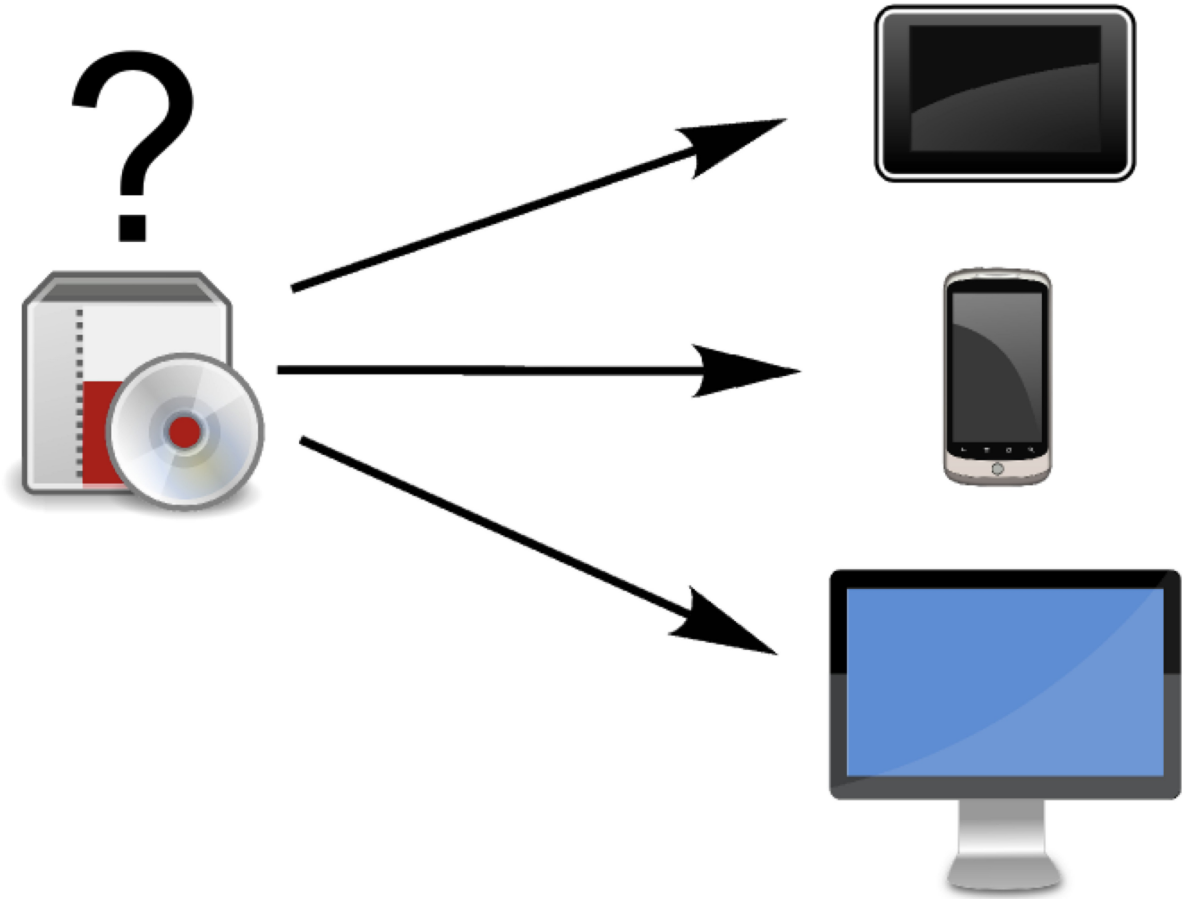
Towards cross-platform application development

Glenn Cavarlé
Alain Plantec
Vincent Ribaud
Christophe Touzé

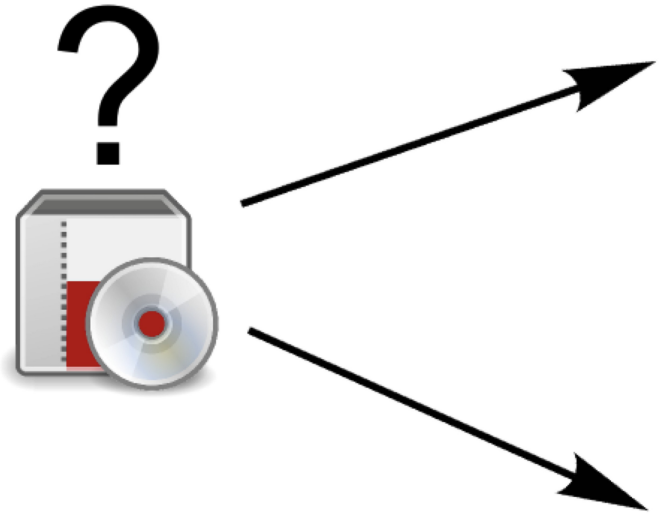
Problem #1

- One application but several target platforms



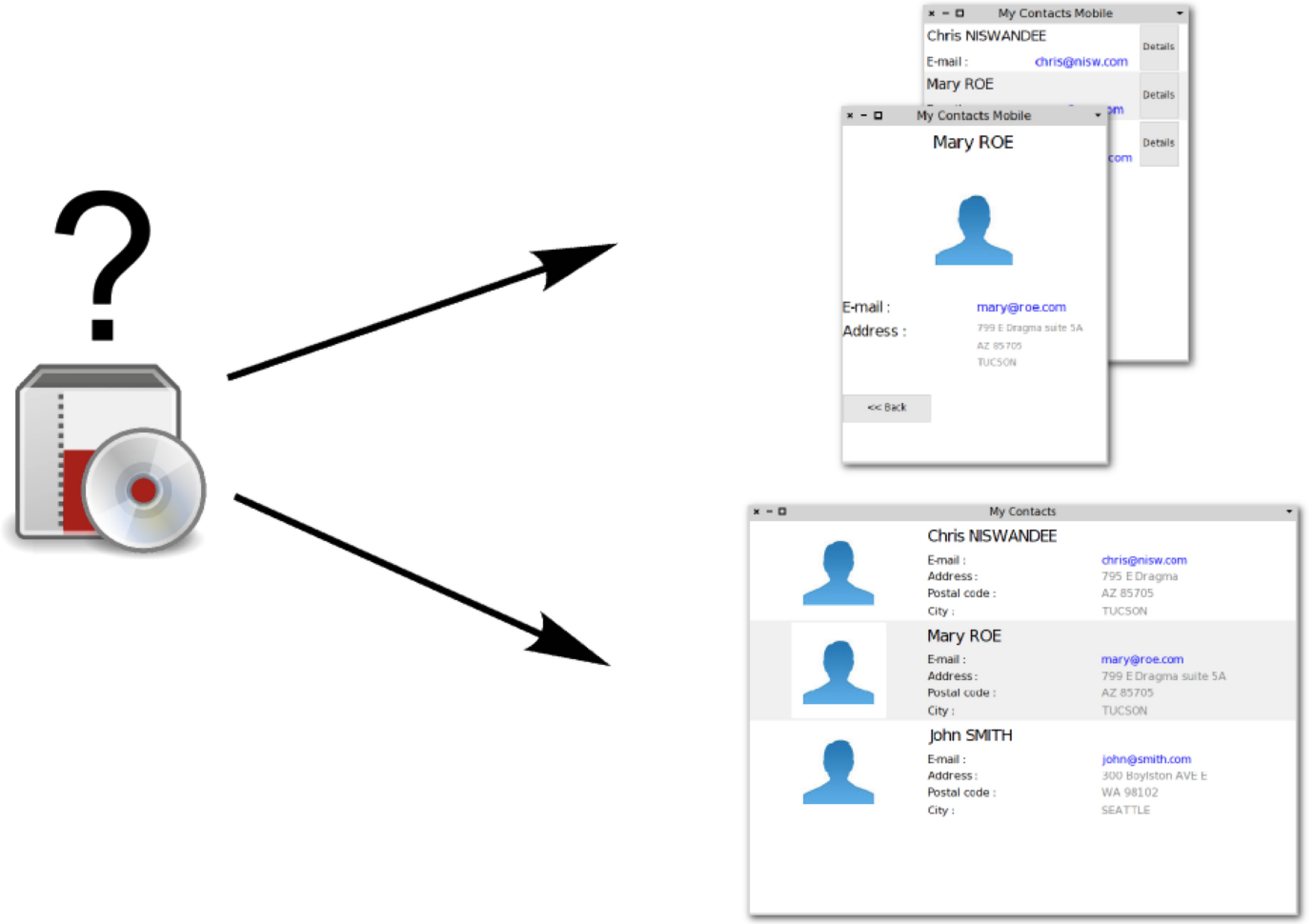
Problem #2

- One application but several presentations



Problem #3

- One application but specific behaviours



In short

- Native development context =
  - An application model (data and behavior)
  - A GUI

- Cross-platform development context =
  - An application model (data and behavior)
  - A GUI
  - ***+ A platform***

Our goals

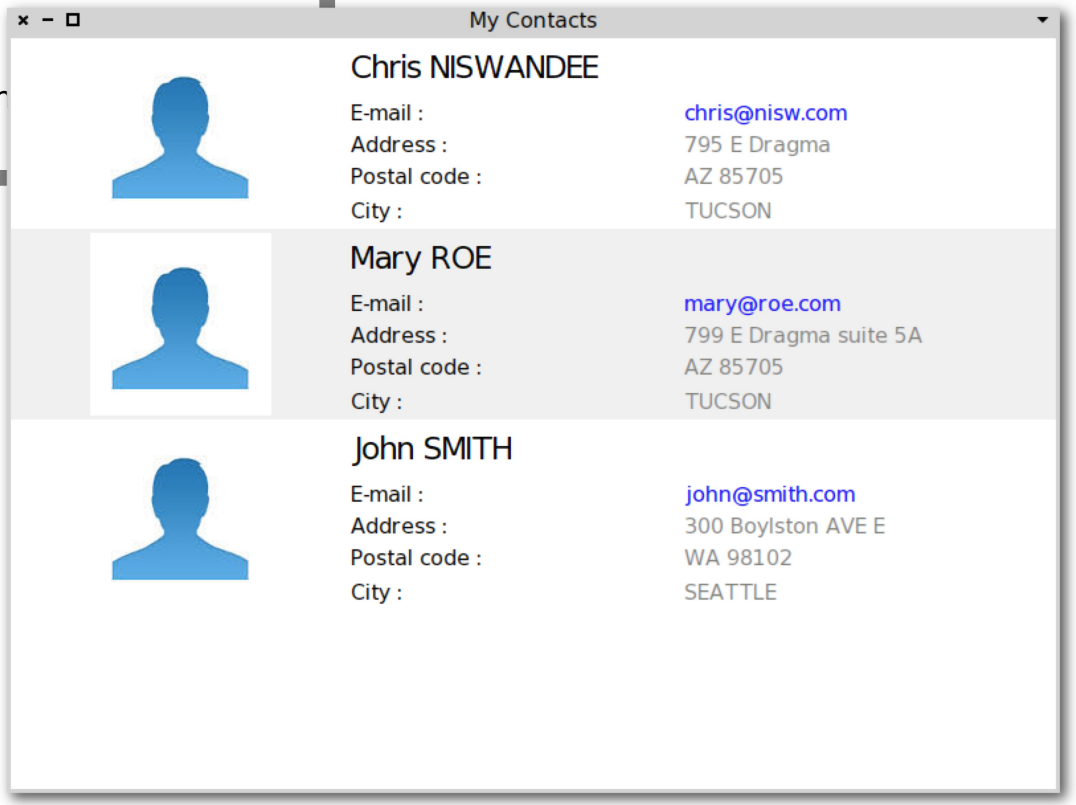
- Whatever the platform
  - A single development environment
  - A single application model
  - Be allowed to run an application without code generation
  - Be allowed to use code generation but late in the development process

A cross-platform framework

Run an application

```
DAEnvironment new
        platform: DATestPlateform desktop800x600 ;
        requirements: DATestRequirements mobileAndDesktop;
        adaptation: DAMorphAdaptation new;

        withinDo: [
                        MyContactsApplication r
                ]
```



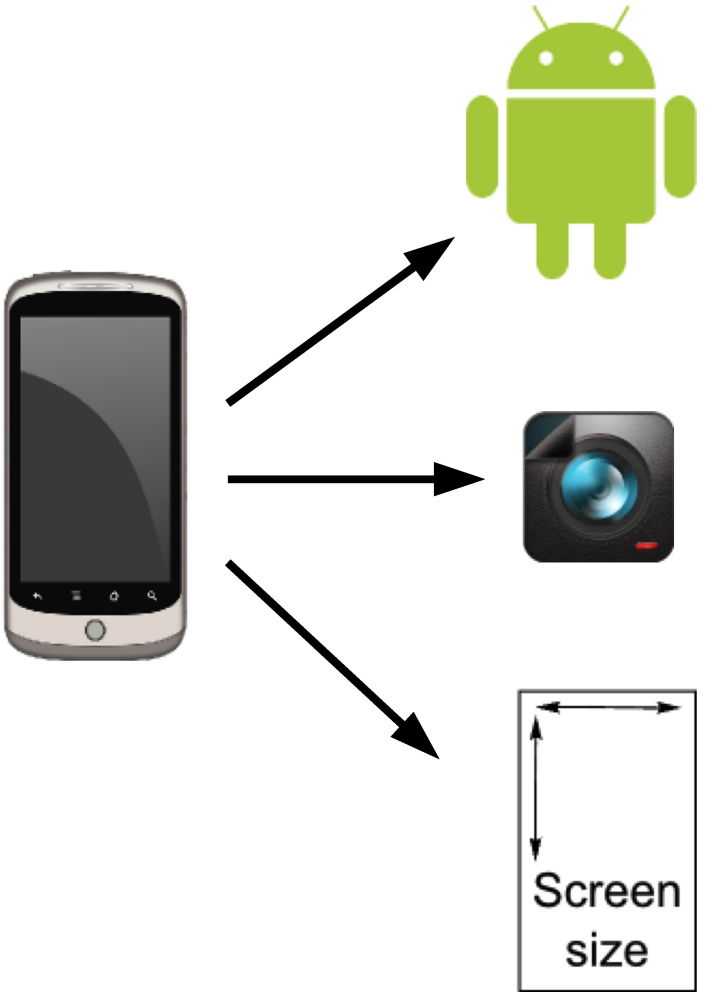
Run an application

```
DAEnvironment new
    platform: DATestPlateform desktop800x600 ;
    requirements: DATestRequirements mobileAndDesktop;
    adaptation: DAMorphAdaptation new;

    withinDo: [
            MyContactsApplication new open
        ]
```

Modeling platforms

```
DAPlateform new
    propertyAt: #kind put: #mobile;
    propertyAt: #os put: #android;
    propertyAt: #camera put: true;
    propertyAt: #screenWidth put: 360;
    propertyAt: #screenHeight put: 460;
```

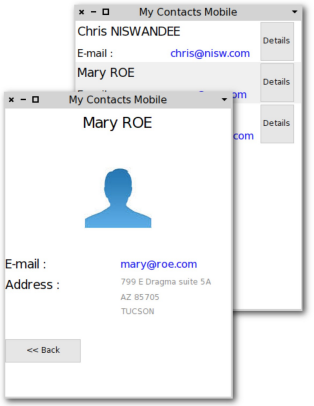


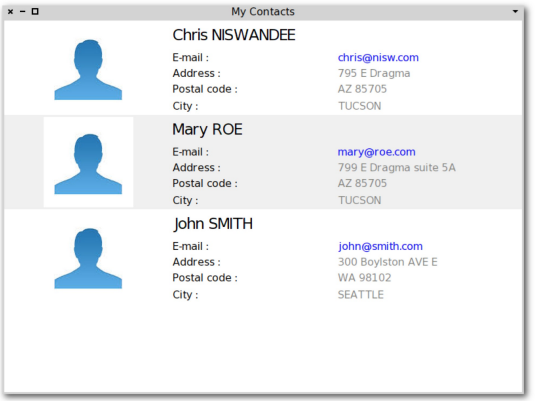
Modeling constraints

```
DARequirementRepository new

  add:
            (DARequirement new
                  name: #mobile;
                  constraint: [:plfm |
                        (plfm propertyAt: #kind) = #mobile ];
                  yourself);


 add:
            (DARequirement new
                  name: #largeScreen;
                  constraint: [:plfm |
                        (plfm propertyAt: #screenWidth) >= 800 and:
                        (plfm propertyAt: #screenHeight) >= 600];
                  yourself)
```

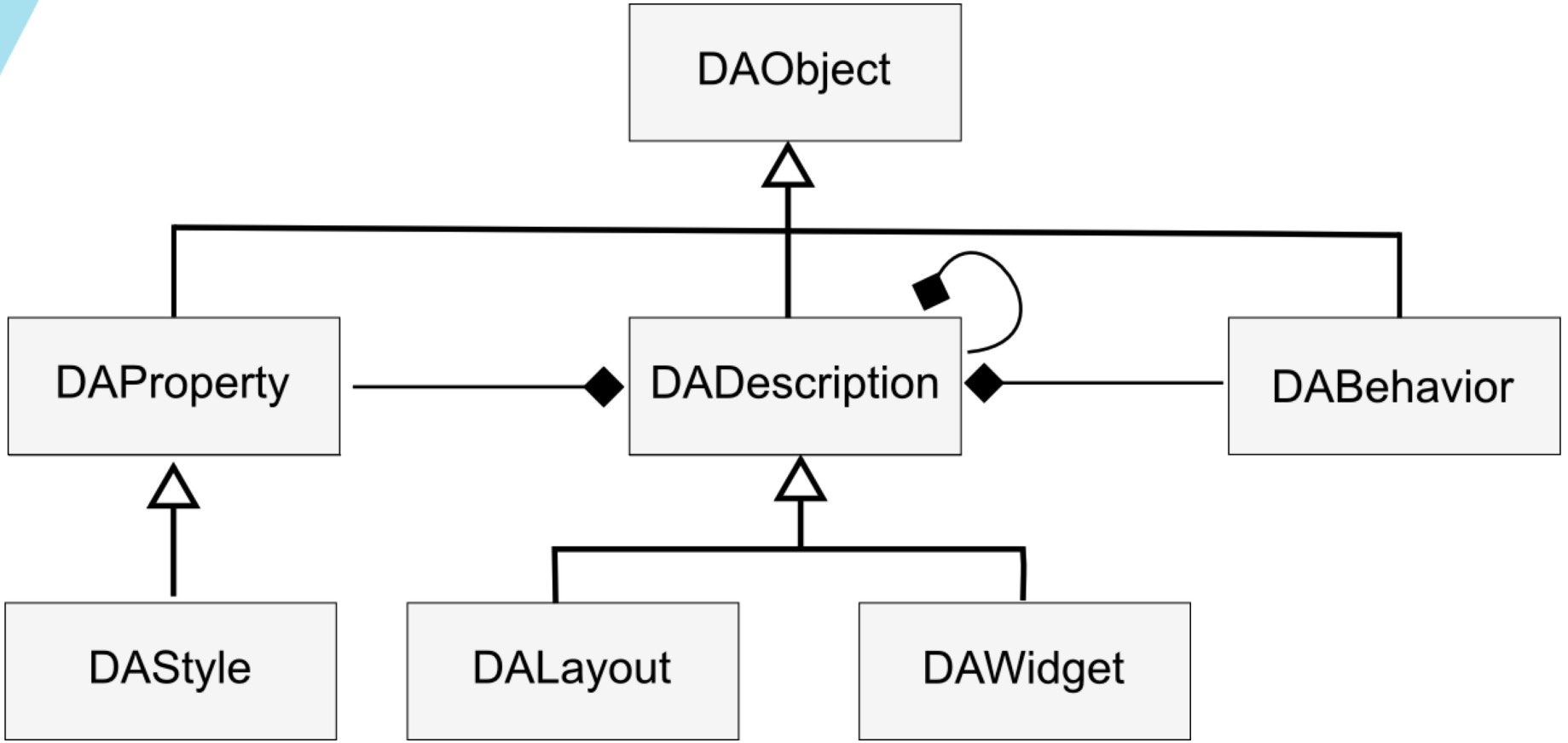




What is an object in Dali

- Made of entities
  - Properties
  - Behaviors
  - GUI

More formally



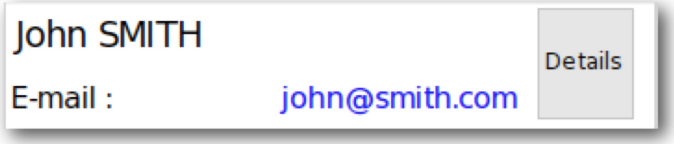
Modeling objects

```
ContactItemWidget>>declareDetailsButton
    <dali:#mobile>
    ^ DAButton new
            rid: #detailsButton;
            backgroundColor: Color lightGray;
            yourself
```

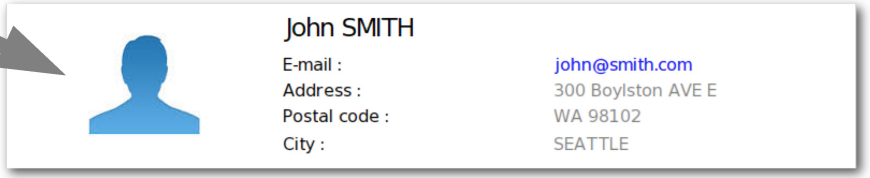
Modeling objects

(plfm propertyAt: **#kind**) = **#mobile**

```
ContactItemWidget >> declareDetailsButton
    <dali:#mobile>
    ^ DAButton new
        rid: #detailsButton;
        backgroundColor: Color lightGray;
        yourself
```



DATestPlateform mobile360x480



DATestPlateform desktop800x600

Declaring properties

```
ContactItemWidget>>declareContactModel
<dali>
^ DAProperty new
        rid: #contactModel;
        kind: #ContactModel;
        yourself
```

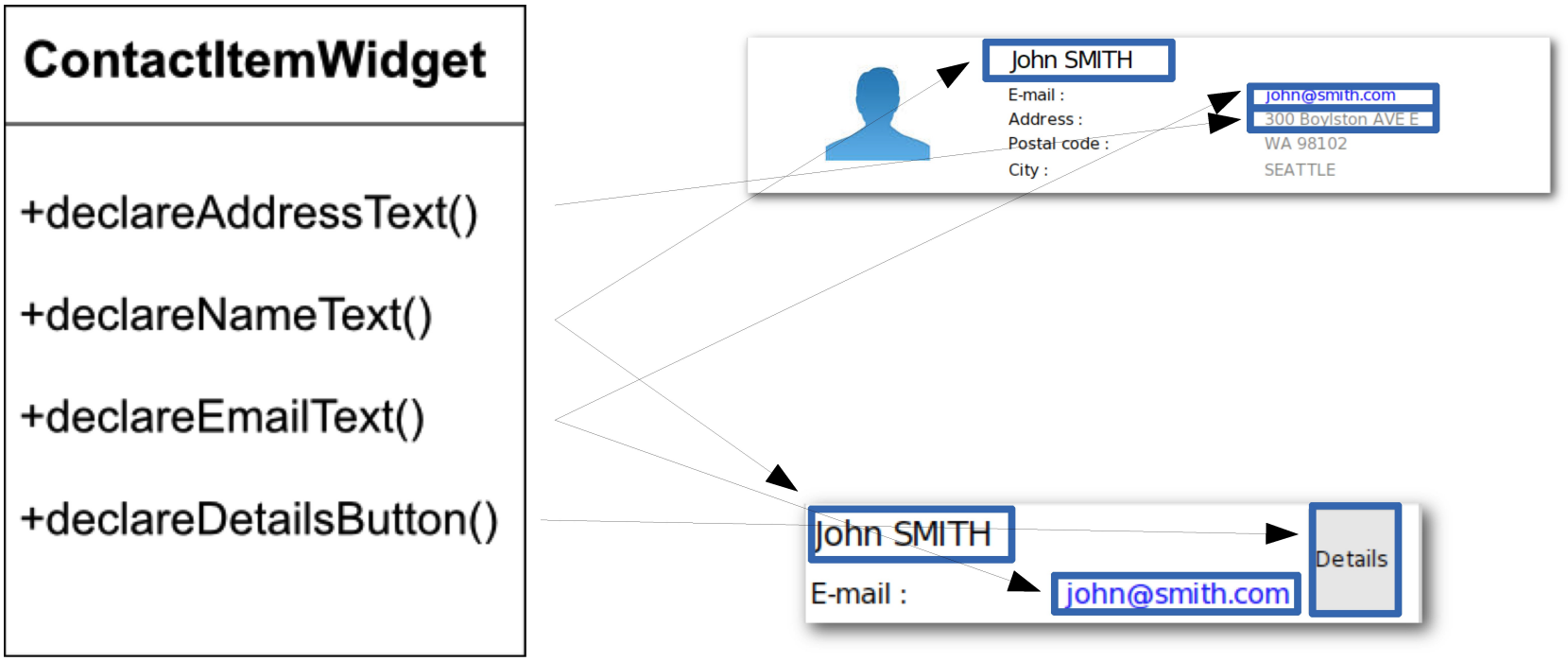
Declaring behaviours

```
ContactItemWidget>>declareClickReaction
    <dali:#mobile>
    ^ DAReaction new
            event: DAClickEvent;
            senderAccessor: #detailsButton asDaliAccessor;
            opReference: #openDetails;
            yourself
```

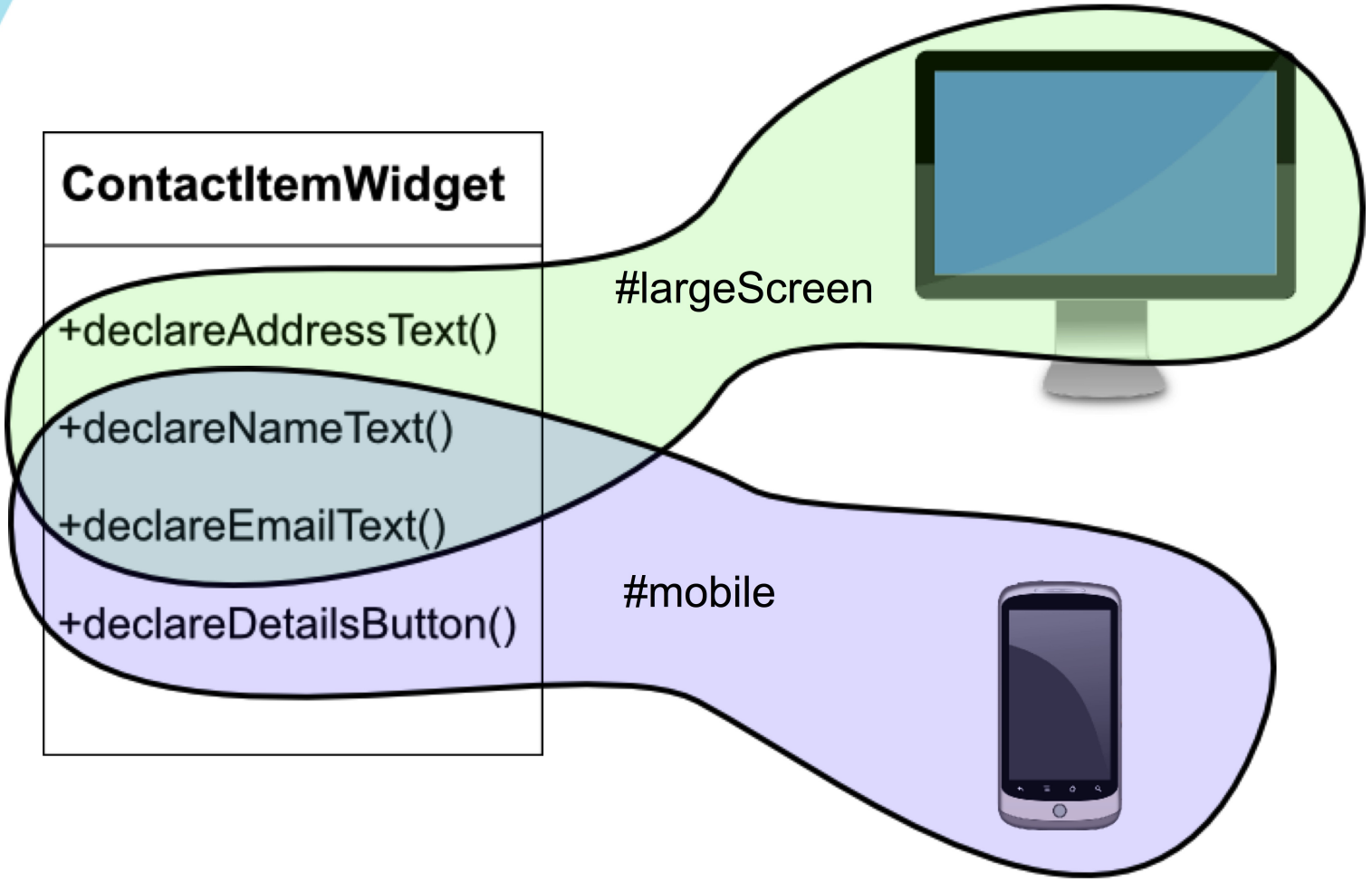
What is an object in Dali

- Made of entities
  - Properties
  - Behaviors
  - GUI
- **According to the platform**
  - **An entity may be used or not**

More visually



More visually



A focus on styles and events

- Layer of logical widgets
- No technical constraints

Styles API

- According to the W3C CSS standard

'margin-top', 'margin-bottom'

Value:	<margin-width> | inherit
Initial:	0
Applies to:	all elements except elements with table display types other than table-caption,
Inherited:	no
Percentages:	refer to width of containing block
Media:	visual
Computed value:	the percentage as specified or the absolute length

http://www.w3.org/TR/CSS2/box.html#margin-properties

Styles API

- According to the W3C CSS standard
- Implemented using Traits

```
DATWithMargin >> declareMarginTop
DATWithMargin >> declareMarginBottom
DATWithMargin >> marginTop
DATWithMargin >> marginTop:
DATWithMargin >> marginBottom
DATWithMargin >> marginBottom:
...
```

Events API

- According to the W3C DOM Events standard

`click`

Type	`click`
Interface	`MouseEvent`
Sync / Async	Sync
Bubbles	Yes
Target	`Element`
Cancelable	Yes

http://www.w3.org/TR/DOM-Level-3-Events/#event-type-click

Events API

- According to the W3C CSS standard
- Implemented using class hierarchy and Traits

```
DATWithOnClickEvent >> dispatchClickEvent
DATWithOnClickEvent >> onClick:
...
```

Example

```
DAWidget subclass: #MyWidget
        uses: DATWithOnClickEvent + DATWithMargin
        instanceVariableNames: ''
        classVariableNames: ''
        category: 'Dali-Widget-Example'
```

In a nutshell

- Single development environment
- Single application model
- Agile approach
- Widget APIs according to well known standards

Future Work

- Whole application generation
- Aspect oriented mechanism in addition to the use of pragmas
- Slot perspectives

Thank you !