# Constraining application behaviour by generating languages

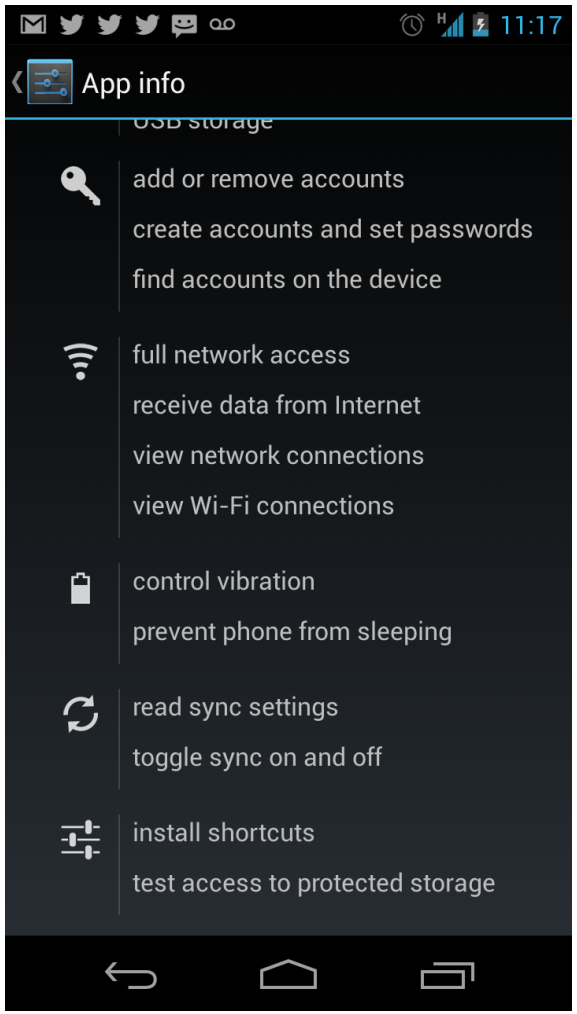
ELS 2015

Paul van der Walt
paul.vanderwalt@inria.fr

INRIA Bordeaux

20 April, 2015

# What does this mobile app do?



# On Android permissions

- Android has a notion of "permissions", but
  - Permissions apply to entire app
  - All-or-nothing for the user
- To curb privacy breaches, like
  - Advertising libraries regularly exfiltrate data,
  - Twitter, LinkedIn apps stealing contact list,
  - *etc.*

# On Android permissions

- Android has a notion of "permissions", but
  - Permissions apply to entire app
  - All-or-nothing for the user
- To curb privacy breaches, like
  - Advertising libraries regularly exfiltrate data,
  - Twitter, LinkedIn apps stealing contact list,
  - *etc.*
- We can do better :)

# Running example: EvilCam!

# Running example: EvilCam!

Supposedly:

- Takes a picture
- Applies sepia filter
- Displays it to user

# Running example: EvilCam!

Supposedly:

- Takes a picture
- Applies sepia filter
- Displays it to user
- . . . and shows an advert

# Running example: EvilCam!

Supposedly:

- Takes a picture
  $\rightarrow$ camera permission
- Applies sepia filter
- Displays it to user
- . . . and shows an advert

# Running example: EvilCam!

Supposedly:

- Takes a picture
  → camera permission
- Applies sepia filter
- Displays it to user
- . . . and shows an advert
  → network permission

# Potential data flow

What you hope:

- camera $\rightarrow$ your screen
- internet $\rightarrow$ fetch advert
- nothing more.

# Potential data flow

What you hope:

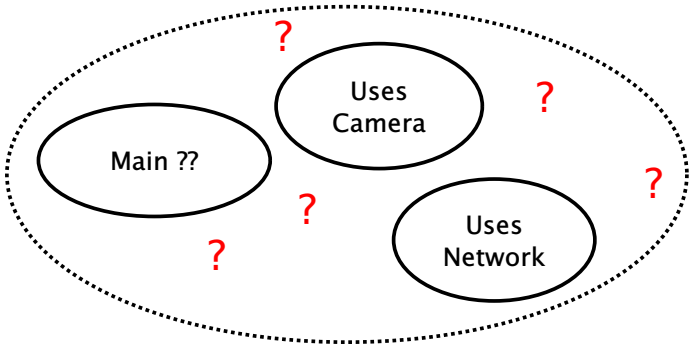
- camera → your screen
- internet → fetch advert
- nothing more.

Reality:

- image → `stalkme.net` and `nsa.gov`

# How can we curb this?

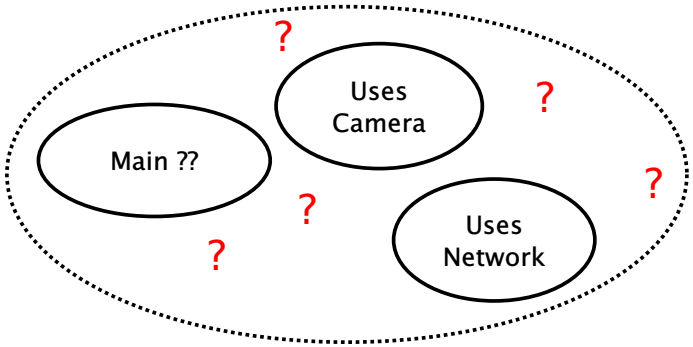
The Android permission model:




Even with conservative permissions, behaviour is unpredictable.

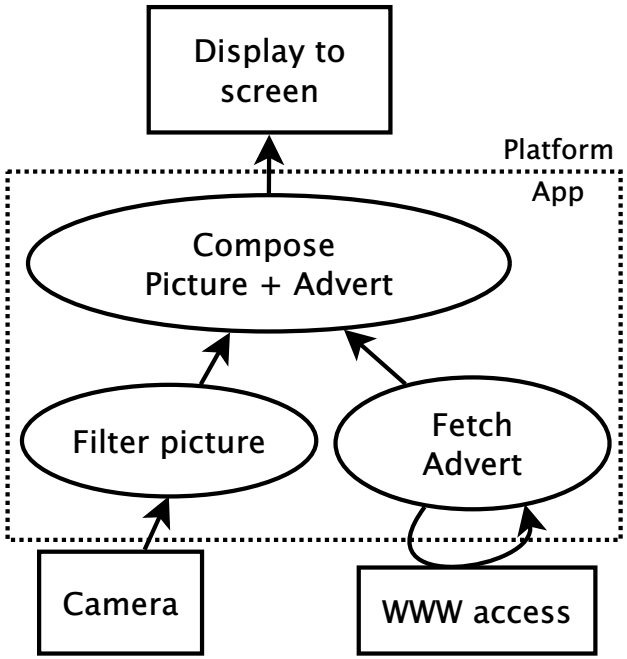
# How can we curb this?

The Android permission model:




Even with conservative permissions, behaviour is unpredictable.

A better way of doing it:



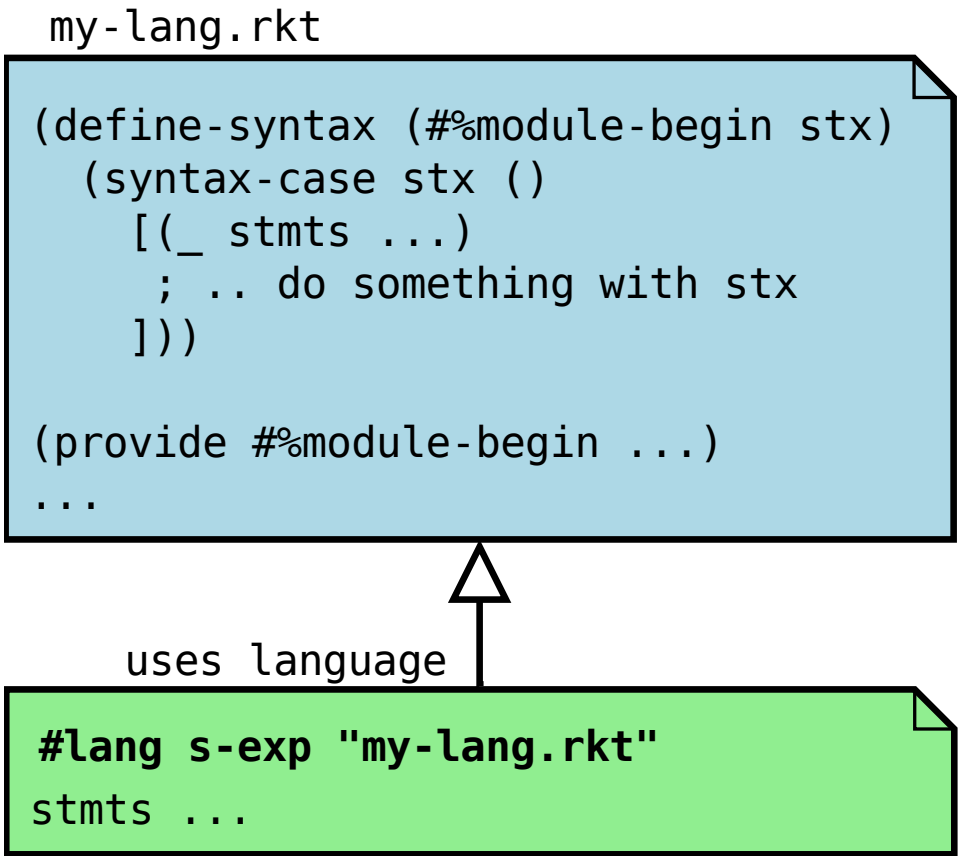
# Our goals

- ✓ Give the user clarity via spec
- ✓ Force the app to conform to spec
- ✓ Guide the developer where possible

# The proposed approach

- Provide a DSL to write up spec
  (encoding of flow diagram shown before)
- Provide another DSL based on that, to implement app
- *I.e.*, tower of languages

# Racket and **#lang**



# `#lang` providing `#lang`

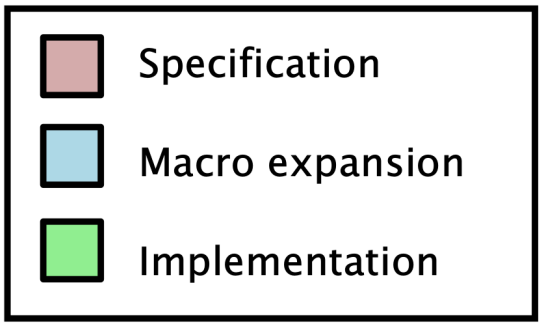
Relation between specifications and implementation:

# **#lang** providing **#lang**

Relation between specifications and implementation:

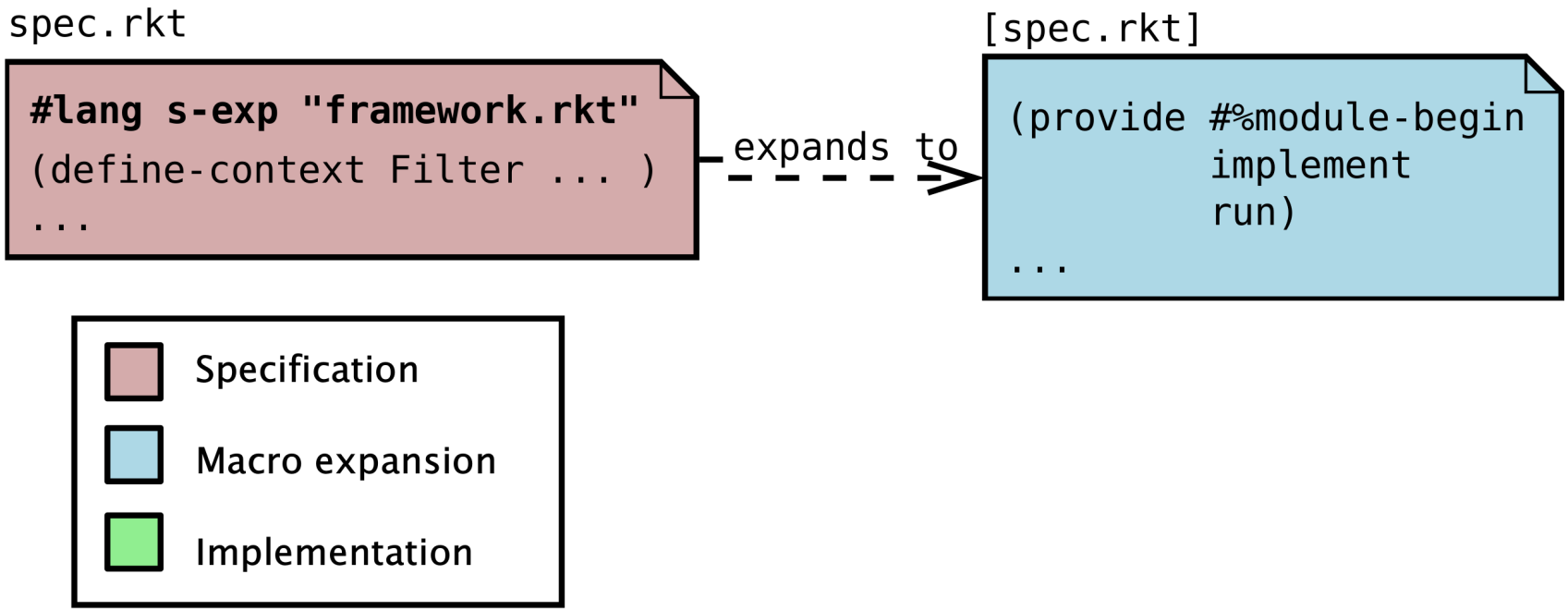
spec.rkt

```
#lang s-exp "framework.rkt"
(define-context Filter ... )
...
```



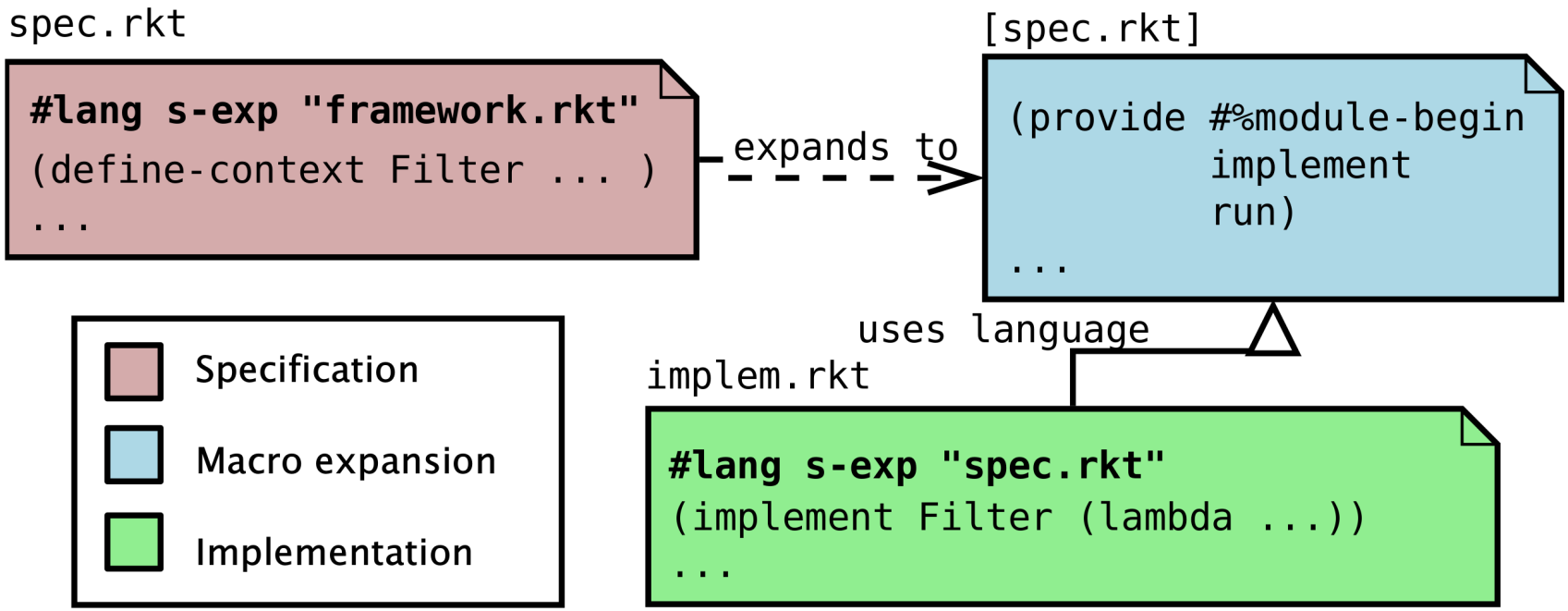
# `#lang` providing `#lang`

Relation between specifications and implementation:



# #lang providing #lang

Relation between specifications and implementation:



# Specifications

```
1 #lang s-exp "framework.rkt"
2 ;;; Specifications file, webcamspec.rkt
```

# Specifications

```
1 #lang s-exp "framework.rkt"
2 ;;; Specifications file, webcamspec.rkt
3 (define-context Filter                      ; name
4                 Picture                     ; return type
5                 [when-provided Camera]) ; subscribed to
6
7 (define-source  Camera Picture) ; built-in
8 ;; ...
```

# Specifications

```
1 #lang s-exp "framework.rkt"
2 ;;; Specifications file, webcamspec.rkt
3 (define-context Filter                    ; name
4                 Picture                   ; return type
5                 [when-provided Camera]) ; subscribed to
6
7 (define-source  Camera Picture) ; built-in
8 ;; ...
```

The types allow us to generate function contracts.

# Implementation

The developer does the following:

```
1 ;;; Implementation file, webcamimpl.rkt
2 #lang s-exp "webcamspec.rkt"
```

# Implementation

The developer does the following:

```
1 ;;; Implementation file, webcamimpl.rkt
2 #lang s-exp "webcamspec.rkt"
3 (implement Filter
```

# Implementation

The developer does the following:

```
1 ;;; Implementation file, webcamimpl.rkt
2 #lang s-exp "webcamspec.rkt"
3 (implement Filter
4   (lambda (pic)
```

# Implementation

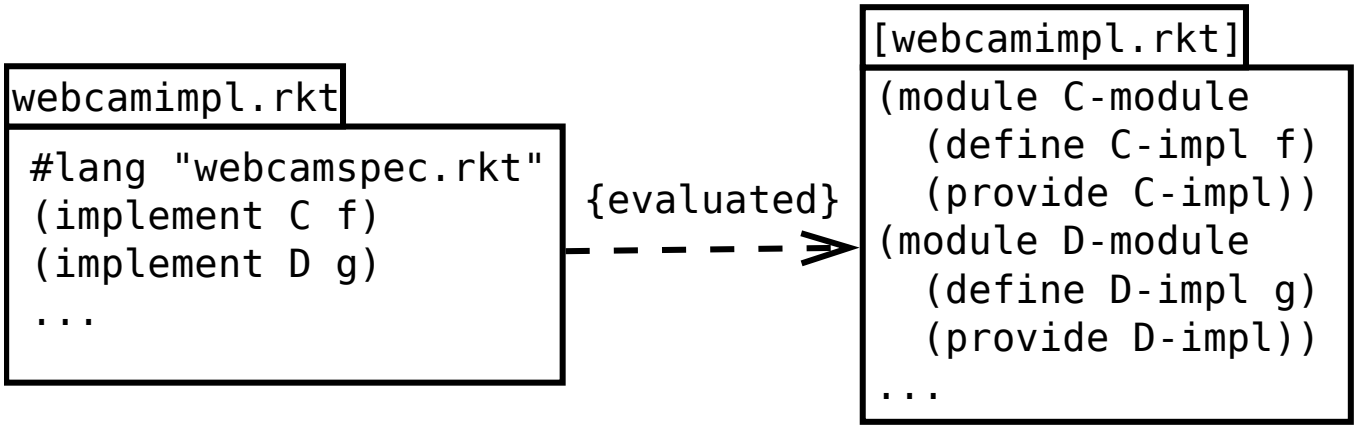
The developer does the following:

```
1 ;;; Implementation file, webcamimpl.rkt
2 #lang s-exp "webcamspec.rkt"
3 (implement Filter
4   (lambda (pic)
5     (let* ([canvas  (make-bitmap pic ..)])
6       ; ... do magic, change colours or whatever
7       canvas)))
8 ;; ...
```

...which will be turned into submodules.

# Separation into submodules

Compartmentalise snippets to enforce safety: $C$ and $D$ cannot communicate.



# Implementation

So, `implement` expands to:

```
1 (module webcamimpl "webcamspec.rkt"
```

# Implementation

So, `implement` expands to:

```
1 (module webcamimpl "webcamspec.rkt"
2   (module Filter-module racket/gui
```

# Implementation

So, `implement` expands to:

```
1 (module webcamimpl "webcamspec.rkt"
2   (module Filter-module racket/gui
3     (define/contract Filter-impl
4       (-> bitmap%? bitmap%?)
```

# Implementation

So, `implement` expands to:

```
1 (module webcamimpl "webcamspec.rkt"
2   (module Filter-module racket/gui
3     (define/contract Filter-impl
4       (-> bitmap%? bitmap%?)
5           ;; the lambda from the previous step
6     )
7     (provide Filter-impl))
8   ...)
```

# Implementation

So, `implement` expands to:

```
1 (module webcamimpl "webcamspec.rkt"
2   (module Filter-module racket/gui
3     (define/contract Filter-impl
4       (-> bitmap%? bitmap%?)
5           ;; the lambda from the previous step
6     )
7     (provide Filter-impl))
8   ...)
```

`webcamspec` also

- checks that all `defines` have `implements`
- and provides `run`

# Revisiting the goals

- ✓ The specification gives the user an idea of data flow

# Revisiting the goals

- ✓ The specification gives the user an idea of data flow
- ✓ Submodules ensure no unwanted communication

# Revisiting the goals

- ✓ The specification gives the user an idea of data flow
- ✓ Submodules ensure no unwanted communication
- ✓ Function contracts give hints to the developer

# Revisiting the goals

- ✓ The specification gives the user an idea of data flow
- ✓ Submodules ensure no unwanted communication
- ✓ Function contracts give hints to the developer
- ✓ The developer is warned if implementation doesn't match spec

# Want a demo?

– code available on my home page –

http://people.bordeaux.inria.fr/pwalt

# Conclusion

- DSL for tailoring DSLs? :)
- More confidence for user of app
- Relatively nice interface for app developer
- Not 100% practical or watertight – just a proof of concept
  - Use of libraries is tricky
  - Use of `eval` is not permitted
- Would be a great improvement on current mobile permissions systems