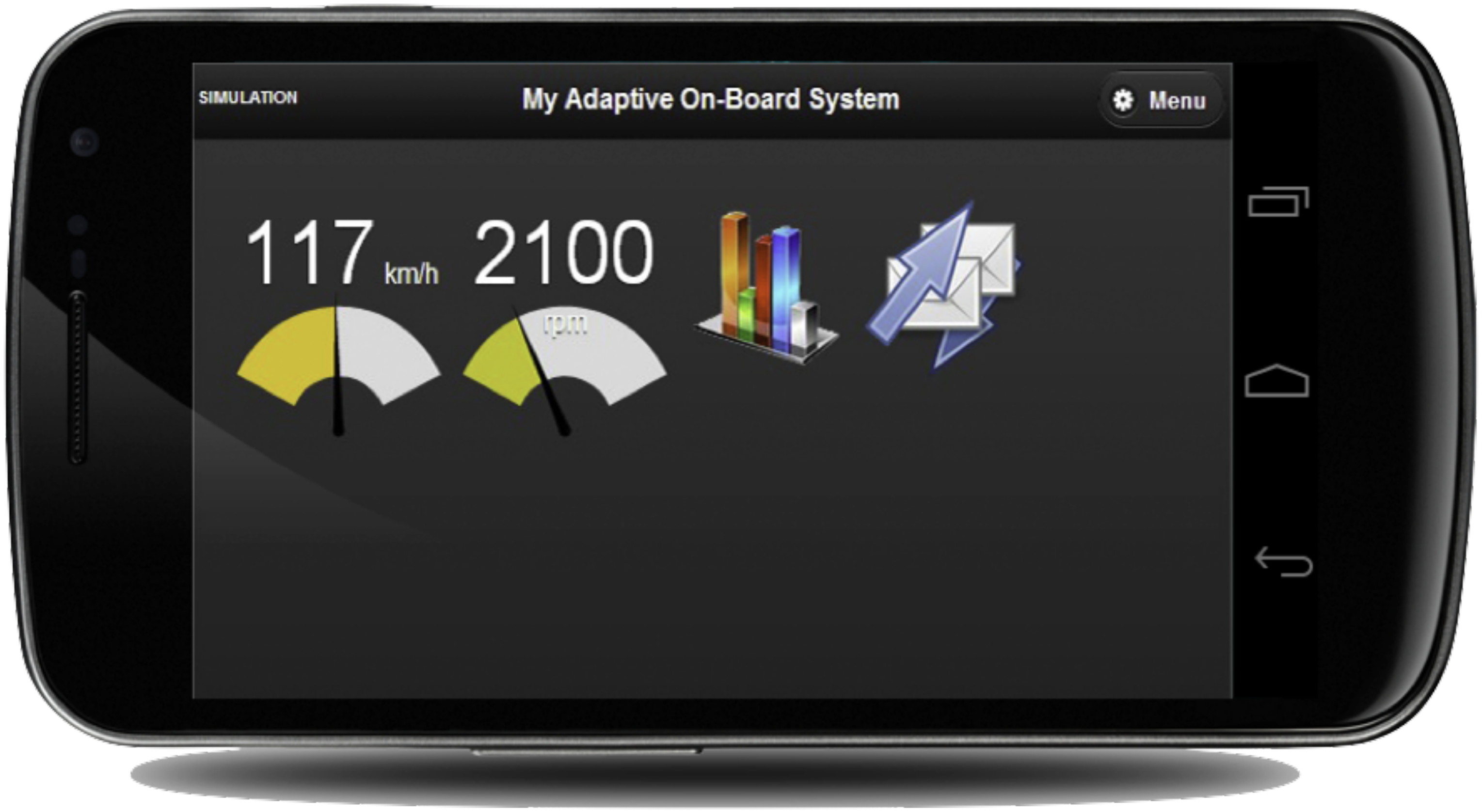
OBD Bluetooth adapter

Android device
(Samsung Galaxy Nexus i9250,
Android 4.3 Jelly Bean)

N. CARDOZO, K. MENS, S. GONZALEZ, P.-Y. ORBAN, W. DE MEUTER. Features on Demand.
International Workshop on Variability Modelling of Software-Intensive Systems, **2014**.

Case Study: On-Board Car System

Context Traits

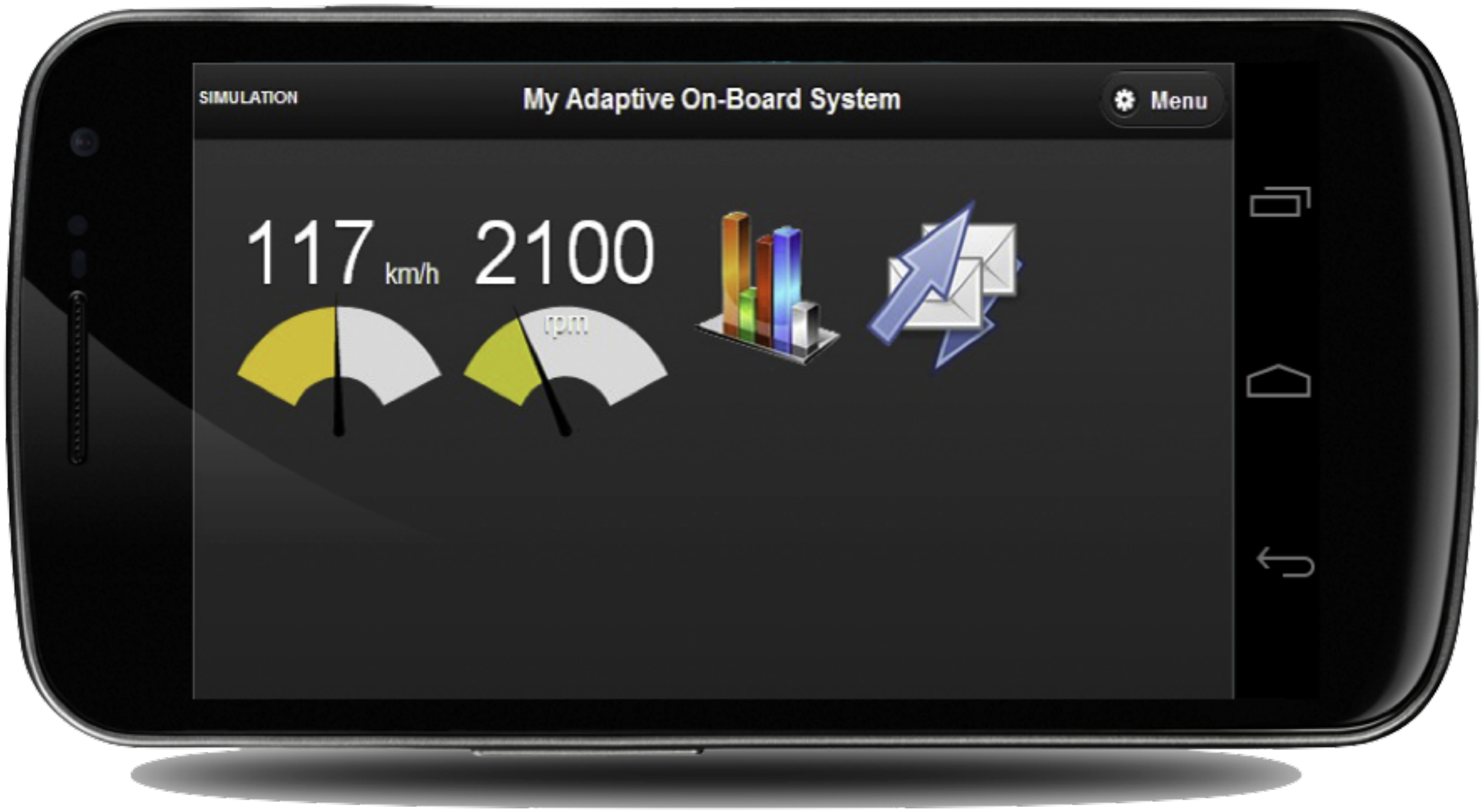


P.-Y. ORBAN. Using Context-Oriented Programming for Building Adaptive Feature-Oriented Software for Car On-Board Systems. Master thesis in Computer Science, Université catholique de Louvain, 2013

Case Study: On-Board Car System

Context Traits

Context-specific features



SIMULATION

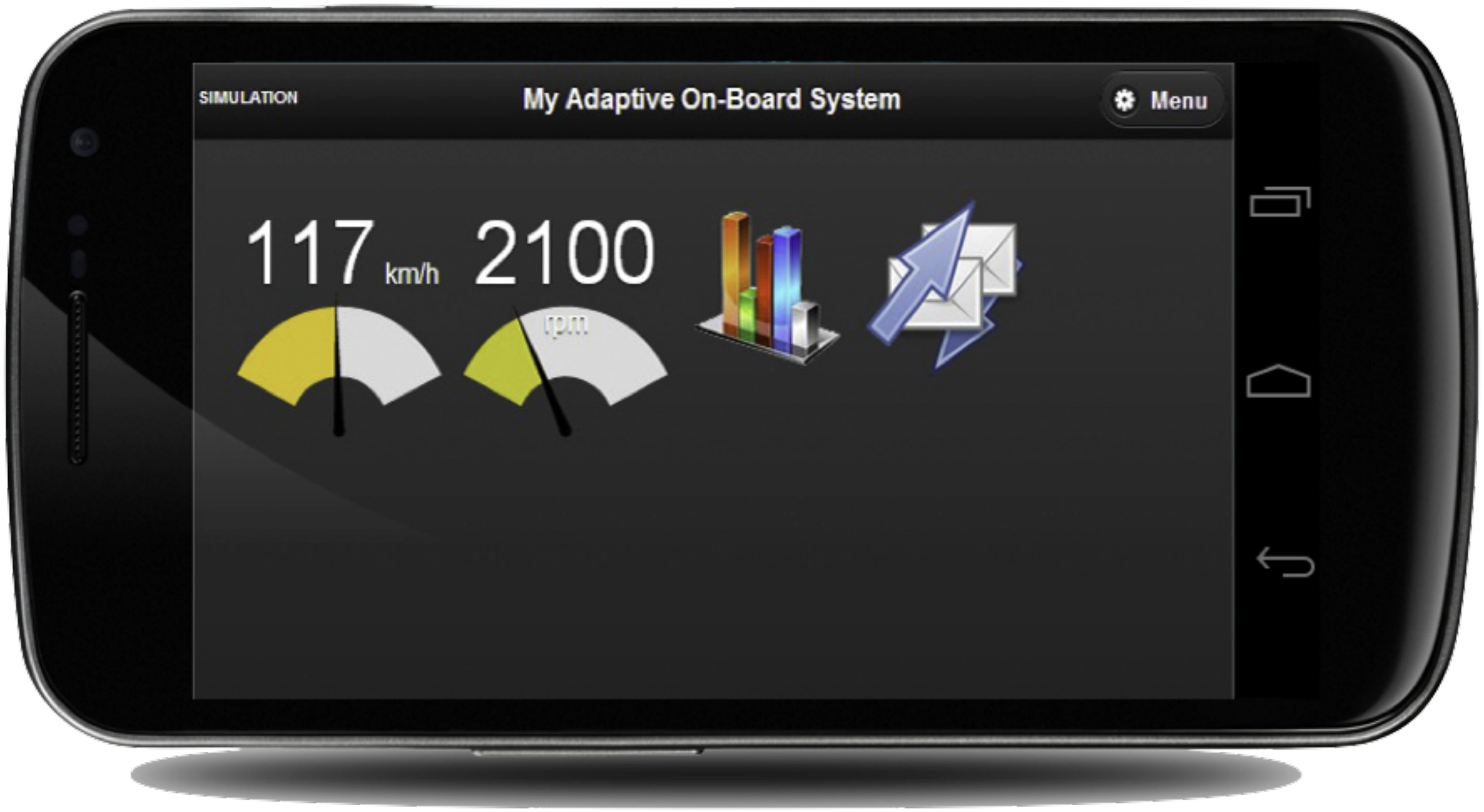
My Adaptive On-Board System

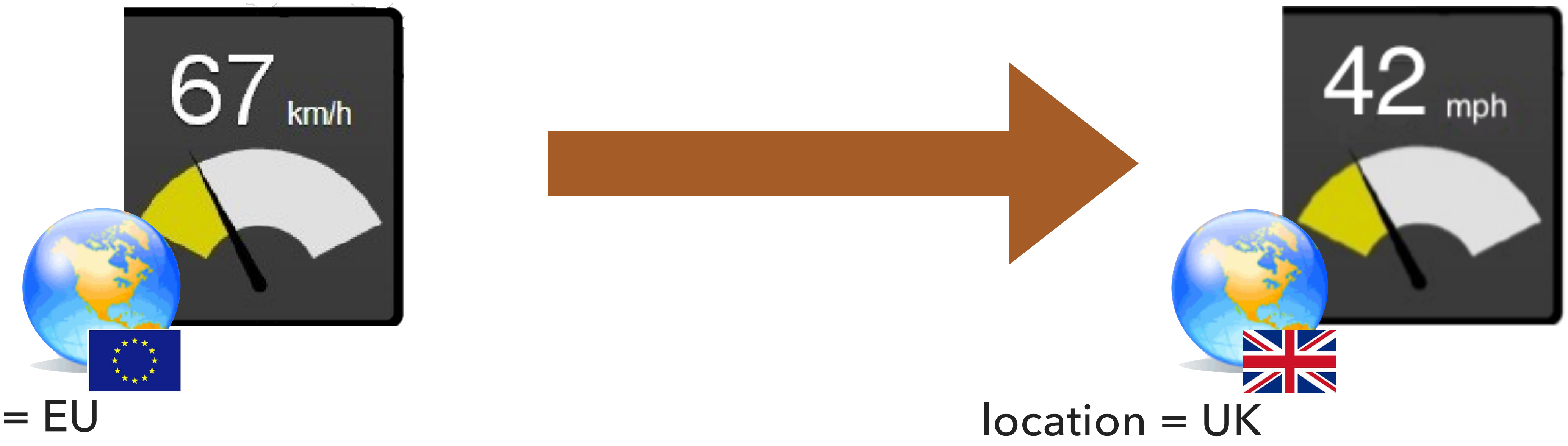
Display speed reading using
the metric system units

location = EU

Case Study: On-Board Car System

Context changes trigger behavioural adaptation





Case Study: On-Board Car System

Context Traits

```
ImperialSystem = Trait({
   var CONV_RATIO = 0.621371192;
   getSpeed: function(msg) {
     _val = this.proceed();
     Math.round _val * CONV_RATIO; }

   getHtml: function() {
     display.setGaugeDisplay(this.proceed().replace("km/h", "mph")); }
   });
```

Display speed reading using the metric system units

Display speed reading using the imperial system units

67 km/h
42 mph

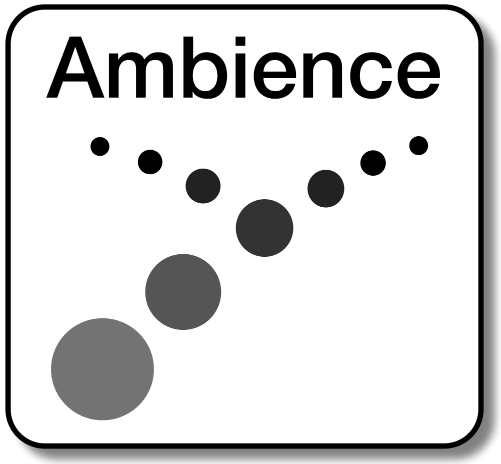
location = EU

location = UK

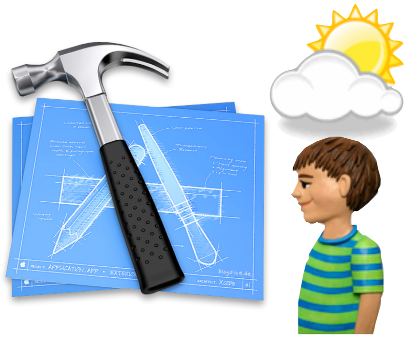
How?

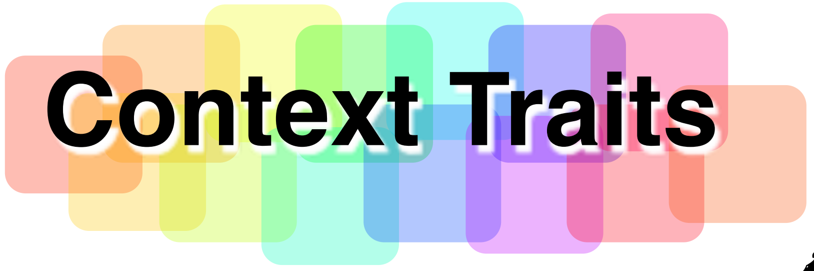
implementation of context-driven software adaptability through ...

... method dispatch ... and method pre-dispatch.



Subjective-C





2008 — 2010 — 2013

Method Pre-Dispatch

Whenever a **context is (de)activated**

For every class c and selector s the context adapts, find all *active** methods

$$M(c, s)=\{ m_1, m_2, m_3, \ldots, m_n \}$$

Reorder them according to specificity

$$m_1 < m_2 < m_3 < \ldots < m_n$$

and **deploy** the first one m_1

- m_1 is the most specific implementation for the current context
- `resend` invokes the remaining methods in order
- m_n is (usually) the default implementation

* methods defined for contexts that are currently active

Method Dispatch

Whenever a **message is sent**
(to receiver *r*, with selector *s* and arguments *a*)

Find all *active** methods that match the message

$$M(r,s,a) = \{ m_1, m_2, m_3, \ldots, m_n \}$$

Reorder them according to specificity

$$m_1 < m_2 < m_3 < \ldots < m_n$$

and **invoke** the first one m_1

- m_1 is the most specific implementation for the current context
- `resend` invokes the remaining methods in order
- m_n is (usually) the default implementation

* methods defined for contexts that are currently active

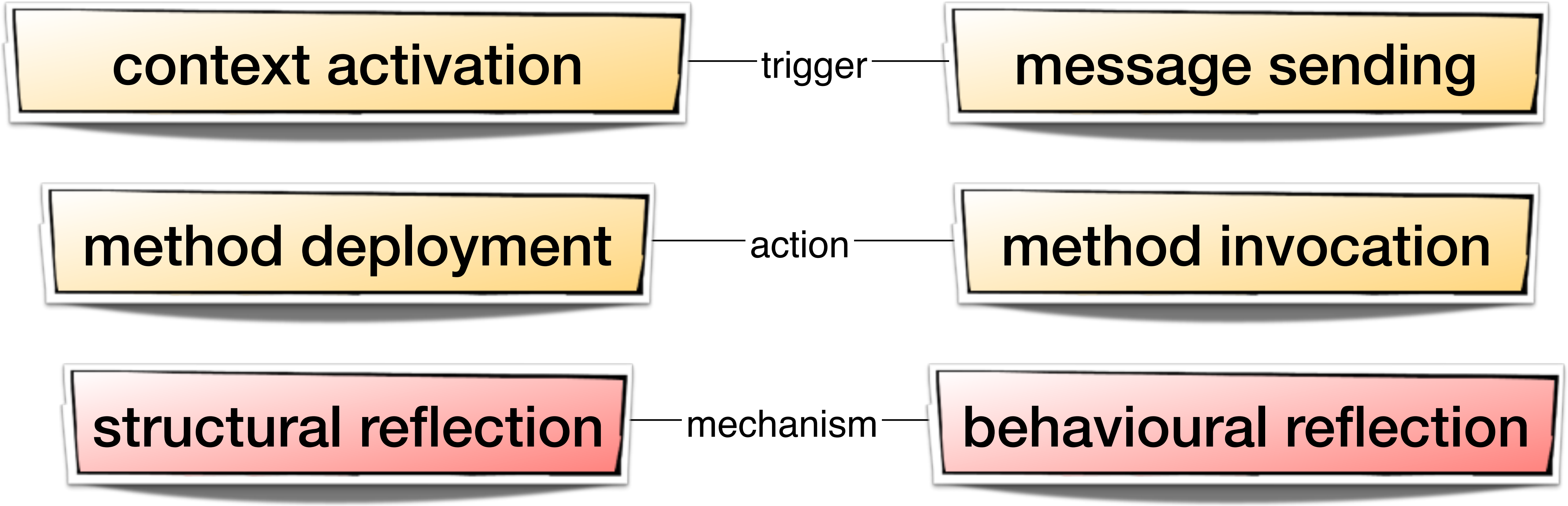
Comparison Of Implementation Techniques

Method Pre-Dispatch

$M(c, s)=\{ m_1, m_2, m_3, ..., m_n \}$

Method Dispatch

$M(r,s,a)=\{ m_1, m_2, m_3, ..., m_n \}$



context activation — trigger — message sending

method deployment — action — method invocation

structural reflection — mechanism — behavioural reflection

(more commonly supported)

(more powerful)

Method Pre-Dispatch

$$M(c, s)=\{ m_1, m_2, m_3, \ldots, m_n \}$$

$$M(r,s,a)=\{ m_1, m_2, m_3, \ldots, m_n \}$$

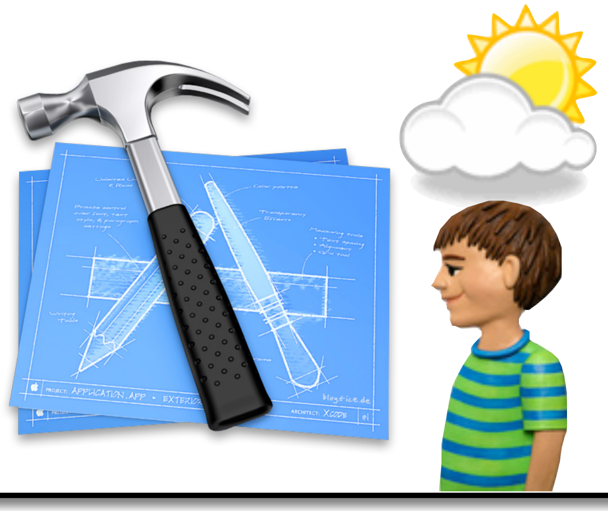
Is the method order always defined?

- Could there be no applicable methods?
  - ➡ default implementation
- Could there be non-comparable methods?
  - ➡ the order should be total
  - ➡ if not, we’re in **trouble**

Method Pre-Dispatch

Case Study

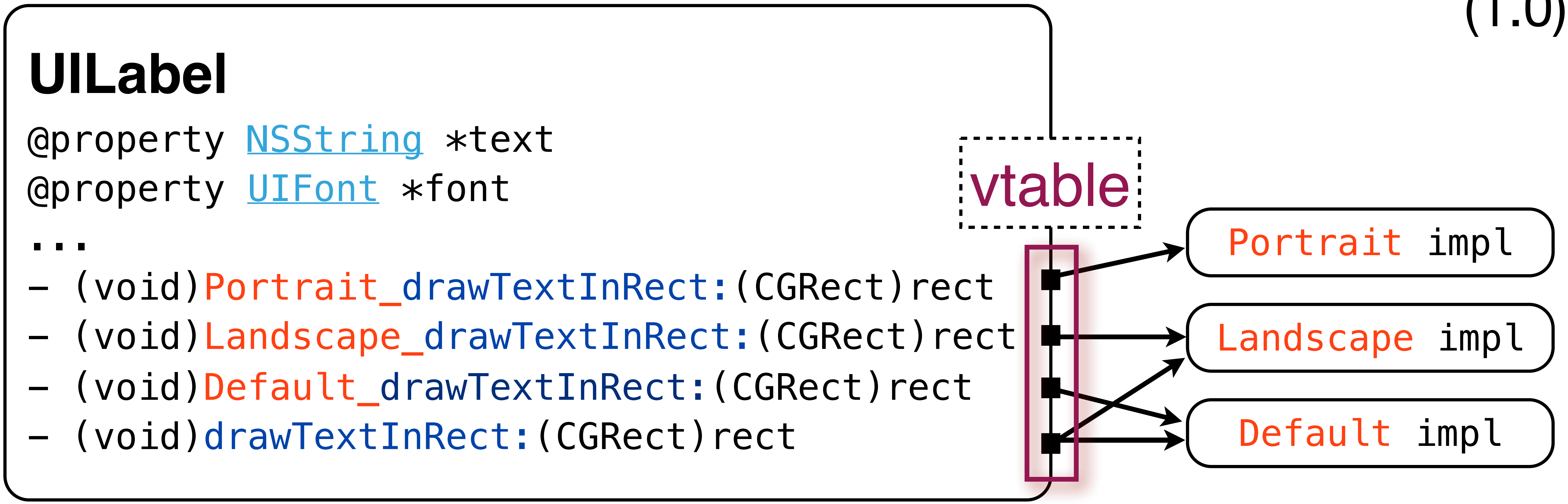
Subjective-C



Dynamic Method Pre-Dispatch

Subjective-C

(1.0)



UILabel

```
@property NSString *text
@property UIFont *font
...
- (void)Portrait_drawTextInRect:(CGRect)rect
- (void)Landscape_drawTextInRect:(CGRect)rect
- (void)Default_drawTextInRect:(CGRect)rect
- (void)drawTextInRect:(CGRect)rect
```

```
@activate(Landscape);

@deactivate(Landscape);
```

- no additional cost for method invocations
- cost incurred at context switching time

Structural Reflection in Objective-C

Subjective-C

Class Introspection

```
Method class_getInstanceMethod(Class aClass, SEL aSelector)
Method class_getClassMethod(Class aClass, SEL aSelector)
```

Class Intercession

```
BOOL class_addMethod(Class cls, SEL name, IMP imp, const char *types)
```

Method Introspection

```
IMP method_getImplementation(Method method)
```

Method Intercession

```
IMP method_setImplementation(Method method, IMP imp)
```

Behavioural Reflection in Objective-C

Subjective-C

Invocation Reification

```
...
NSMethodSignature *signature = defaultMethod->signature;
NSInvocation *invocation =
   [NSInvocation invocationWithMethodSignature:signature];
[invocation setTarget:receiver];
[invocation setSelector:adaptedMethod->selector];
va_list arguments;
va_start(arguments, methodSelector);
int arg = va_arg(arguments, int);
[invocation setArgument:&arg atIndex: 0];
...
```

Invocation Activation

```
[invocation invoke];
...
void *result;
[invocation getReturnValue:result];
return result;
```

Summary

- ✓ Definition of context
  - ✓ Reifies the circumstances in which the software executes
  - ✓ Frame of reference to define adaptations
- ✓ Behaviour adaptability
  - ✓ Language abstractions
  - ✓ Modularity of adaptations
- ✓ Context discovery
- ✓ Context management
- ➡ Consistency management

Conclusion : Mind The Context !

Richard Gabriel, 2006

We need to use softer, more dynamic architectures that support adding or replacing modules after deployment and architectures where objects can be repaired in situ, methods changed / added, internal state restructured, and object hierarchies rewired. We also need new types of languages to describe the architecture of our systems.

Additional Reading

N. CARDOZO, S. GONZALEZ, K. MENS, R. VAN DER STRAETEN, J. VALLEJOS, T. D'HONDT. Semantics for Consistent Activation in Context-Oriented Systems. Information and Software Technology, 58:71-94, **2015**.

N. CARDOZO, K. MENS, S. GONZALEZ, P.-Y. ORBAN, W. DE MEUTER. Features on Demand. International Workshop on Variability Modelling of Software-Intensive Systems, **2014**.

N. CARDOZO, S. GONZALEZ, K. MENS, R. VAN DER STRAETEN, T. D'HONDT. Modeling and Analyzing Self-adaptive Systems with Context Petri Nets. Symposium on Theoretical Aspects of Software Engineering, **2013**.

S. GONZALEZ, K. MENS, M. COLACIOIU, W. CAZZOLA. Context Traits: dynamic behaviour adaptation through run-time trait recomposition. International conference on Aspect-Oriented Software Development, **2013**.

E. BAINOMUGISHA, A. CADIZ, P. COSTANZA, W. DE MEUTER, S. GONZALEZ, K. MENS, J. VALLEJOS, T. VAN CUTSEM. Language Engineering for Mobile Software. Chapter of the *Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications*, IGI Global, **2012**.

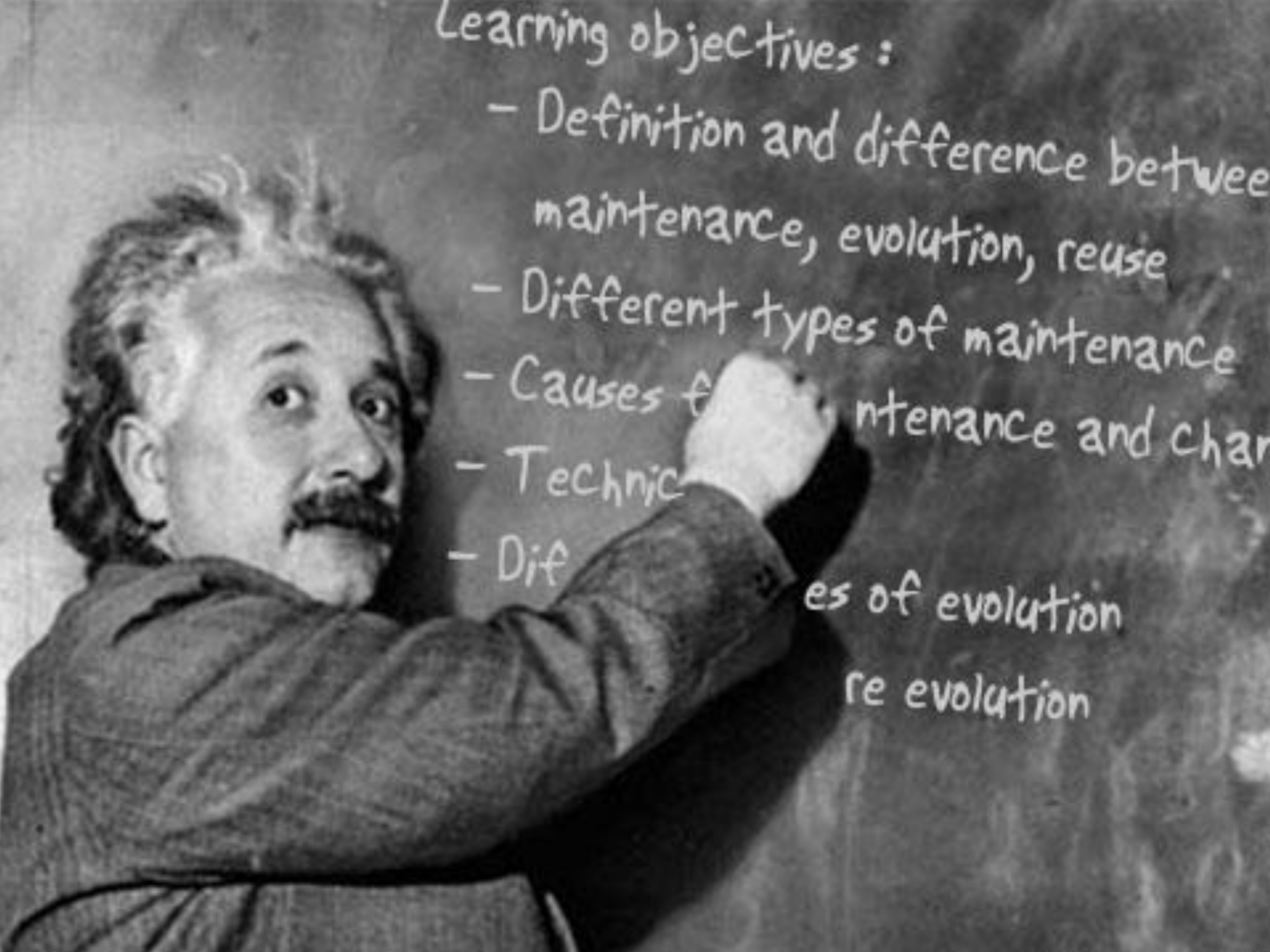
N. CARDOZO, S. GUNTHER, K. MENS, T. D'HONDT. Feature-Oriented Programming and Context-Oriented Programming: Comparing Paradigm Characteristics by Example Implementations. International Conference on Software Engineering Advances, **2011**.

S. GONZALEZ, N. CARDOZO, K. MENS, A. CADIZ, J-C. LIBBRECHT, J. GOFFAUX. Subjective-C: Bringing Context to Mobile Platform Programming. International Conference on Software Language Engineering, **2010**.

J. VALLEJOS, S. GONZALEZ, P. COSTANZA, W. DE MEUTER, T. D'HONDT, K. MENS. Predicated Generic Functions: Enabling Context-Dependent Method Dispatch. International Conference on Software Composition, **2010**.

S. GONZALEZ, K. MENS, A. CADIZ. Context-Oriented Programming with the Ambient Object System. Journal of Universal Computer Science, 14(20):3307–3332, **2008**.

S. GONZALEZ, K. MENS, P. HEYMANS. Highly Dynamic Behaviour Adaptability through Prototypes with Subjective Multimethods. Symposium on Dynamic Languages, **2007**.



Learning objectives :

- Definition and difference betwee maintenance, evolution, reuse
- Different types of maintenance
- Causes f ntenance and char
- Technic
- Dif

es of evolution

re evolution

POSSIBLE QUESTIONS

- What is the main difference between traditional software systems and context-aware systems?
- Explain, in your own words, what problems context-oriented programming tries to solve.
- Explain, in your own words, what context-oriented programming is.
- Two different techniques exist for implementing dynamic adaptation of software behaviour to context: method dispatch and method pre-dispatch. Briefly explain and compare these two techniques.
- One particular technique for implementing dynamic adaptation of software behaviour to context is that of method pre-dispatch. Explain that technique in detail and illustrate it with a concrete example.

CLASS… IS… DISMISSED.