Lessons Learned in Building Canappi, a Model Driven Mobile Application Platform

Dr. Jean-Jacques Dubray

Bio

- Docteur-Ingénieur, University of Aix-Marseille I, Luminy
- Founder of IFE Technologies (1991-1997), NeXTStep, Mode Driven Process Control System for manufacturing advanced materials
- Member of the Research Staff at Hughes Research Laboratory (1995-1997)
- From 1997-2010, focused on SOA and BPM
  - Built various model driven Business Process Engines, B2B integration Servers, Composite Application Frameworks
  - Editor of the OASIS ebXML Business Process Spec, contributed to WS-CDL
  - Initial Co-author of SCA / SDO
- Since 2010, focusing on Mobile Solutions
  - AT&T, Service Delivery Platform Architect
  - Canappi, Founder

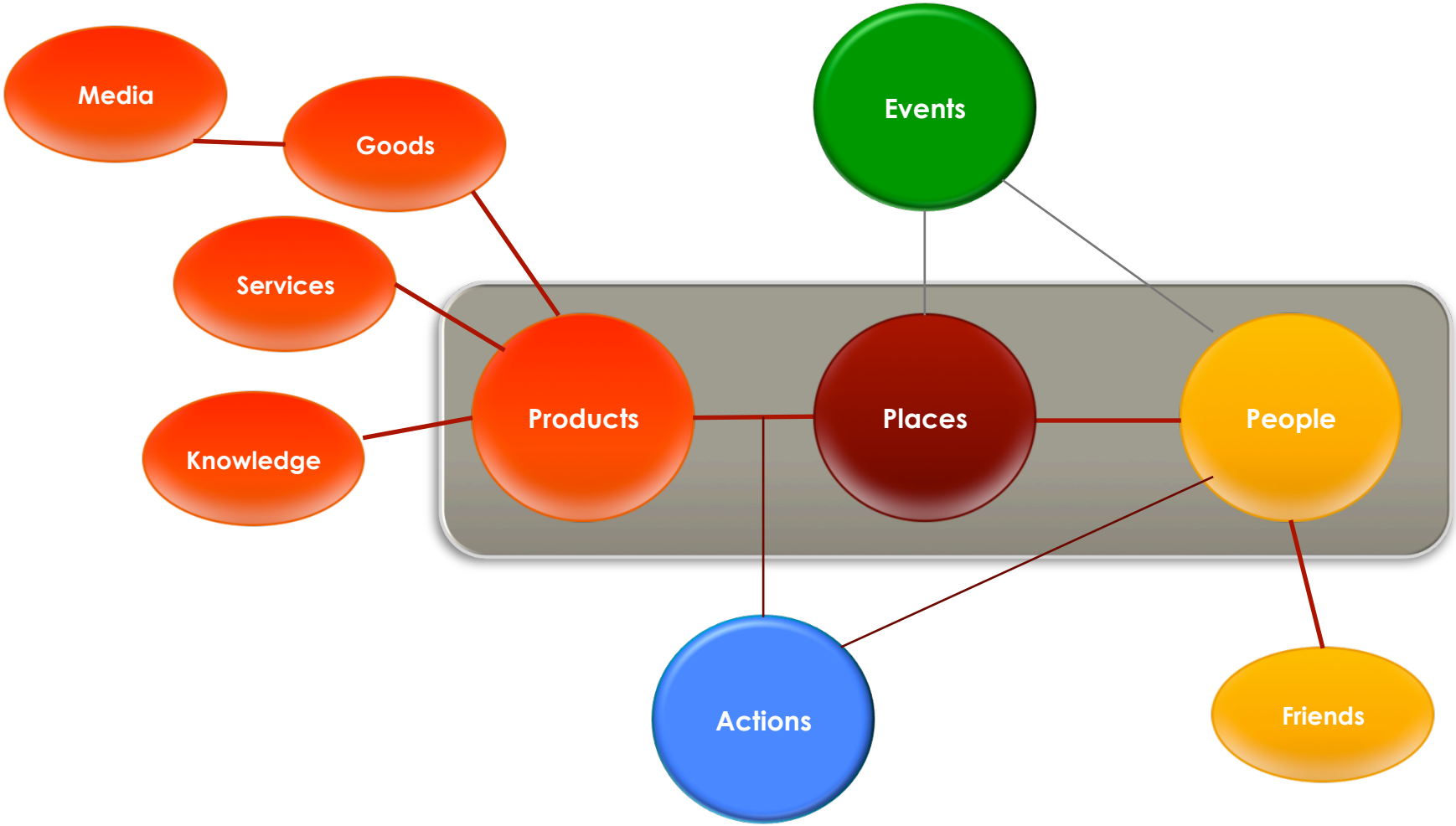
What did I learn?

- Model Driven Engineering Makes you humble
- Model Driven Engineering is not easy (by far)
- Model Driven Engineering can ~~rarely~~ cannot feed your family
- I spent 20 years building model driven systems, do I want to spend the next 20 years?
  - You bet
- Does the industry want to move to a model driven software development foundation?
  - No
  - It is all about data and scale

Outline

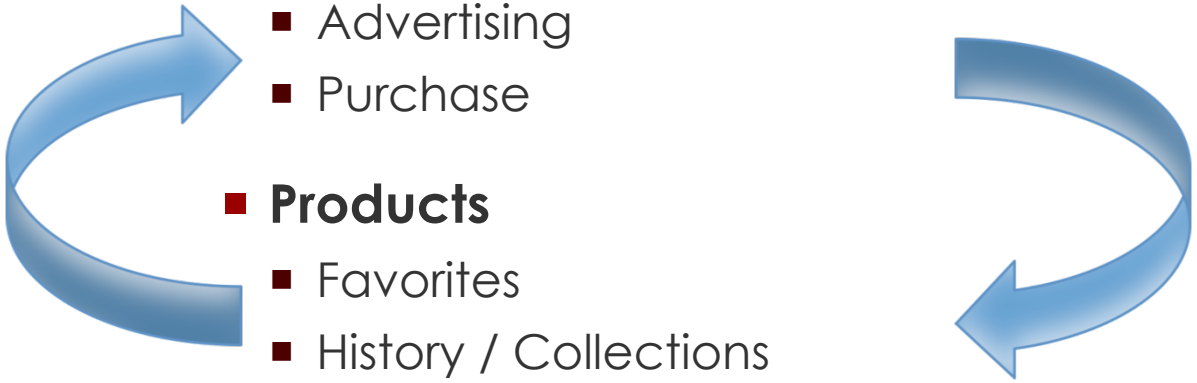
- What is different about mobile solutions?
- What challenges should you expect when trying to build a mobile solution?
- What did we build to address these challenges?
- What did we learn?
  - Technology
  - Metamodel
- Where do we go from here?

Mobility is Bringing Location Back into Focus



Mobility Enables you to Take Action !

- **Places, Friends, Events**
  - Advertising
  - Purchase
- **Products**
  - Favorites
  - History / Collections

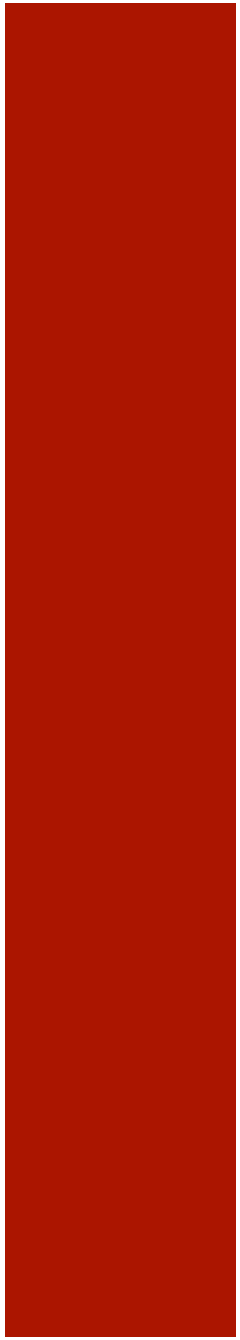


Actions

Mobility is Also About Converged Apps

- **Convergence**
  - Voice
  - Messaging
  - Location
  - Data
  - Media / IPTV
  - Social

Challenges Facing Mobile Developers



The form factor is different

Form Factor	End-User	Developer
End-users carry their clients with them	No longer need to access the same application on a number of computers (home, work, friends...).	Need to support multiple experiences for each type of mobile client (phone, tablet, laptop), in a cross platform way, which can be even more challenging than cross browser support
Apps are used in a Context	Use apps in a particular context (walking, crowds, watching TV, …)	Need to focus on the best UX for each context of utilization
Screen size	Users seek optimized UX for small screens. They want the best experience for each form factor.	Need to forget all current UX design rules and reinvent them for mobility
Power/ Bandwidth	Battery life/Bandwidth becomes a major UX element	Need to develop software with least power/ bandwidth consumption

The form factor is different (Cont'd)

Form Factor	End-User	Developer
Sensors	Expect apps take advantage of all on board sensors (camera, motion detectors, …)	Need to master specialized libraries and adapt their apps to device capabilities
Near-field communication	Bootstrap sophisticated processes which are based on user identity (payments, local actions …)	Complex back-end integration to securely share user data and manage transactions
Voice and Notifications	Expect seamless integration between apps and voice and messaging, including speech processing	Create compelling user experience and integrate notifications in application workflow

Source: "Challenges and Opportunities in Mobile Application Development" J.J. Dubray, Kurt Schmidt
http://www.infoq.com/articles/mobile-dsl

Mobile Software is End-User Centric

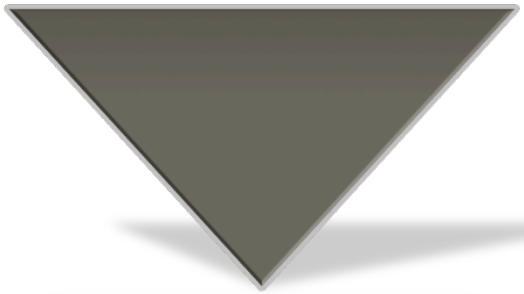
Successful Mobile Apps are Actually a Family of Variants

- Developers have many more opportunities to create apps dedicated:
  - to specific market segments
  - With seasonal features
  - For many types of devices,
  - Across several major platforms
- Across varying device form factors

- You have to be ready to deliver a family of apps, not just "one app"

Mobility is just …
the Tip of the Iceberg

Clients



Data

Desktop Companion

Too Many Moving Parts That are Difficult to Keep in Sync

yp CLICK LESS. LIVE MORE.

facebook DEVELOPERS

Zillow.com Your Edge in Real Estate

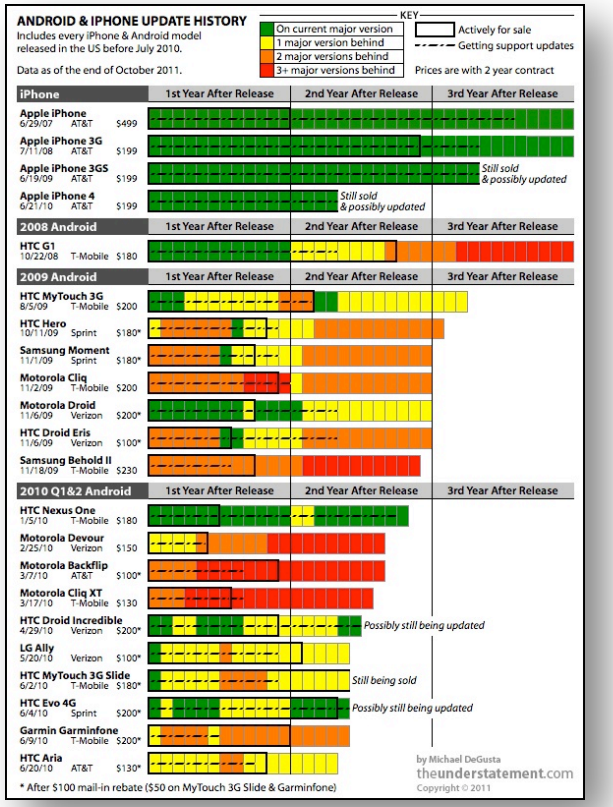
twitter

YouTube

WSO2

Mobile Apps Need Constant Refactoring

- Successful mobile applications may grow to reach tens of millions of users, in a short time frame, creating a scalability nightmare and the need to constantly re-architect the back-end ahead of the growth of the user base
- iOS SDK, Android SDK and leading Web Frameworks (Sencha, JQueryMobile…) release new major versions are a rapid rate
- On the Android side, device vendors don't bother upgrading old devices, which leads to a very unhealthy fragmentation of the market



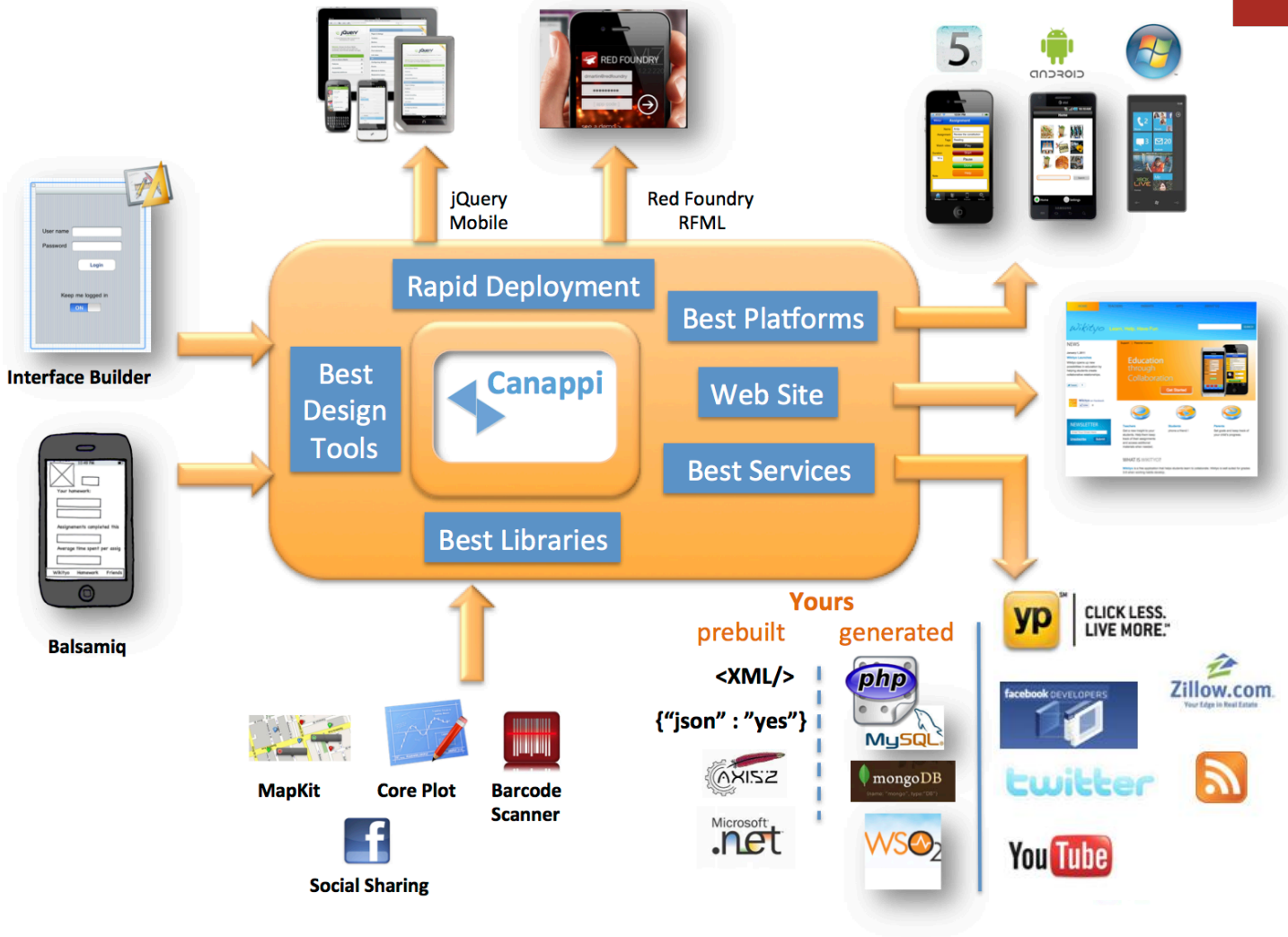
Bottom Line …

- With such short development cycles, this rapidly changing, fragmented and technologically complex new frontier is set for the perfect storm
- The seasonal aspects and the relative high degree of competition between apps put even more pressure on the development cycles,

Developing Mobile Solutions

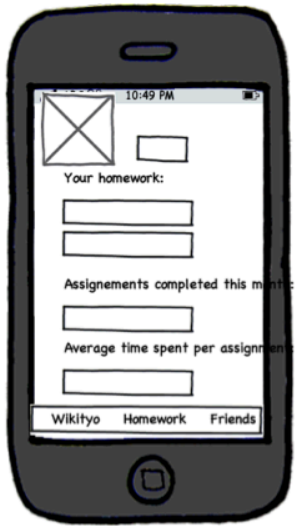
With Canappi

What does Canappi do?



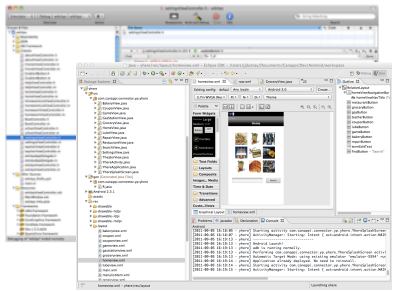
What's the process?

Design → Solution → Finalize → Deploy



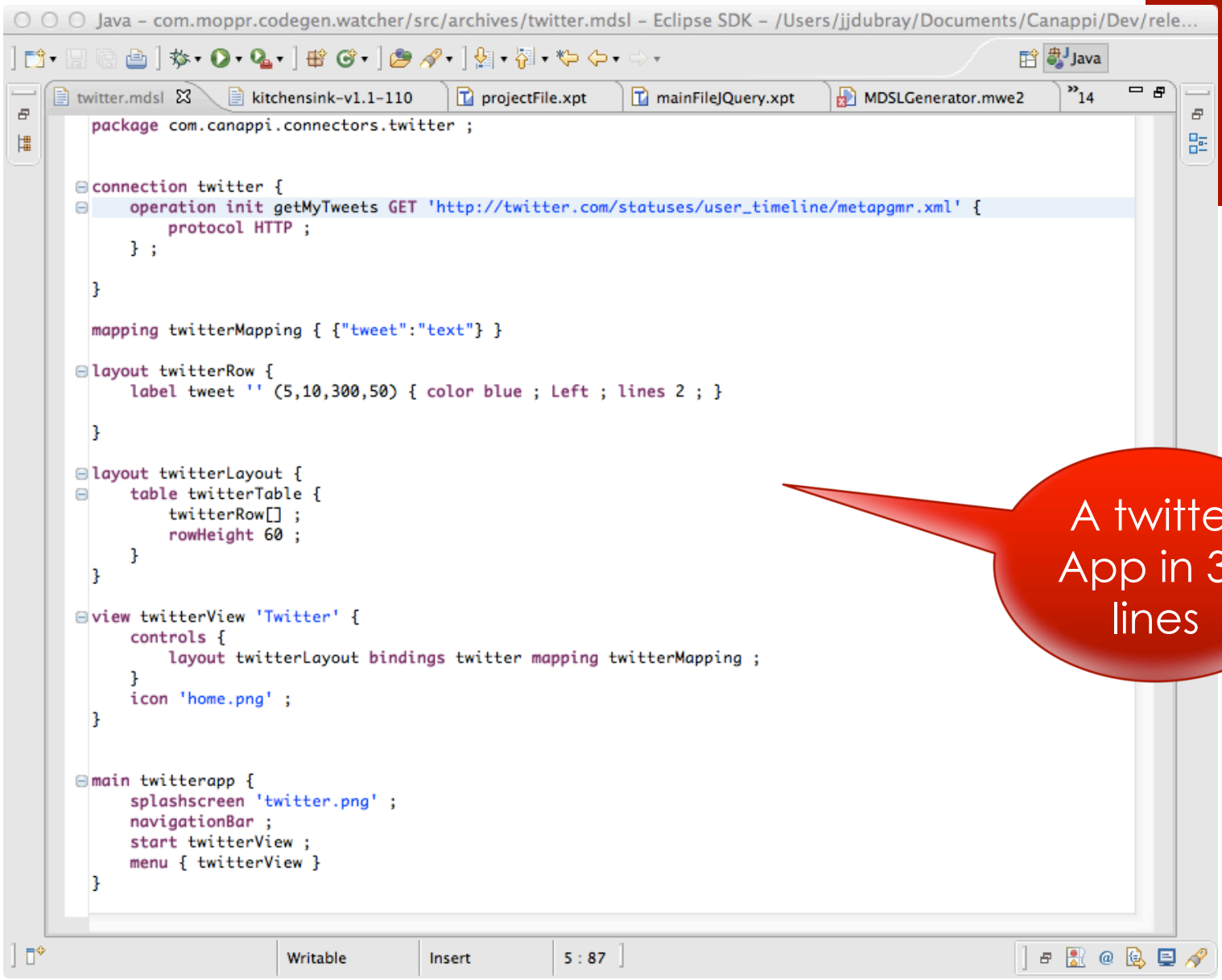
balsamiq

Canappi



Xcode
Android SDK
Nokia WRT
VisualStudio
Sencha
PHP/MySQL

Android
Symbian
WM7
…

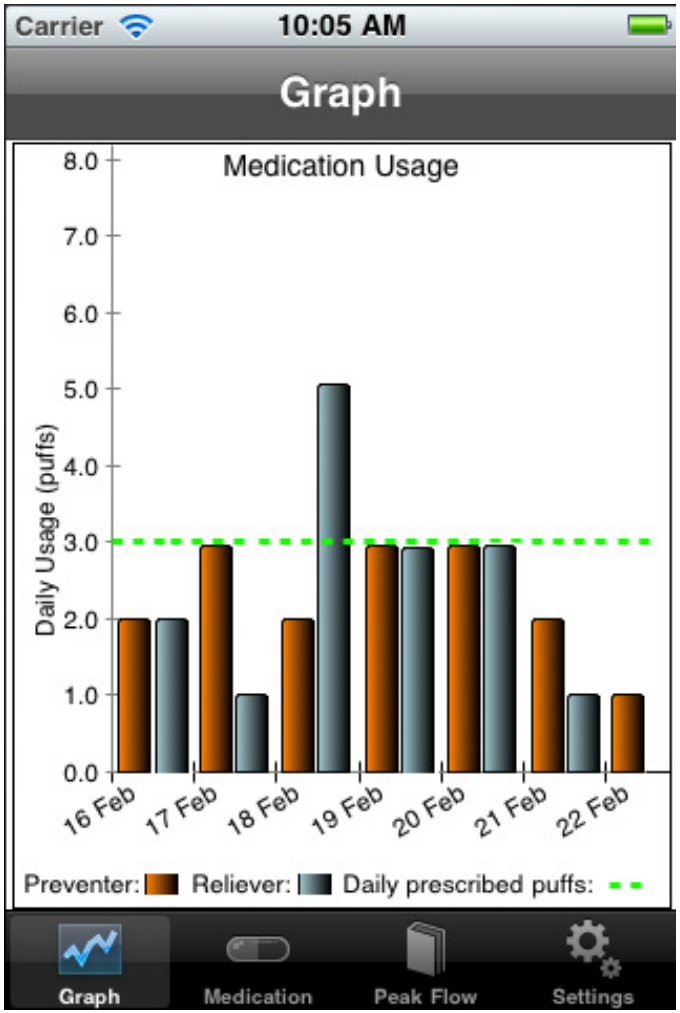

Java – com.moppr.codegen.watcher/src/archives/twitter.mdsl – Eclipse SDK – /Users/jjdubray/Documents/Canappi/Dev/rele...
Java
twitter.mdsl
kitchensink-v1.1-110
projectFile.xpt
mainFileJQuery.xpt
MDSLGenerator.mwe2
14
package com.canappi.connectors.twitter ;
connection twitter {
operation init getMyTweets GET 'http://twitter.com/statuses/user_timeline/metapgmr.xml' {
protocol HTTP ;
} ;
}
mapping twitterMapping { {"tweet":"text"} }
layout twitterRow {
label tweet '' (5,10,300,50) { color blue ; Left ; lines 2 ; }
}
layout twitterLayout {
table twitterTable {
twitterRow[] ;
rowHeight 60 ;
}
}
view twitterView 'Twitter' {
controls {
layout twitterLayout bindings twitter mapping twitterMapping ;
}
icon 'home.png' ;
}
main twitterapp {
splashscreen 'twitter.png' ;
navigationBar ;
start twitterView ;
menu { twitterView }
}
Writable
Insert
5 : 87
A twitter App in 30 lines

Wikityo

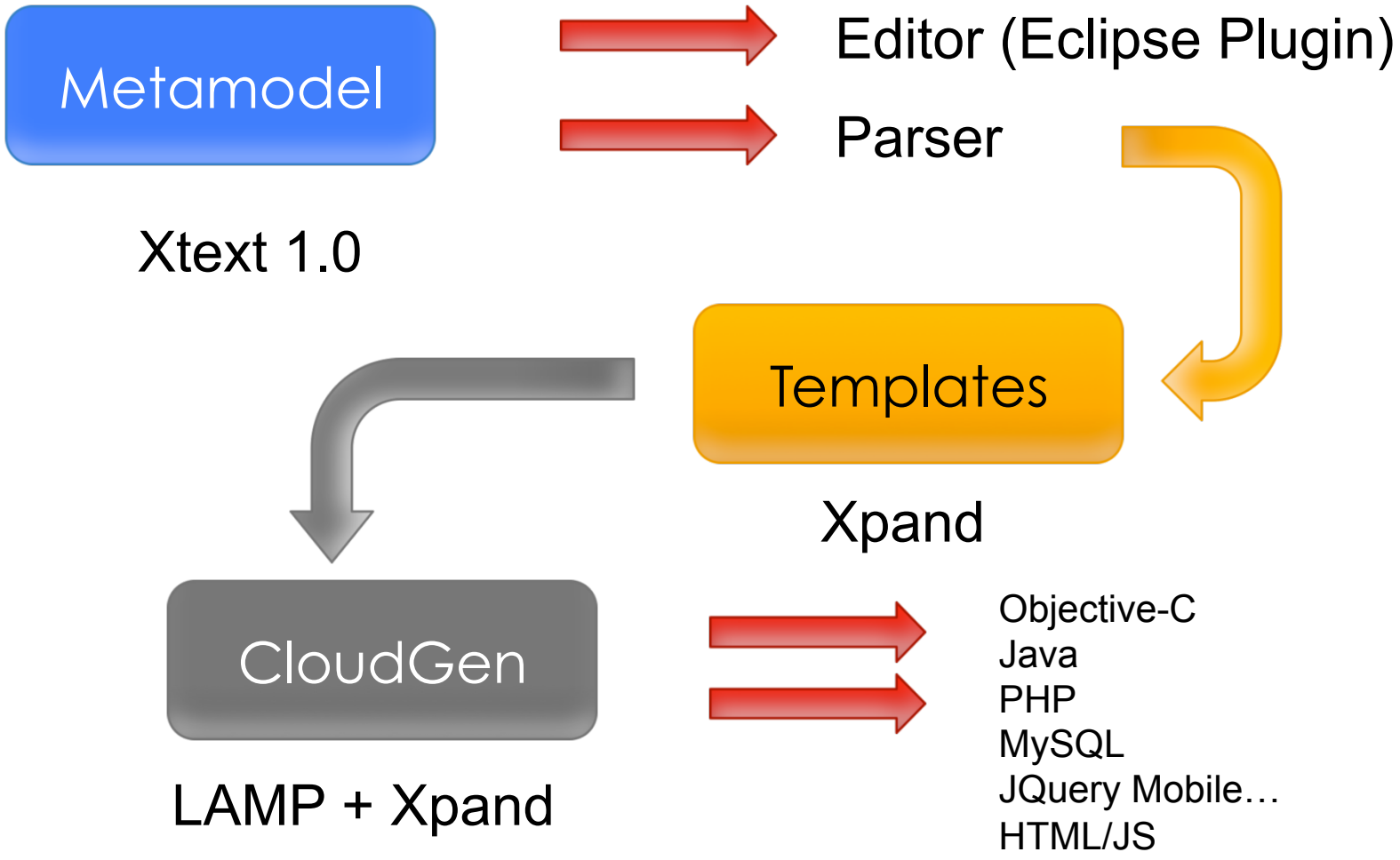
A homework management system that helps kids collaborate

Smart Inhaler Live

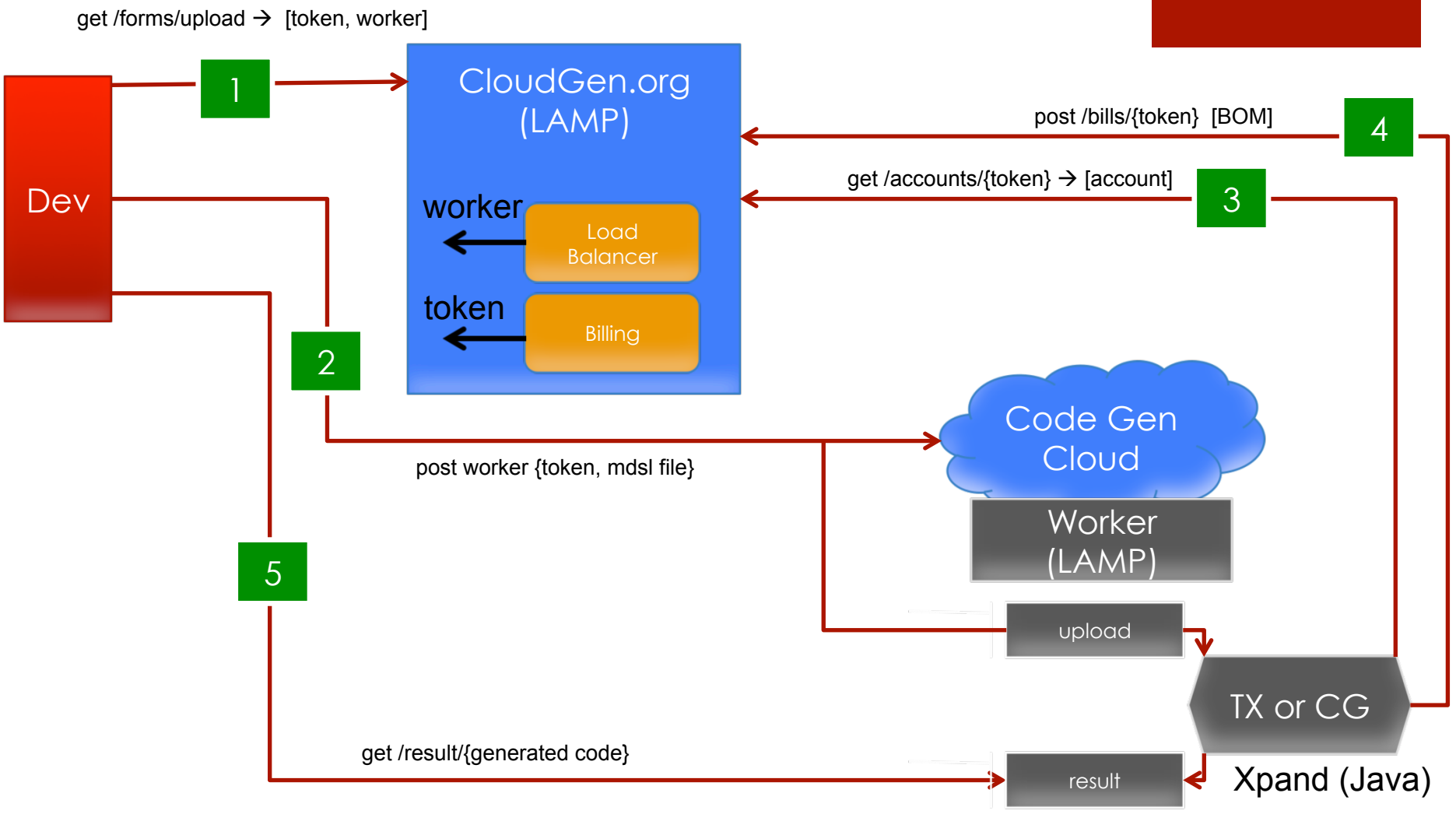
Life changing application for patients with COPD



How does it do it?



CloudGen


get /forms/upload → [token, worker]
1
Dev
CloudGen.org
(LAMP)
worker
Load
Balancer
token
Billing
post /bills/{token} [BOM]
4
get /accounts/{token} → [account]
3
2
post worker {token, mdsl file}
Code Gen
Cloud
Worker
(LAMP)
upload
TX or CG
5
get /result/{generated code}
result
Xpand (Java)

Lessons Learned

Technology Perspective

The Amazing

- Language Workbenches are amazing, truly amazing
- I architected / wrote several large scale interpreters (BPM, Composite Applications, …)
  - Language Workbenches bring a 20 fold productivity increase
- Refactoring your metamodel is straightforward
  - Type checking in Xpand would have been nice
- Regression testing is trivial
  - "Kitchen sink" Test Suite

The Amazing (cont'd)

- Generating code for different platforms from a common metamodel is simple
  - Android uses "resource files"
  - We chose to not use Interface Builder in iOS and generate 100% Objective-C
  - Every platform requires a different template structure
- The future of MDE is textual
  - Lots of tools available to manage "code"
  - Execution Semantics are easier to embed
  - Create visual representations from the model when needed

The Good

- Xpand / Xtend Code Generation is basically a batch process
  - Would benefit from real management capabilities for server side code generation

The Bad

- No constraint language, only custom java code
- Round tripping
  - Mockup Tools → MDSL → Generated Code

- Xtext evolves too fast
  - Canappi is using Xtext 1.0, with little hope to move to 2.0
  - It would mean a complete rewrite
- Mobile technologies evolve too fast, Xtext doesn't make the process to support new versions much easier
  - We would need several more developers just to keep up with change (basically a complete rewrite for Xtext 2.0)

The Bad (cont'd)

- Xpand is grossly under designed
  - Context, Context, Context
  - 1000+ LOC of Java extensions: Xtend 1.0 was useless
    - containsClass
    - Enum management
    - String manipulation
    - Counters / Iterators
    - Processing queues
    - Platform specific behavior
    - Resolution adaptation
  - No "Style Sheet" concept
    - Code generators should be massively parametrizable

Modeling Perspective

The Amazing

- Full Stack Programming Model
  - Entity Model / Data Services
  - Connections to Web APIs
  - Views and data binding
- Dedicated programming Model
  - A simple Twitter App in 30 lines of (mdsl) code
  - Interface Builder is based on a 25 year old Client-side MVC programming model
  - Android has a more modern programming model that goes beyond MVC, but is not "full-stack"
    - Does not support calling 3rd party Web APIs or direct integration with App Engine

The Good

- You can make the language as compact as needed
- You can adopt any syntax
- Xtext 2.0 offers now a type system but I am not sure it is fully needed, constraints are more important

The bad

- No mater how well you design your metamodel, every project has requirements that are not supported
  - → Custom code which mean loss of modeling environment
  - Maybe Interface Builder can teach us a thing or two about metamodel design and programming model integration
- Developers are reluctant to learn a technology that is not mainstream
  - It is not about reusing the skills of developers since the skills are really about the solution model (a.k.a the SDK)
  - It is about what you can put on your resume

What's Next?

What (kind of) works?

- The Web
  - Yes, you heard me correctly, the Web programming model is one of the most successful
- HTML, Javascript, CSS and HTTP form a powerful programming model
  - Yet, broken, **because it never intended to be one**
  - Elements of the programming model are not integrated
  - Server side is absent, HTTP is not enough
- Programming models must embrace and extend that separation of concern, while being more integrated

What is Critically Lacking in Programming Models?

- A protocol centric programming model
  - Document-centric, Object-Centric, Actor-Centric?
  - Webmachine → "executable HTTP"
  - Jim Waldo's paper may no longer apply because we live in a world of Service, and we build "Composite Applications" where **we don't own every part of the system**

A better approach is to accept that there are irreconcilable differences between local and distributed computing, and to be conscious of those differences at all stages of the design and implementation of distributed applications.

Jim Waldo et al, 1994

What is Critically Lacking in Programming Models

- A general assembly mechanism
  - Assembling code is the new frontier
    - Spring
    - SCA
    - OSGi
    - Closures

“Assembly” is the next frontier for Software and System Engineering

- Assemble code
- Assemble parts
- Assemble systems

Building communicating systems is not a well-establish science, or even a stable craft; we do not have an agreed repertoire of construction for building an expressing interactive systems, in the way that we (more-or-less) have for building sequential computer programs.
But nowadays, most computing involves interaction –and therefore involves systems with components which are concurrently active. Computer Science must therefore rise to the challenge of defining an underlying model.

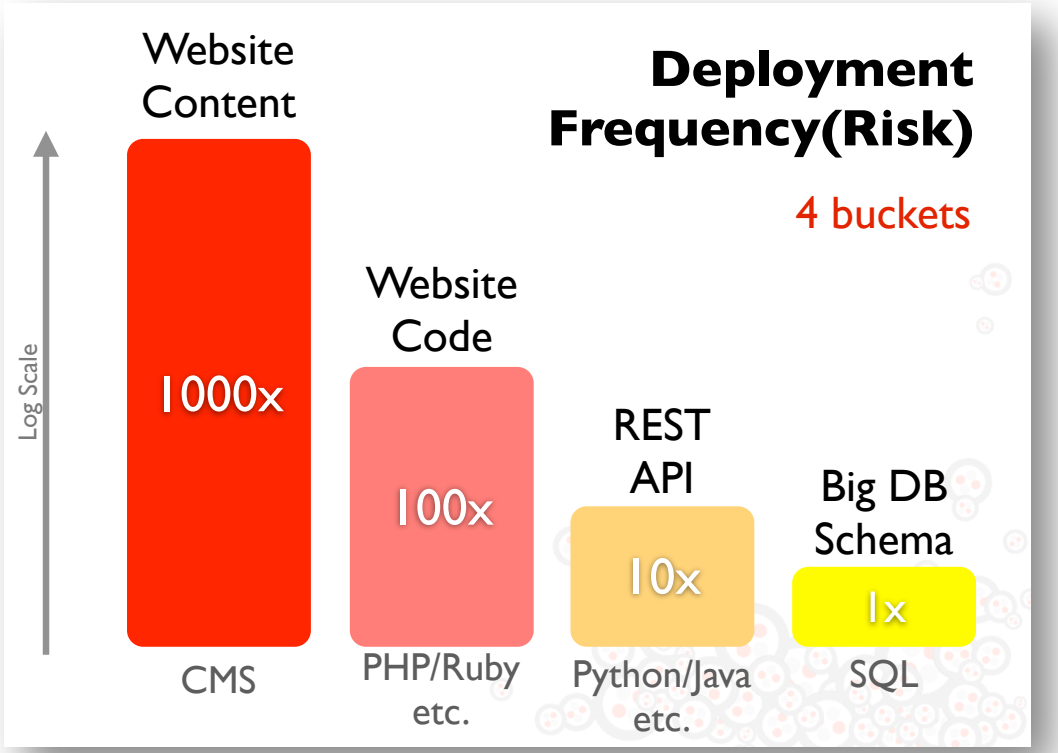
Robin Milner, “Communicating and Mobile Systems: the π-Calculus"

- So yes, MDE has missed the boat, but it is unlikely that the level of “assembly” mechanisms that are needed today will be achieved by something other than MDE

What is Critically Lacking in Programming Models?

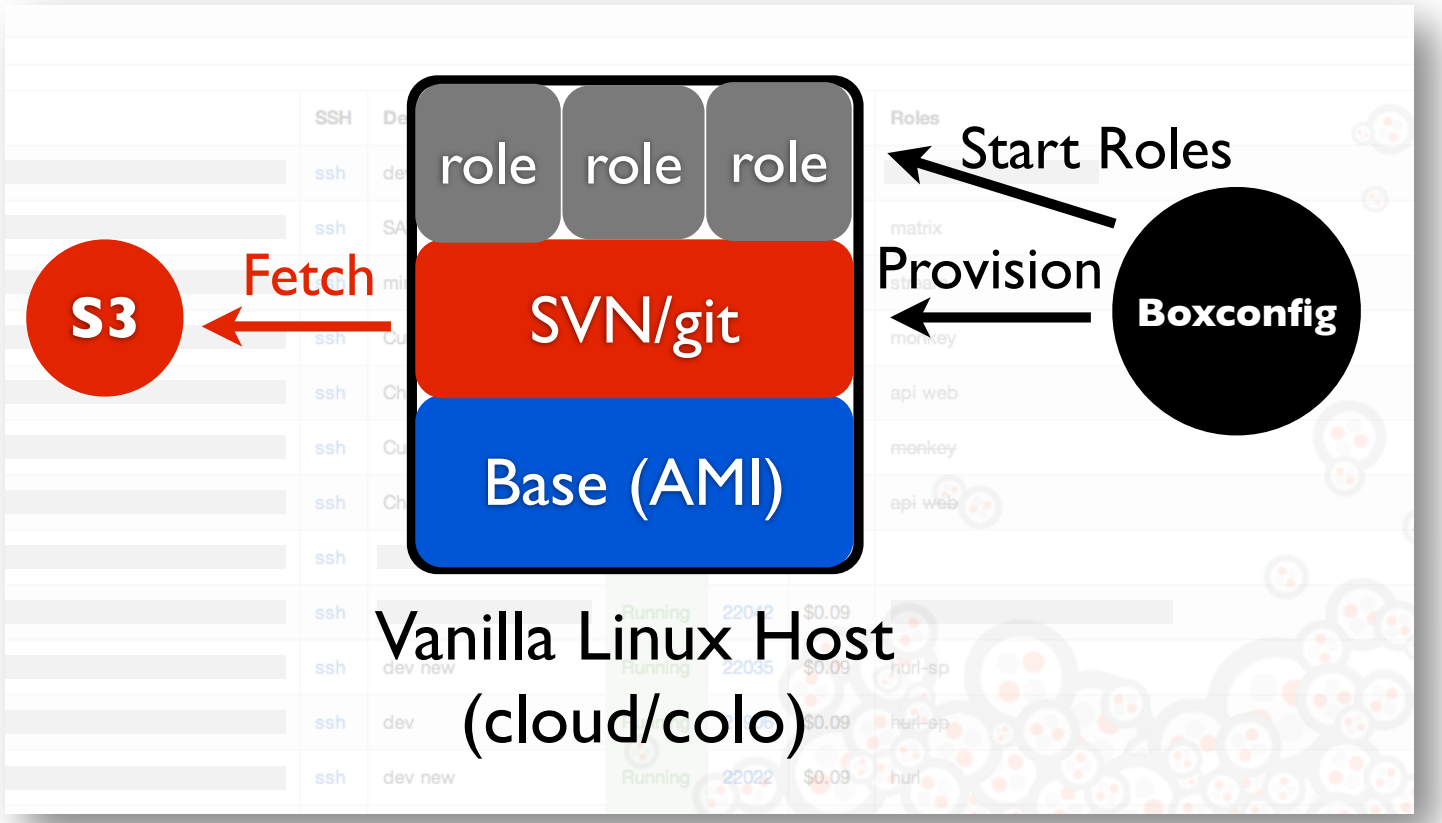
- A deployable programming model
  - Cloud is changing drastically how we deploy solutions
    - Continuous Deployment
    - Scale-out Deployment

Source: Evan Cooke, Qcon 2011



Cluster Deployment with Boxconfig

Source: Evan Cooke, Twilio, Qcon 2011



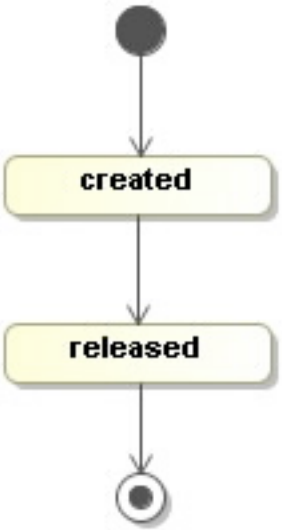
MetaArchitecture Framework

- L'Ecole des Mines de Nantes and Jean's Group in particular has been one of the strongest influence in the way I think about modeling and programming
- How do you translate this sentence into a practical approach to "execution semantics definition"?
- How can you apply it in an "Assembly" context, rather than a pure "execution semantics"

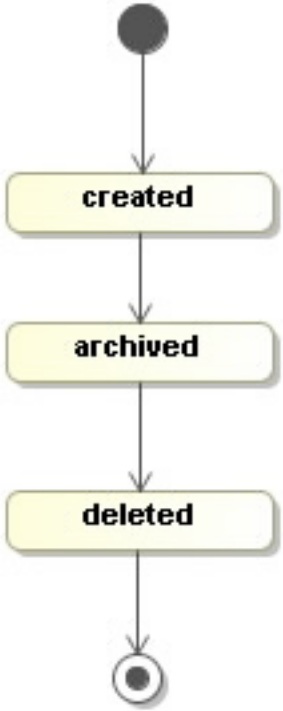
A DSL may have an execution semantics definition. This semantics definition is also defined by a transformation model mapping the DDMM onto another DSL having itself an execution semantics or even to a GPL. The firing rules of a Petri net may for example be mapped into a Java code model.

KM3, F. Jouault & J. Bezivin

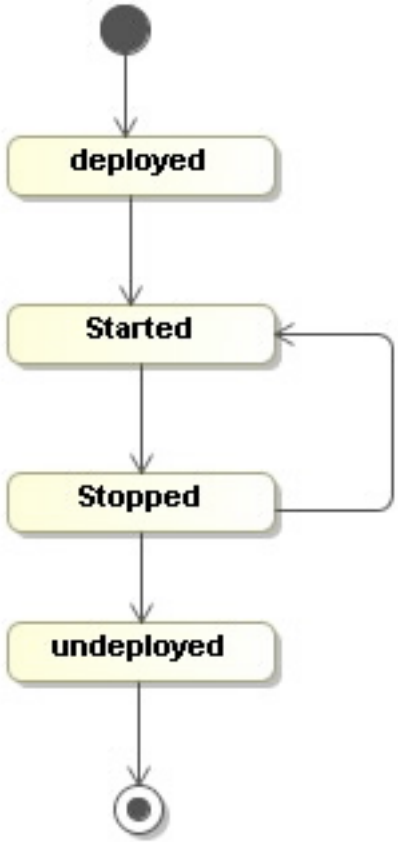
MAF
Architecture Elements and their lifecycle



Object



Information Element



Service

It is essential to distinguish the "actions" from the "code"

- Lifecycle Actions
  - Which element can invoke them
  - Which one can be invoked
  - Great opportunity to define and enforce constraints on the system
- Inter-Actions
  - Idempotency
  - In relation to lifecycle actions
- "Code" is no longer relevant, because we are no longer orchestrating the behavior of a microprocessor
  - We are choreographing inter-actions

Conclusion

- Mobility opens a new, fascinating, era of computing
  - Don't just build a desktop app running in a mobile phone
- Building successful mobile apps is not that easy
  - Create a software factory
- Create a modern programming model
  - Tweaking programming languages will not be enough
  - "Assembly" is the next frontier to software engineering
    - Executable → Actionable
  - MetaArchitecture Framework