



MobiCloud

Power of Clouds in Your Pocket : An Efficient
Approach for Cloud Mobile Hybrid Application
Development

**Ashwin Manjunatha, Ajith H. Ranabahu , Amit Sheth,
Krishnaprasad Thirunarayan**

**Kno.e.sis Center
Wright State University**

Agenda

- What are Cloud-Mobile hybrids ?
 - Why are they hard to build ?
- Building Cloud-Mobile hybrids with DSLs
 - Pros and cons
 - Current DSL
- MobiCloud Toolkit
 - Ongoing work
 - Future work
- Questions





What are Cloud-Mobile Hybrids ?

Spectrum of Computing Power



Google
App Engine

amazon
web services™

Salesforce

Windows Azure



NOKIA
Connecting People

BlackBerry



Portability



Computing Power



Scalable
On-demand
Economical

Portable
Ubiquitous
Connected

Cloud-Mobile Hybrids (CMH) ?



- Applications that span over multiple devices
 - Back-end in Cloud
 - Front-end in a mobile device
- Both components needed for the app to function
- Front-end is not just a webpage !
 - A native Iphone / Android / Blackberry app

Simple Example



- The Facebook App
 - The Facebook client you have in your smart phone !
 - Mobile front-end for Facebook
 - Most Facebook activity happens via Mobile devices !
 - Many actions need extensive processing in the Facebook backend
 - Both parts (back-end and the front-end) required for the complete experience

Another Research Oriented Example



- Privacy score¹
 - Introduced by IBM researchers
 - Measures relative exposure of private data on a social network
 - Provides a similar number to “credit score”
- Requires heavy calculations in the back-end
- Front-end is simply one number !
 - Perfect for a mobile device !

1. A Framework for Computing the Privacy Scores of Users in Online Social Networks,

Kun Liu, [Evimaria Terzi](#)

Ninth IEEE International Conference on Data Mining, 2009

Are CMH Apps hard to build ?



- Yes - very much !!
- Clouds are heterogeneous
 - Write the back-end to suit Amazon – You can't move to Google !!
- Same in mobile devices
 - Need to write different apps for different devices !
 - Android / Iphone / Blackberry are all different



Our solution to this Problem...

Use a DSL



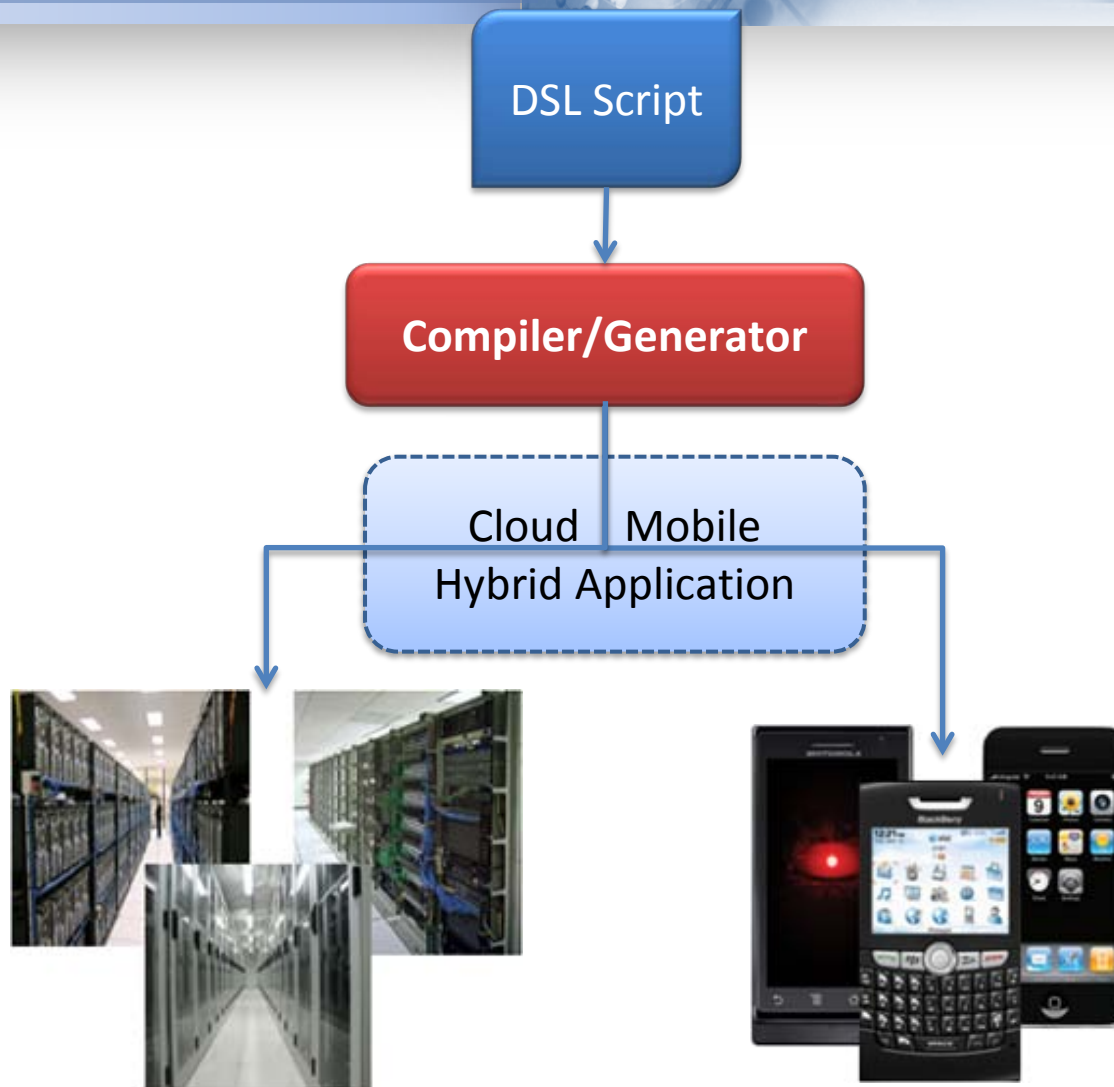
- DSL ?
 - **Domain Specific Language**
 - A mini language for a special purpose
 - Make / Ant
 - Matlab
 - Many other examples
- Use a DSL to ***generate*** an application

What changes by using a DSL ?



- Reduce Complexity
 - As DSLs are designed for specific domain, they reduce complexity!
- Increases Flexibility and avoids vendor lock-in
 - Use single DSL to design applications with different combinations of Cloud and mobile platforms
 - Generate apps for Amazon, Google, Android, Blackberry using just a single script
- Ease of use
 - No separation between front-end / back-end
 - No service interfaces to worry about (auto generated)
 - Communication interfaces are a major source of errors and incompatibilities

A birds-eye view of what we do



Is this the silver bullet ?



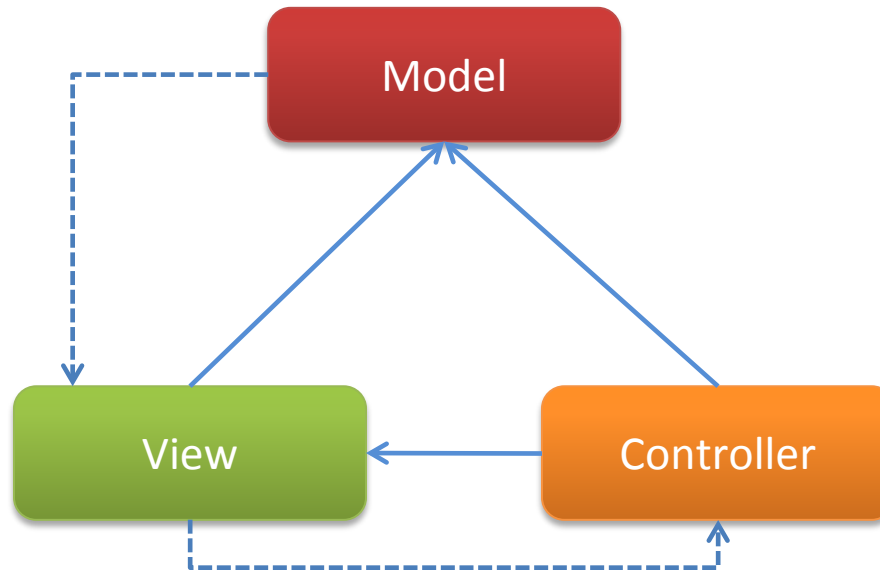
- Nope !
 - Less control over the code
 - E.g. Extensive GUI customization and device integration not possible
 - Covers only the 80% case
 - E.g. Not suitable for games or other UI intensive applications



Our current DSL

DSL design principle

- Based on Model-View-Controller (MVC) design pattern



A very simple “Hello World”



```
recipe :helloworld do
```

```
  metadata :id => 'helloworld-app'
```

```
  # models
```

```
  model :greeting, {:message => :string}
```

```
  #controllers
```

```
    controller :sayhello do
```

```
      action :retrieve, :greeting
```

```
    end
```

```
  # views
```

```
    view :show_greeting,
```

```
      {:models => [:greeting],
```

```
       :controller => :sayhello, :action => :retrieve}
```

```
  end
```

Metadata – details that need to be attached to the whole application

Models

Controllers

Views

A slightly complicated Example (task manager)



```
recipe(:todolist) do

  # specific metadata for this app
  metadata({:id => 'todo-list'})

  model(:todoitem, {:name=>:string, :description => :string, :time =>
:string, :location => :string})
  model(:user, {:name=>:string, :bday => :string})

  #controllers
  controller(:todohandler) do
    action :create, :todoitem
    action :retrieve, :todoitem
    action :update, :todoitem
    action :delete, :todoitem
  end

  # views
  view :todo_add, {:models >[:todoitem], :controller => :todohandler, :action
=> :create}
  view :todo_show, {:models =>[:todoitem], :controller =>
:todohandler, :action => :retrieve}

end
```

Regular development vs DSL for CMH



Regular Development

- Developed as two applications
- Different platforms need new effort.
- Significant effort in creating code and other artifacts
- Highly customizable

DSL based Development

- Developed as a single application
- Generators create functionally equivalent applications for multiple platforms
- Minimum effort in creating all required artifacts
- Limited customization

Other Benefits of DSL Based Development



- Convenient integration of other features
 - Location based services
 - Integration with available location sensors, e.g GPS
 - Non functional features
 - Security
 - Social Network features
 - Publish to Facebook, Twitter etc

MobiCloud Online Toolkit – Step1



MobiCloud

DSL based Cloud-Mobile hybrid Applications
part of the **cirroculmus** project

[Home](#)[About](#)[Tutorial](#)[Resources](#)[Team](#)

Create Your Application

Describe your application using our simple Domain Specific Language (DSL). You can modify one of the existing sample applications or just start from scratch. Don't forget to take a look at the tutorials.

Populate with

```
1
2 recipe(:todolist) do
3
4   # specific metadata for this app
5   metadata({:id => 'ashwin-app'})
6
7   # models
8   model(:todoitem, {:name=>:string, :description => :string, :time => :string, :location => :string})
9   model(:user, {:name=>:string, :bday => :string})
10
11  #controllers
12  controller(:todohandler) do
13    action :create, :todoitem
14    action :retrieve, :todoitem
15    action :update, :todoitem
16    action :delete, :todoitem
17  end
18 end
```

Position: Ln 1, Ch 1 Total: Ln 23, Ch 636

Services Research Lab, Metadata and Languages Lab @ Kno.e.sis Center

MobiCloud Online Toolkit – Step2



MobiCloud
DSL based Cloud-Mobile hybrid Applications
part of the **cirrocumulus** project

Home About Tutorial Resources Team

Select the Targets

- Android Android 1.5 based application
- Google Appengine A Java Google Appengine application that can be deployed via the Appengine SDK
- BlackBerry BlackBerry application compatible with all blackberry devices
- Amazon EC2 Tomcat based Java Application to host in Amazon EC2 or your local server

[Next step](#)

Services Research Lab, Metadata and Languages Lab @ Kno.e.sis Center

Credits: CSS from Free CSS Templates, Javascript based code editor from [EditArea](#)

MobiCloud Online Toolkit – Step3



MobiCloud
DSL based Cloud-Mobile hybrid Applications
part of the **cirriculumulus** project

Home About Tutorial Resources Team

Download your code
You can download the generated code and start customizing it right away or deploy the basic version.

- [todolist_blackberry.zip](#)
- [todolist_android.zip](#)
- [todolist_gae.zip](#)
- [todolist_ec2.zip](#)

Services Research Lab, Metadata and Languages Lab @ Kno.e.sis Center

Credits: CSS from Free CSS Templates, Javascript based code editor from [EditArea](#)



- Technical report on MobiCloud
 - <http://knoesis.wright.edu/library/publications/MobiCloud.pdf>
- Publicly hosted MobiCloud tool
 - <http://knoesis.org/mobicloud>
- Code repository
 - Coming soon !
- Privacy score paper
 - <http://portal.acm.org/citation.cfm?id=1674659.1677075>

Researchers



Ashwin
Manjunatha



Ajith
Ranabahu



Amit
Sheth



Krishnaprasad
Thirunarayan



Thank you



Questions ?