Yet Another DSL for Cross-platforms Mobile Development

GlobalDSL 2013 # *Session 2*
Montpellier, 2013

Olivier Le Goaer and Sacha Waltham
olivier.legoaer@univ-pau.fr

Agenda

- Context of work
- A survey of existing tools
- Presentation of the *X*$_{MOB}$ solution
- Questions?

Context of work

Development shift

- The new era of mobility
  - People massively use apps on handheld devices (SmartPhones, Tablets)
- From Desktop application to mobile applications
  - Lower ressources (battery, network latency, processor, etc.)
  - Smaller screen and new navigation fashion (“Tap-able” not “clickable”)
- Increased platform heterogeneity
  - Now steady, limited number of desktop OS
  - Yet unsteady, high number of mobile OS

Heterogeneity is back !

« Desktop App »
Development

$$

Windows

Mac OS X

Linux

« Mobile App »
Development

$$$$$, augmented
Time-to-Market

iOS

android

Windows 8 Phone

bada

Firefox OS

BlackBerry

MeeGo

symbian OS

ubuntu

brew

Mobile app: native or web?

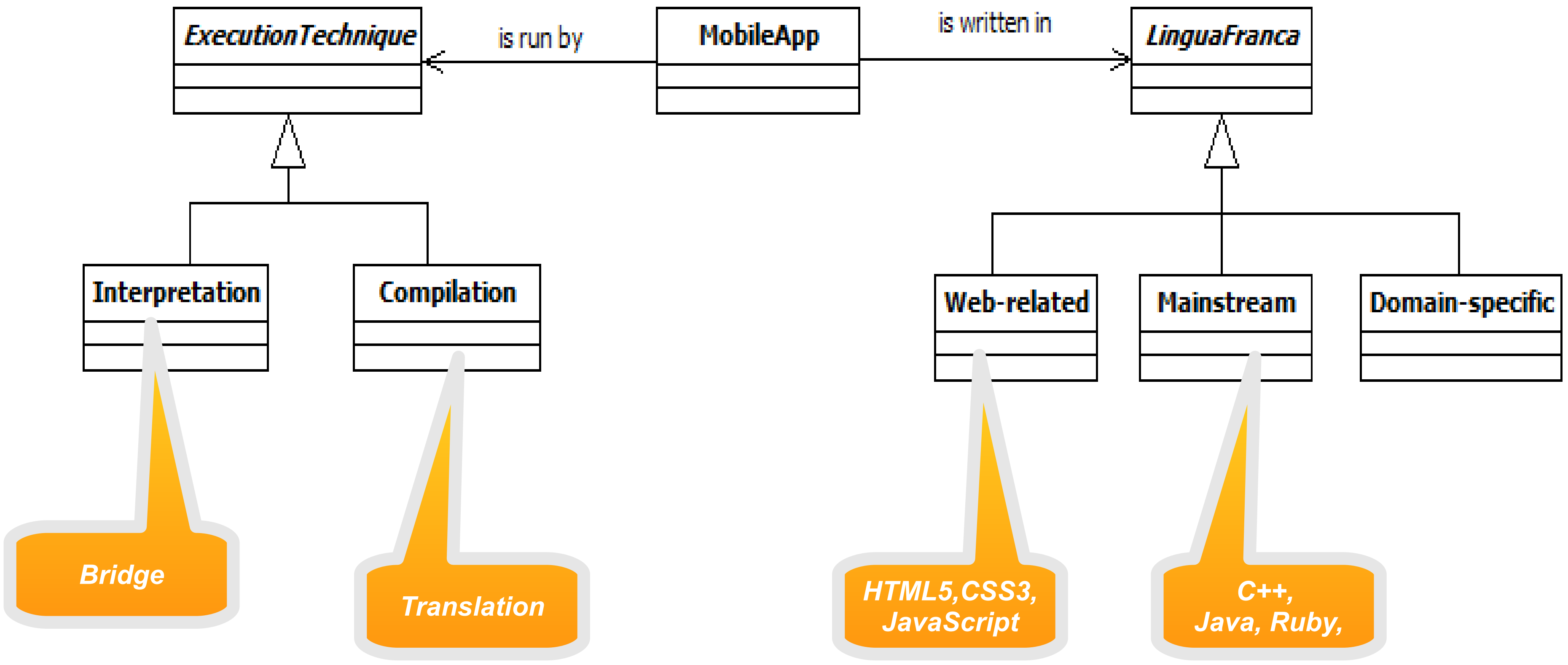
- Two development approaches are competing
  - Native: develop directly for the mobile device
  - Web: develop for a browser installed in the mobile device, and tailored to be « mobile-friendly »
- Tame the development costs...
  - Native: one app per platform 😣
  - Web: solely one app 🙂
    (assuming that nowadays the different browsers evenly process the code)
- ...but do not neglect the « user experience »
  - Native: almost limitless capabilities 🙂
  - Web: limited to the browser capabilities 😕

In a nutshell

- WebApps are the best way to reach the most possible people with the least effort
  - WebApps are inherently cross-platforms but in the agnostic sense (the underlying OS specificities are ignored)
  - The browser already partly solved the heterogeneity question
- NativeApps are the best way to create the best possible experience
  - Native apps are specifically designed for their host, and hence require further solutions to achieve actual cross-platform
  - Cross-platform Mobile development Tools (XMTs) have emerged
  - **This is where the scientific challenge lies**

A survey of existing tools

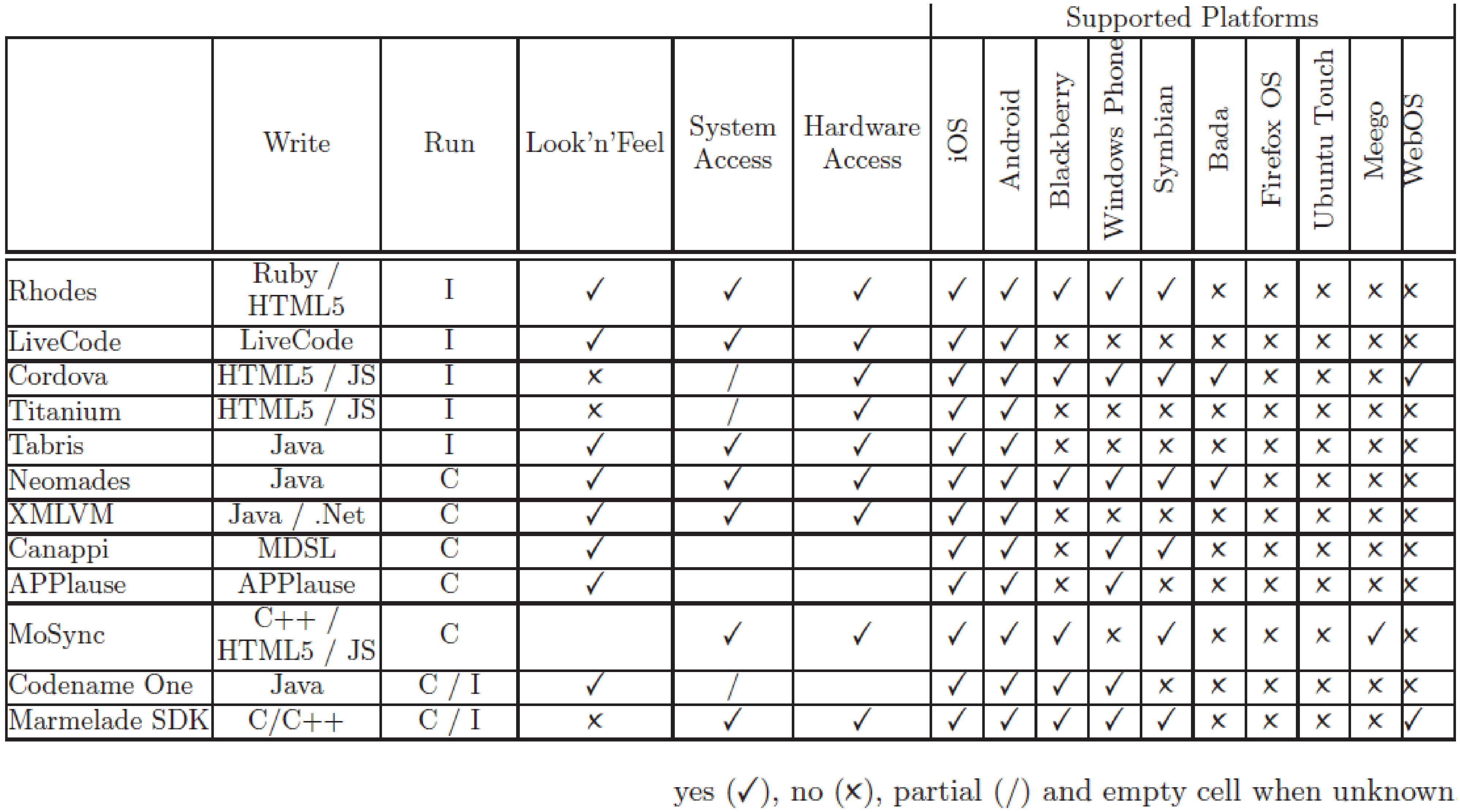
Write Once? Run everywhere?



Reappearance of MDA

- Such heterogeneity and lack of sustainability was already encountered in the past
  - The mid-2000s was the age of « middle**war** »
  - The OMG brought its Model-Driven Architecture (MDA) vision
- When MDA meets the mobile challenge
  - Capture knowledge into models
    - *Describe things independently of mobile platforms (PIM level)*
    - *Platform details are woven subsequently (PSM level)*
  - OMG's contributions (UML2, MOF, QVT) are centre-stage, but domain-specific languages are encouraged
    - *A langage dedicated to the mobile domain rather than a general-purpose language*
  - Offers a basis for further stuff (tests, simulation and analysis, ...)

Overview of existing XMTs



						Supported Platforms									
	Write	Run	Look'n'Feel	System Access	Hardware Access	iOS	Android	Blackberry	Windows Phone	Symbian	Bada	Firefox OS	Ubuntu Touch	Meego	WebOS
Rhodes	Ruby / HTML5	I	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	×
LiveCode	LiveCode	I	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
Cordova	HTML5 / JS	I	×	/	✓	✓	✓	✓	✓	✓	✓	×	×	×	✓
Titanium	HTML5 / JS	I	×	/	✓	✓	✓	×	×	×	×	×	×	×	×
Tabris	Java	I	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
Neomades	Java	C	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×
XMLVM	Java / .Net	C	✓	✓	✓	✓	✓	×	×	×	×	×	×	×	×
Canappi	MDSL	C	✓			✓	✓	×	✓	✓	×	×	×	×	×
APPlause	APPlause	C	✓			✓	✓	×	✓	×	×	×	×	×	×
MoSync	C++ / HTML5 / JS	C		✓	✓	✓	✓	✓	×	✓	×	×	×	✓	×
Codename One	Java	C / I	✓	/		✓	✓	✓	✓	×	×	×	×	×	×
Marmelade SDK	C/C++	C / I	×	✓	✓	✓	✓	✓	✓	✓	×	×	×	×	✓

yes (✓), no (×), partial (/) and empty cell when unknown

Rationales of the X_{MOB} solution

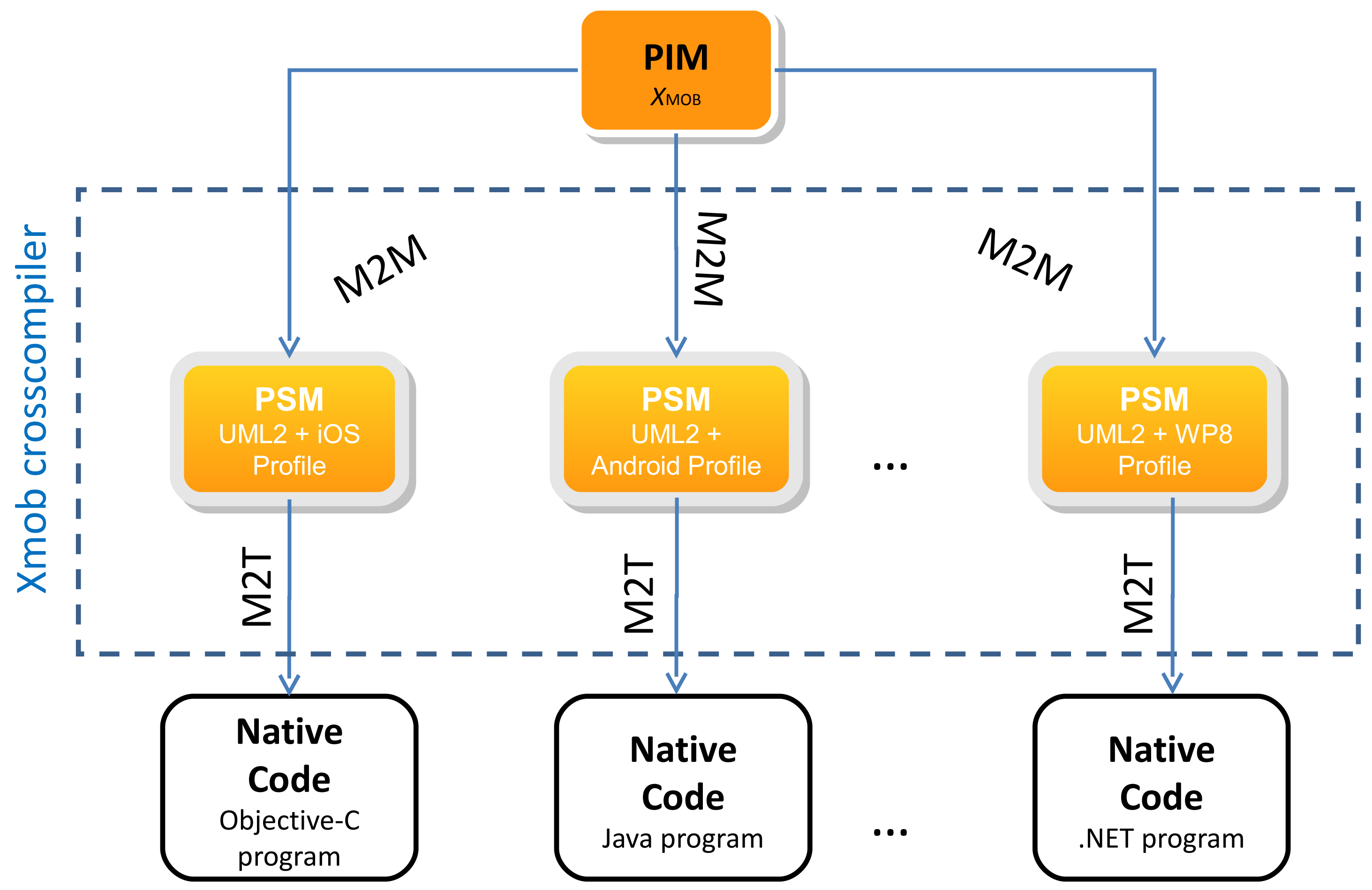
- Domain-specific language
  - Define a language which really fits your needs
  - Decrease the efforts needed to write a (simple) mob app
    - *Obviously far away to be as complete as a general language*
    - *Key idea : « Write less, generate more »*
- Generation of full-native code
  - Because this is the holy grail of mobile programming
- Model-driven Architecture
  - Proved that « it works ». Separation PIM/PSM is useful.
- Reinventing the wheel?
  - Several languages may be competing. Let us try...

Presentation of the *X*MOB solution

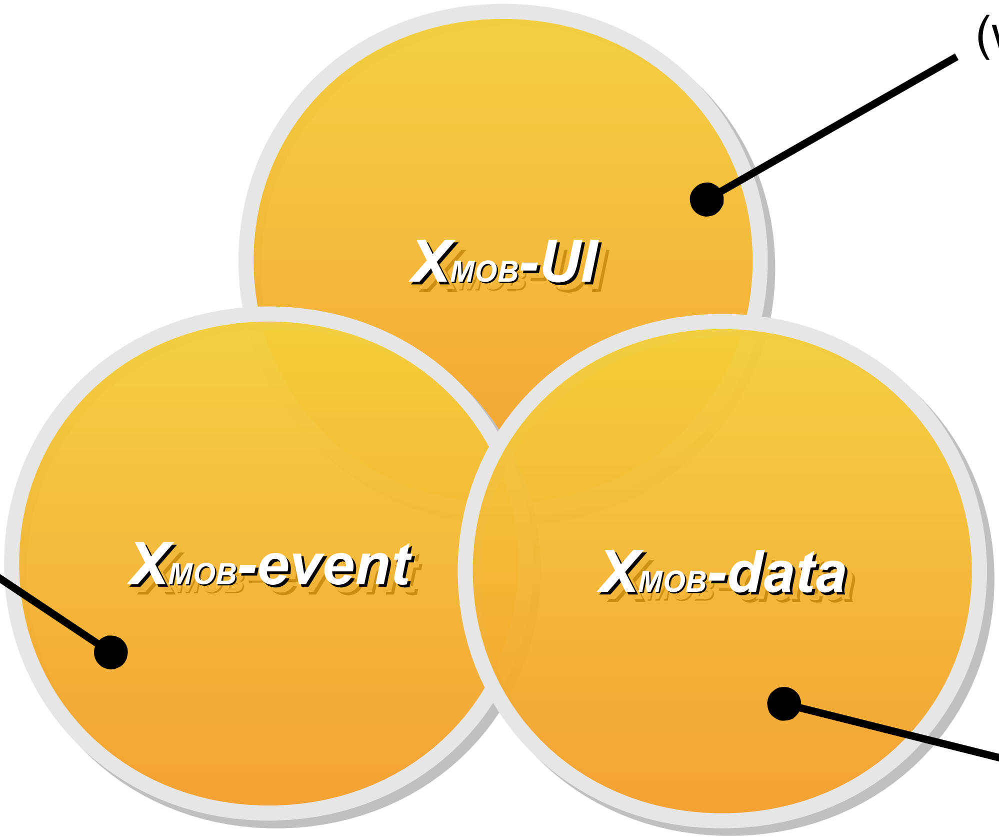
XMOB roadmap

- Set the architecture of the XMOB crosscompiler
  - Envision a MDA-compliant chain to produce native code
- Design the XMOB language
  - Define both its abstract and concrete syntax
- Create UML profiles for each platform
  - Write the associated transformation (M2M and M2T)
  - Android initially, then move onto other platforms
- Deliver the XMOB solution as an Eclipse Plugin
  - Built on top of EMF
  - But the generated code ought to be reworked into specific IDE

XMOB Cross-compiler Architecture



XMOB sub-languages



Deals with display (widgets)

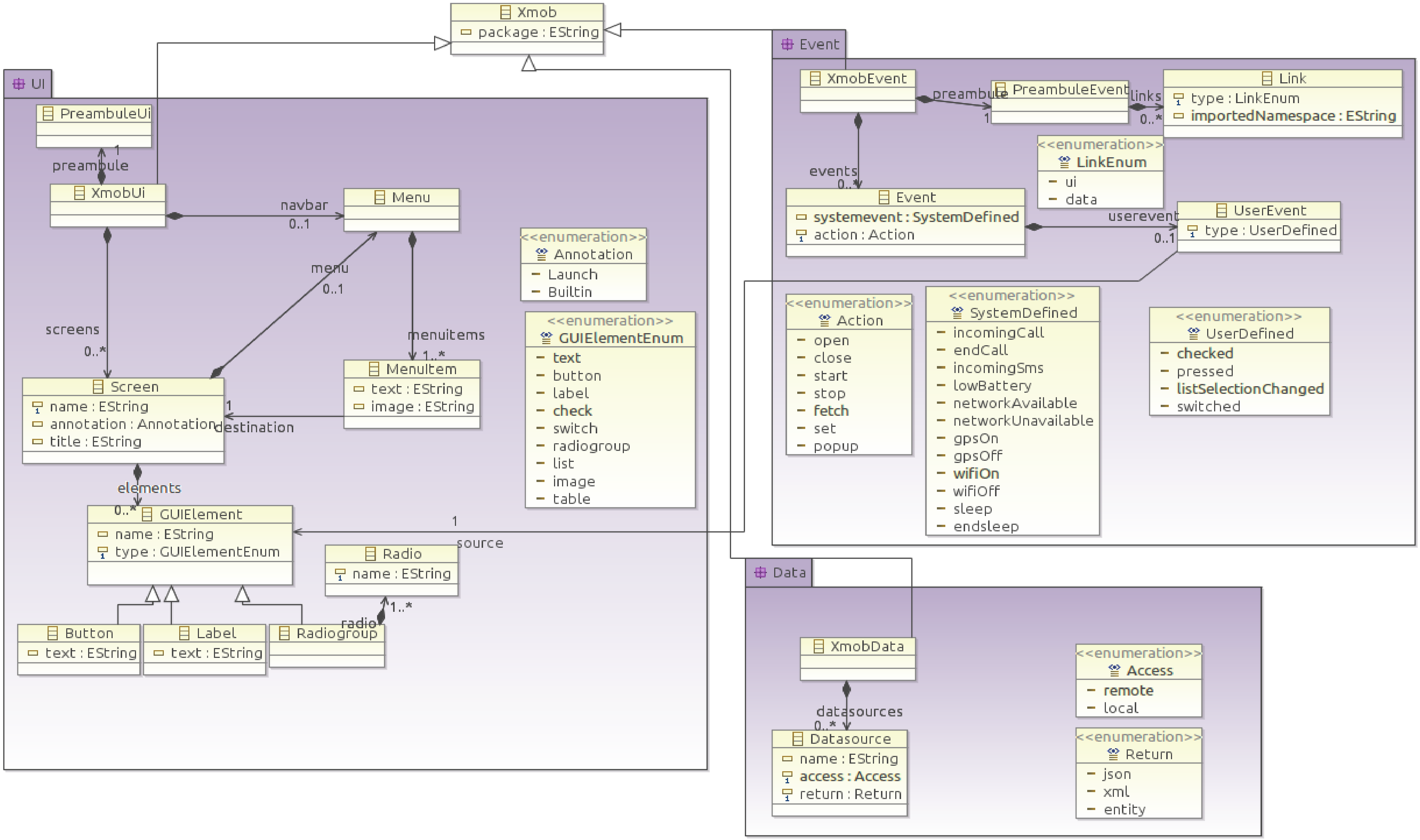
Deals with gluing together UI and data

Deals with data sources (read only)

Mobile-specific shared concepts

- X_{MOB}-UI
  - UI is broken down into a succession of screens
  - UI elements (widgets) are declared inside a screen
  - UI elements will automatically be placed on the screen
- X_{MOB}-data
  - Datasources location: local or remote
  - Datasources format: xml, json, recordset, raw text, ...
- X_{MOB}-event
  - Triggers actions on System-related or UI-related events
  - Actions are based on verbs : open, close, fetch, start, ...

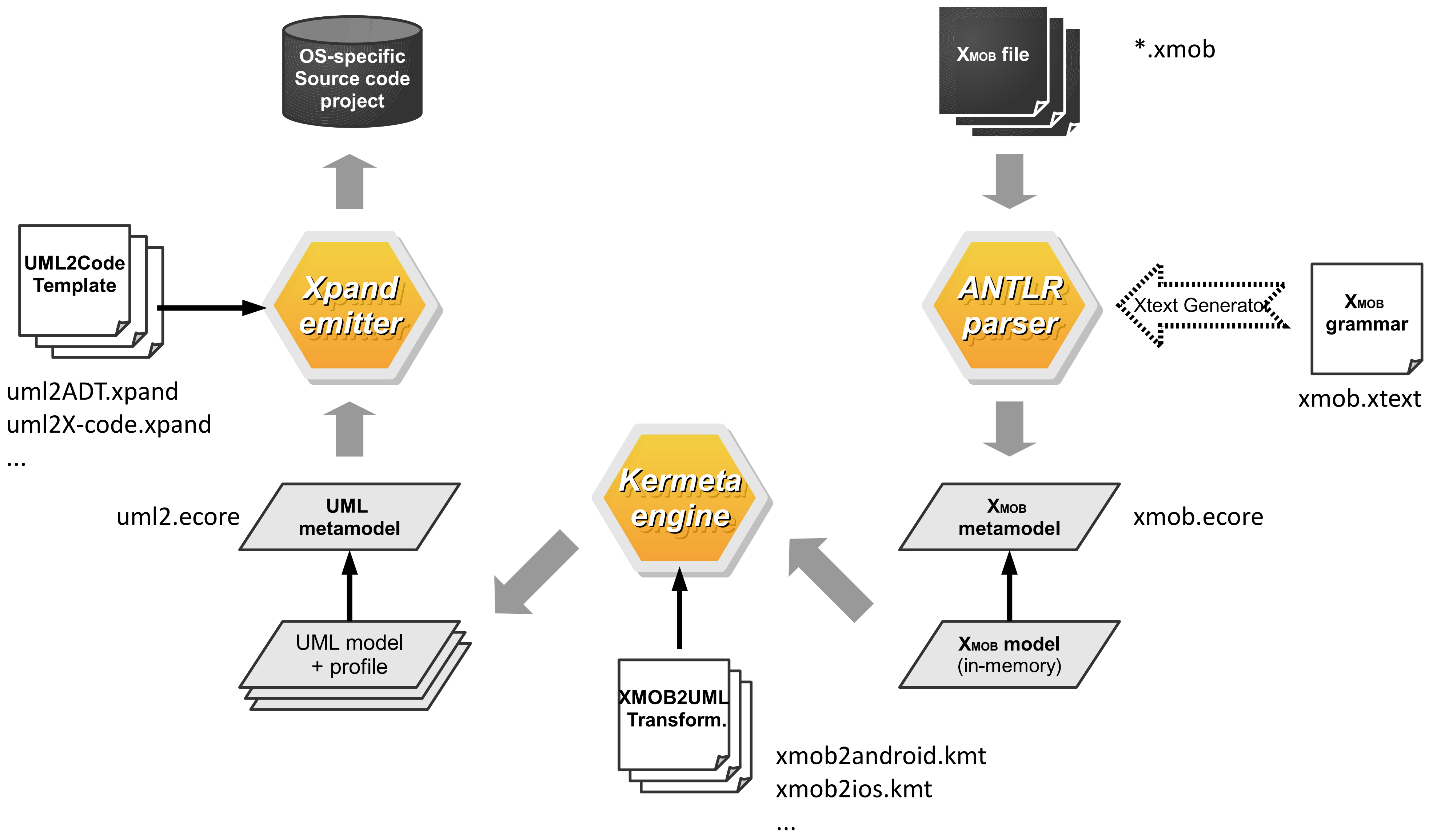
XMOB abstract syntax (Ecore)



XMOB concrete syntax (Xtext)

- XMOB has a textual syntax
  - Better suited than graphical syntax, especially for data and events
- XMOB has multiple files
  - Separated ui/event/data files but same extension « .xmob »
  - Implies to manage properly cross-references and scoping
- No Type System nor Function definition
  - Because XMOB is much more a description language than a programming language
  - Because this is a tedious task: code for type-checking and calls must be added programmatically to the parser

EMF Tooling for the XMOB solution



XMOB snippets

```
#xmob-ui

@Launch
screen main {
        l_welcome as label["Welcome to this app"]
        b_next as button["Proceed to next screen"]
        menu {
                item[image: "settings.png", destination:settings]
                item[text:"Credits", destination:credits]
        }
}

screen next {
        label["Here is some data:"]
        list_somedata as list
}

screen credits {
        label["Contributors:"]
        label["Olivier le Goaer & Sacha Waltham"]
}

screen settings {
        label["music ?"]
        switch_music as switch
}
```

```
#xmob-data

somedata as datasource {
        remote ["http://somewebsite.com/someservice?param=1"]
        return [xml]
}
```

```
#xmob-event

on (pressed[main.b_next]) do { open[next] }

on (networkAvailable[SYSTEM]) do {
        fetch[somedata] into[next.list_somedata]
}

on (switched[settings.switch_music]) do {
        start[player:./music.mp3]
}
```

XMOB snippets

```
#xmob-ui

@Launch
screen main {
        l_welcome as label["Welcome to this app"]
        b_next as button["Proceed to next screen"]
        menu {
                item[image: "settings.png", destination:settings]
                item[text:"Credits", destination:credits]
        }
}

screen next {
        label["Here is some data:"]
        list_somedata as list
}

screen credits {
        label["Contributors:"]
        label["Olivier le Goaer & Sacha Waltham"]
}

screen settings {
        label["music ?"]
        switch_music as switch
}
```

```
#xmob-data

somedata as datasource {
        remote ["http://somewebsite.com/someservice?param=1"]
        return [xml]
}
```

```
#xmob-event

on (pressed[main.b_next]) do { open[next] }

on (networkAvailable[SYSTEM]) do {
        fetch[somedata] into[next.list_somedata]
}

on (switched[settings.switch_music]) do {
        start[player:./music.mp3]
}
```

Android

- BroadcastReceiver
- AsyncTask
- Progress Dialog

- SAX Parser
- ArrayAdapter
- ...

Questions?