Connecting Software Architecture to Implementation: The Next 10 Years

Most Influential Paper of ICSE 2002 Award Talk
2012 International Conference on Software Engineering

Jonathan Aldrich
Craig Chambers
David Notkin

Carnegie Mellon
Google
University of Washington

Thanks!

- We're honored that our paper has had an influence!
  - and the credit goes also to those who influenced and carried forward ArchJava (more in the talk)

- More broadly, our work is one piece of an important trend
  - Software architecture is becoming a valuable abstraction *in practice!*
  - Languages and tools relate code more directly to architecture
  - Tools verify architectural properties of software

- This talk
  - A bit about our ICSE'02 paper
  - How the trend linking architecture and code has grown since
  - What the future may hold

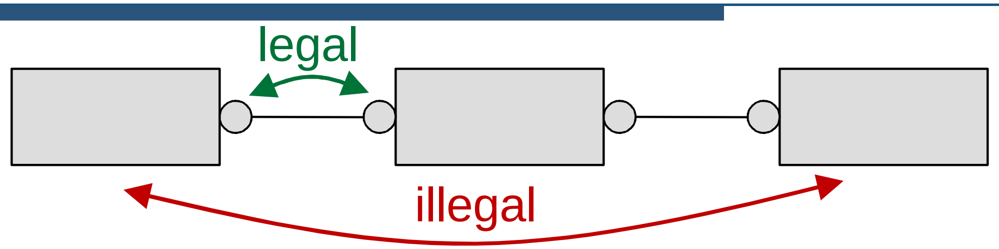
ICSE’02 Research Context

- Software architecture was an established concept
  - *The structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time.* [Garlan & Perry, 1995]

- However, still maturing in practice
  - Last 3 stages of the Redwine-Riddle model [Shaw & Clements 2006]
    - internal/external exploration and enhancement, and popularization

- ICSE’02 Motivation and research question
  - Achieving the benefits of an architecture requires following it. Can we assure that code does so?

Architectural Conformance



- Prior work identified **communication integrity** as a key architectural conformance property

Interfaces [of a component] may communicate directly only if there is an architecture connection between the interfaces [Luckham & Vera 1995]

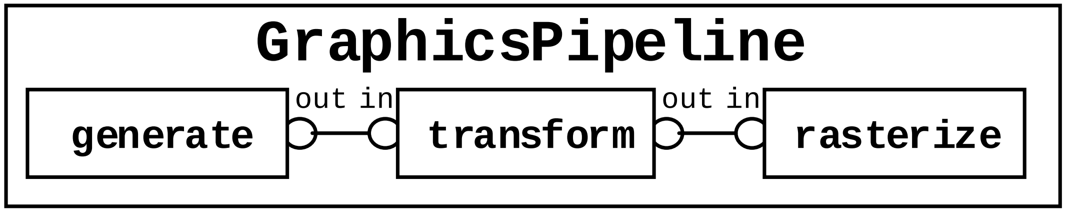
- Prior work on enforcing integrity
  - Theory of conformance [Moriconi et al., 1995]
  - Follow style guidelines [Luckham & Vera, 1995]
  - Use developer-directed analysis to extract (module) architecture [Murphy et al., 2001]

ArchJava's Approach

- Connect architecture with implementation by:
  - Embedding architecture in the programming language
    - Context: component and connector architecture, object-oriented code
  - Using a type system to ensure communication integrity

- Hypothesized benefits
  - **Traceability** – can easily answer architecture questions about code
  - **Communication integrity** – feasible to check
  - **Co-evolution** – architecture and code remain consistent
  - **Executable architecture** – architectural declarations are "live"
  - **Saliency** - architecture becomes an constant part of development

ArchJava Example: Graphics Pipeline



```
public component class Transform {
    public port in {
        provides void draw(Shape s);
    }
    public port out {
        requires void draw(Shape s);
    }
    void draw(Shape s) {
        currentTransform.apply(s);
        out.draw(s);
    }…
```

```
public component class GraphicsPipeline {
    protected Generate generate = ... ;
    protected Transform transform = ... ;
    protected Rasterize rasterize = ... ;
    connect generate.out, transform.in;
    connect transform.out, rasterize.in;
}
```

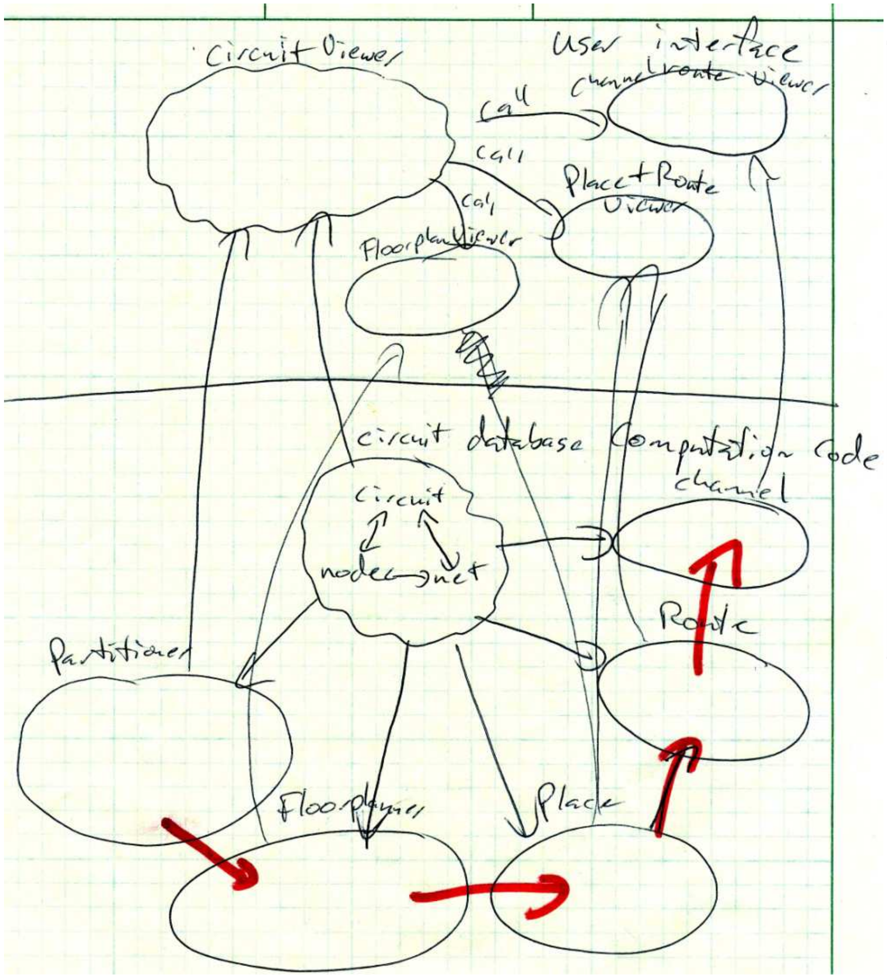
Architectural interfaces

Ordinary Java code

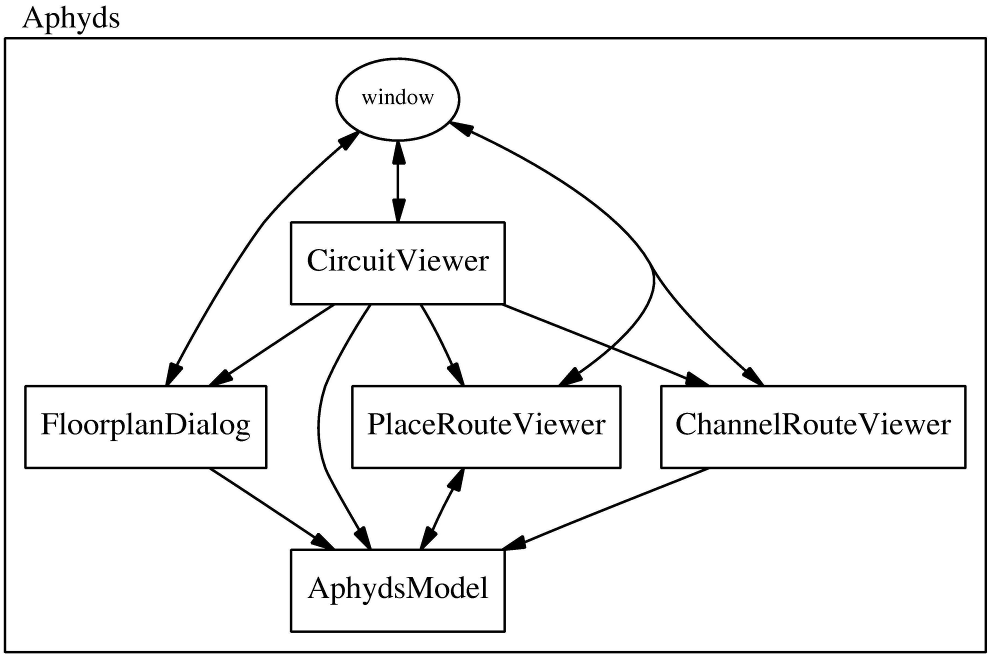
Connections link components

Typing rules prohibit passing component references to another component → cannot bypass connections

[Exploratory] Case Study with Aphyds

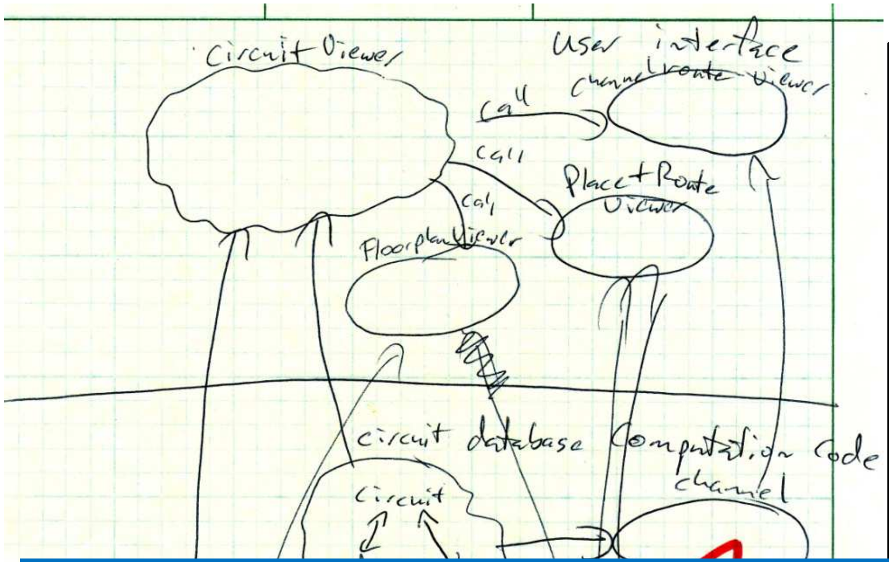


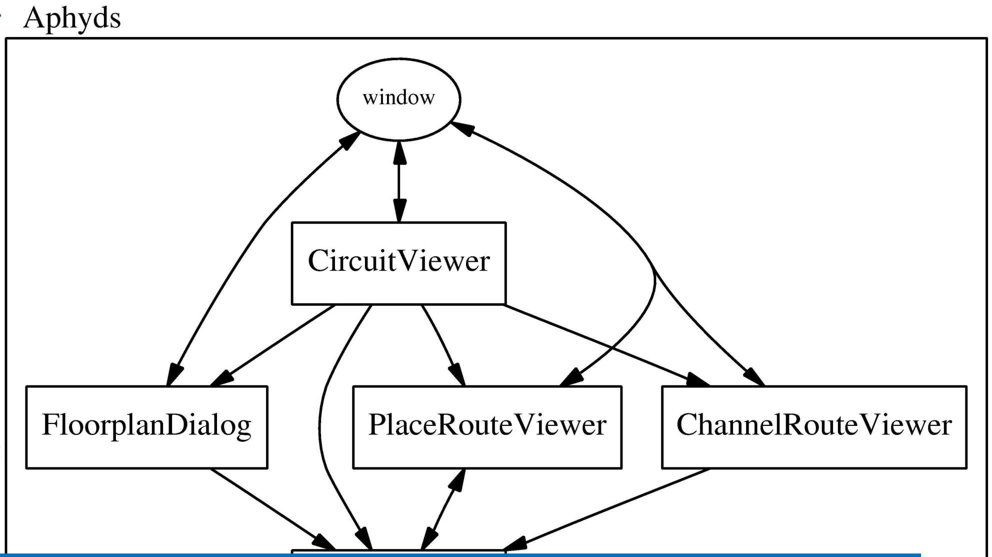
Architectural Drawing
by Developer



Architectural Visualization based
on ArchJava

[Exploratory] Case Study with Aphyds





ed

- Some observations
  - Exploratory study generated interesting hypotheses (see paper)
  - The state of practice is still pretty informal (whiteboard diagrams)
    - Formality must provide value – connection to implementation is one way
  - Programming languages are (still) hard to evaluate
    - Large-scale, in-situ case studies or experiments not always realistic
    - Still worthy of exploration
    - Must find validation appropriate to claims

Why ArchJava Worked (as well as it did)

- Reaction from some: that's impossible!
  - Conformance of a program to an architecture is undecidable
  - Static analysis will have many false warnings due to abstraction
- True. We changed the game:
  - Developers integrate **design intent** into code
    - Using coding idioms that map to architecture
    - Types show how/why code conforms
    - Our goal: developers need not change macro-architecture to do this
  - This is (part of) making architecture more salient to developers
- This is a strength of a language-based approach
  - But also a weakness, creating difficulties for legacy code

The Last 10 Years: ArchJava

- ArchJava extensions
  - **Dynamic architectures** [ECOOP '02]
    - Inspired by Magee & Kramer, Dynamic Structure in Software Architectures, FSE '96
  - Communication integrity with **shared data** [WICSA '08]
    - Building on ownership [Noble et al. '98] and shared data connectors [Garlan and Shaw '93][Moriconi et al., '95]
  - Flexible **connector abstractions** [ECOOP '03]
    - Implement different connector semantics – both dynamics & typechecking
    - Inspired by and evaluated by Medvidovic et al.'s taxonomy [2000]

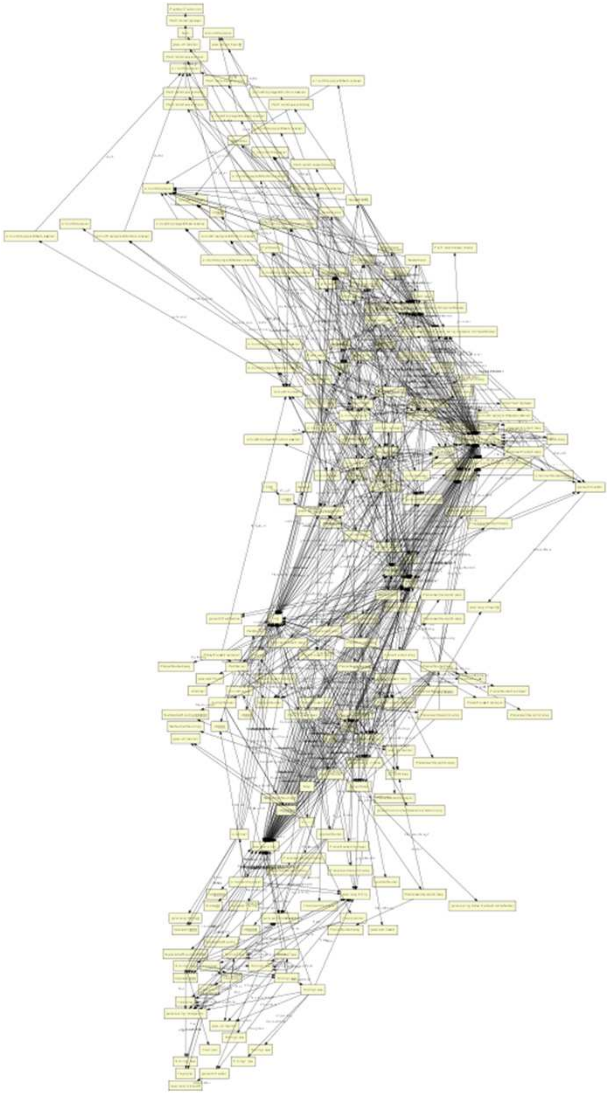
Connecting Architecture to Implementation

A selection of scientific work citing ArchJava:

- Generation & verification of **control systems** [Cassou et al, ICSE '11]
- **Synthesis** of architecturally correct code [Bagheri, ICSE '11]
- Automated **runtime validation** of architecture [Dong et al., '05]
- Architectural annotations in code for **Agile** [ICSE NIER '11]
- Architecture-driven **mobility** frameworks [Malek et al., '10]
- Component-oriented languages with first-class **connectors** [Chen et al '06]
- Checking architecture in legacy **scientific applications** [Woollard et al '09]

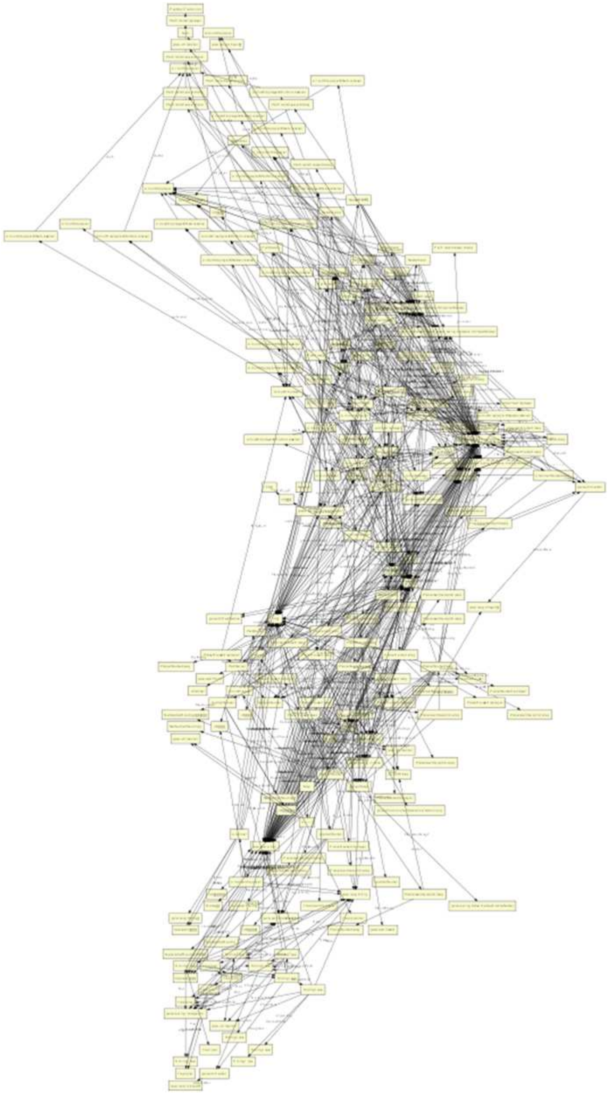
An example I've been involved in: the Scholia system

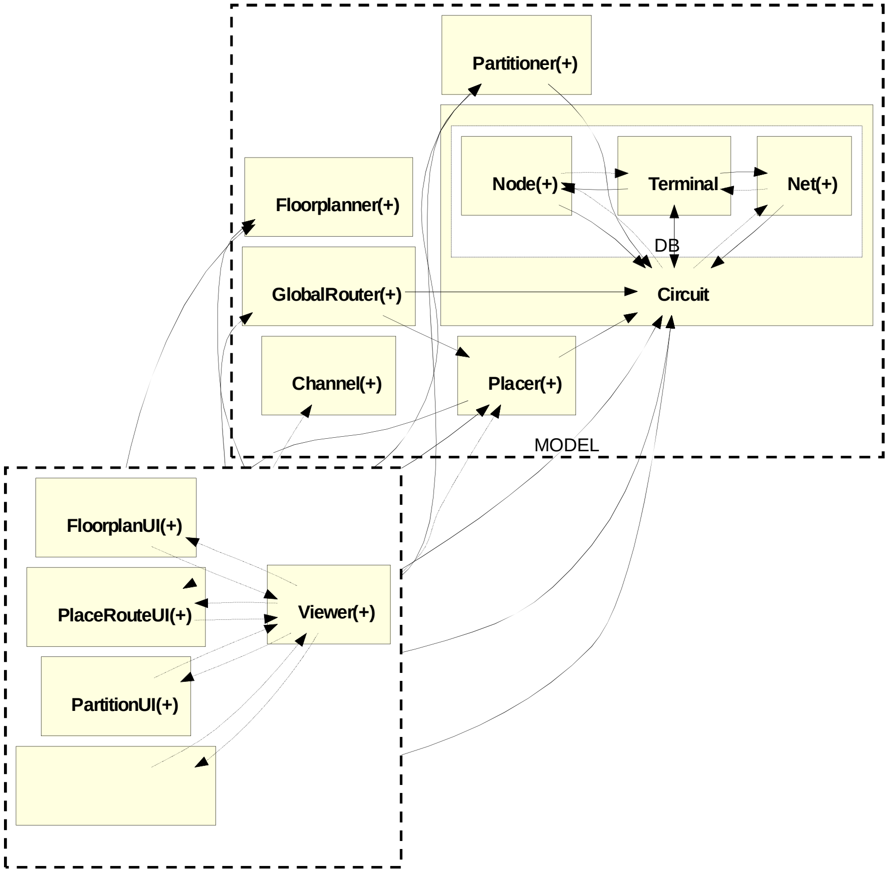
Motivation: Scalable Visualization of Object Graphs



Prior work: too many edges, hard to abstract

Using Design Intent to Extract Object Graphs



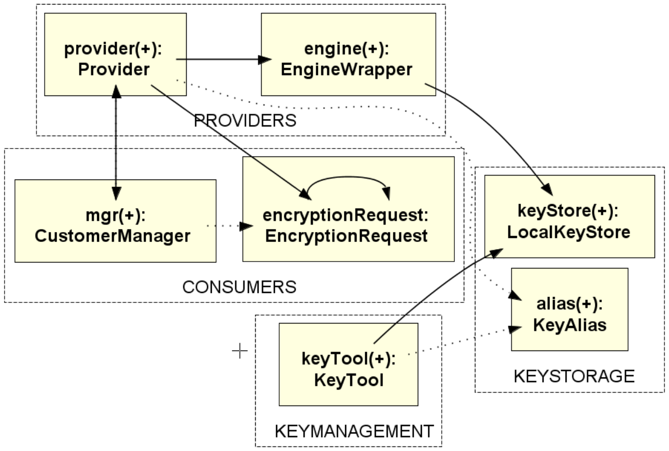


[Abi-Antoun & A, OOPSLA ‘09]

Declaring Architectural Intent

- Labeled groups
  - @Domain: Put in logical part of architecture

```
class Main {
                                        Provider provider;
                                         CustomerManager mgr;
                                           LocalKeyStore keyStore;
}
```

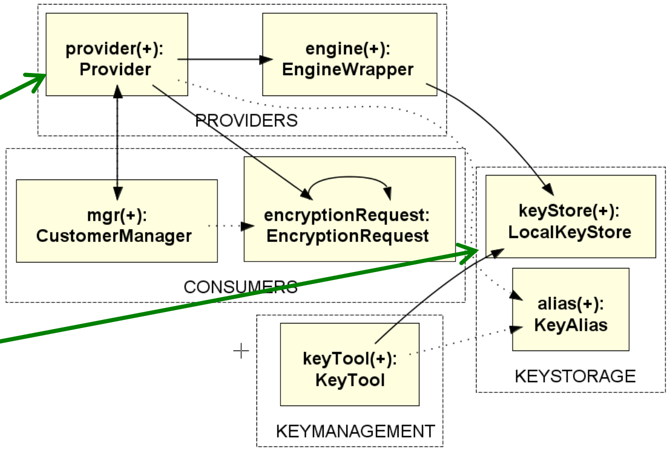


[Abi-Antoun & A, OOPSLA '09]

Declaring Architectural Intent

- Labeled groups
  - @Domain: Put in logical part of architecture

```
class Main {
    @Domain("PROVIDERS") Provider provider;
    @Domain("CONSUMERS") CustomerManager mgr;
    @Domain("KEYSTORAGE") LocalKeyStore keyStore;
}
```

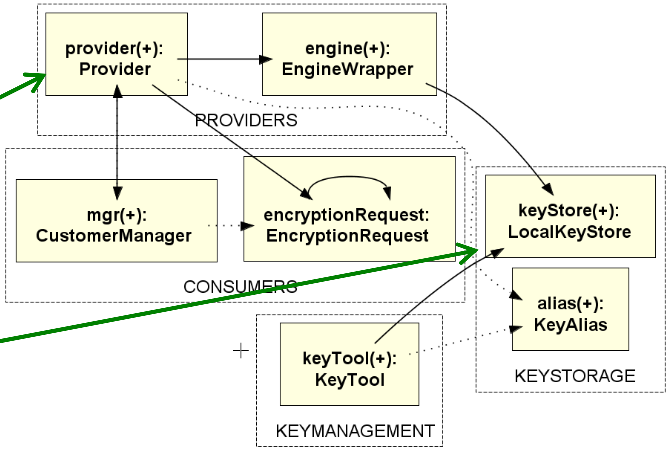


[Abi-Antoun & A, OOPSLA '09]

Declaring Architectural Intent

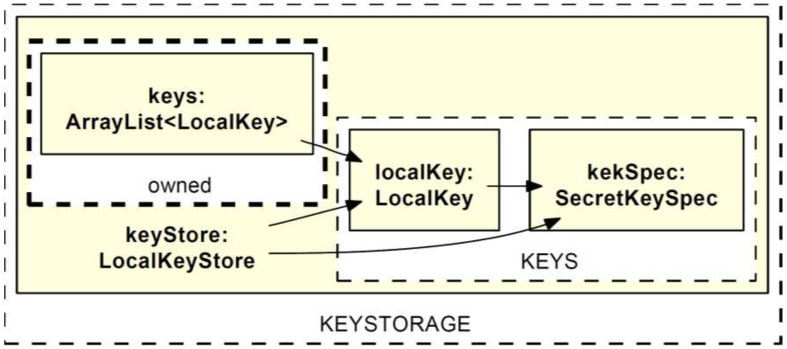
- Labeled groups
  - @Domain: Put in logical part of architecture

```
class Main {
    @Domain("PROVIDERS") Provider provider;
    @Domain("CONSUMERS") CustomerManager mgr;
    @Domain("KEYSTORAGE") LocalKeyStore keyStore;
}
```



- Data structure encapsulation
  - OWNED: Hide data objects within high-level abstractions

```
class LocalKeyStore {
                    List<LocalKey> keys;
}
```

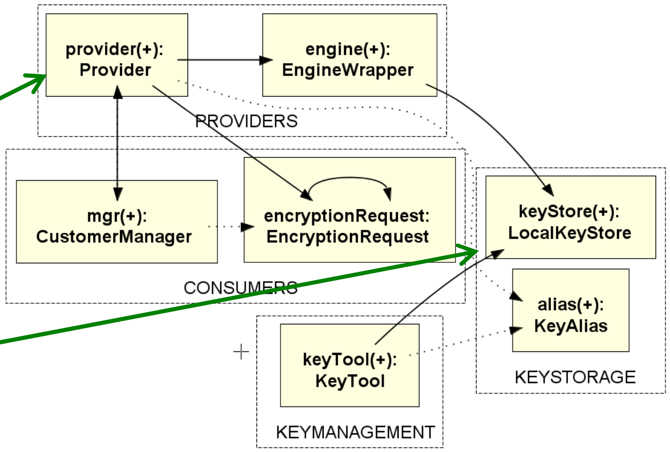


[Abi-Antoun & A, OOPSLA '09]

Declaring Architectural Intent

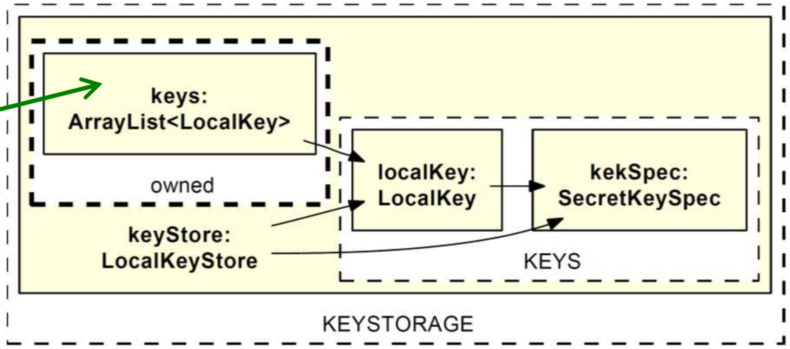
- Labeled groups
  - @Domain: Put in logical part of architecture

```
class Main {
    @Domain("PROVIDERS") Provider provider;
    @Domain("CONSUMERS") CustomerManager mgr;
    @Domain("KEYSTORAGE") LocalKeyStore keyStore;
}
```



- Data structure encapsulation
  - OWNED: Hide data objects within high-level abstractions

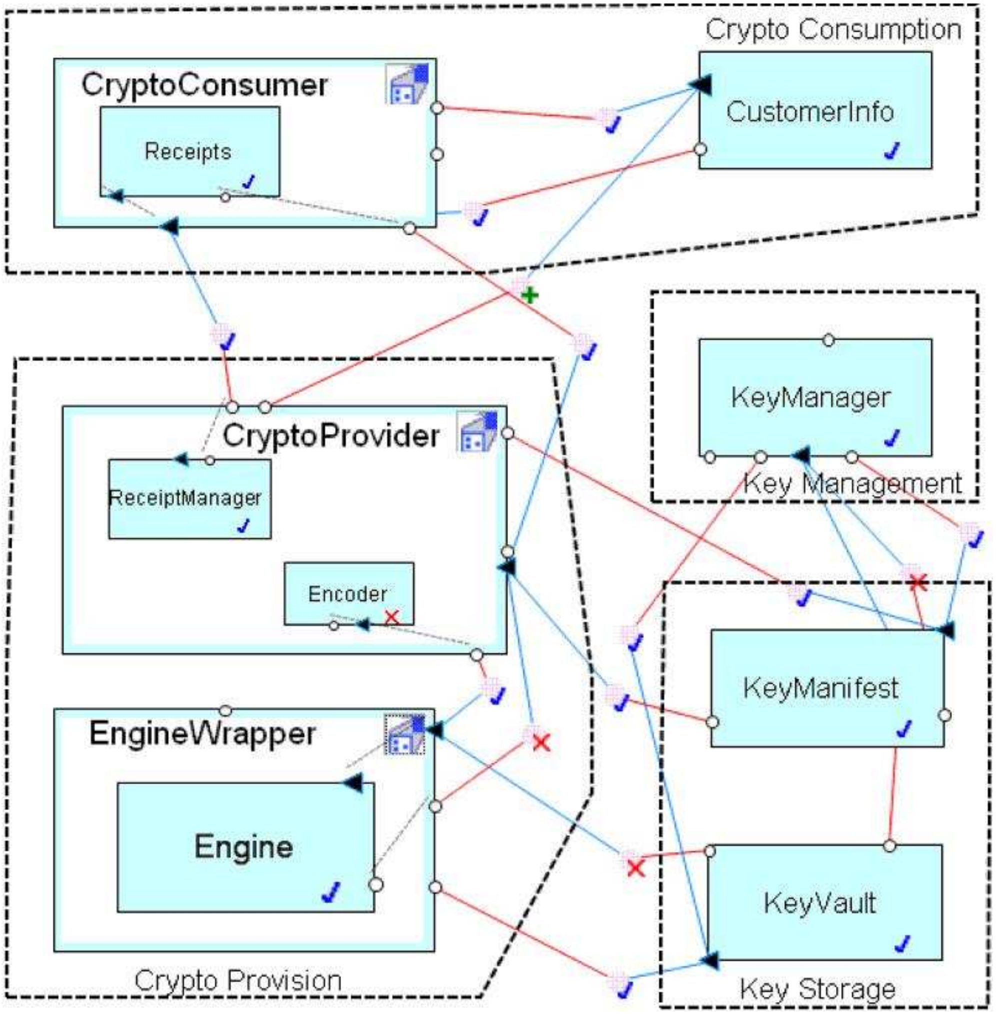
```
class LocalKeyStore {
    @Domain("OWNED<KEYS>") List<LocalKey> keys;
}
```



[Abi-Antoun & A, OOPSLA '09]

CryptoDB Case Study Results

- Architecture shows deltas
  - Intended vs. actual



[Abi-Antoun & Barnes, ASE ‘10]

Realizing the Vision: The Next 10 Years

- What new architecture-implementation connections can we make?

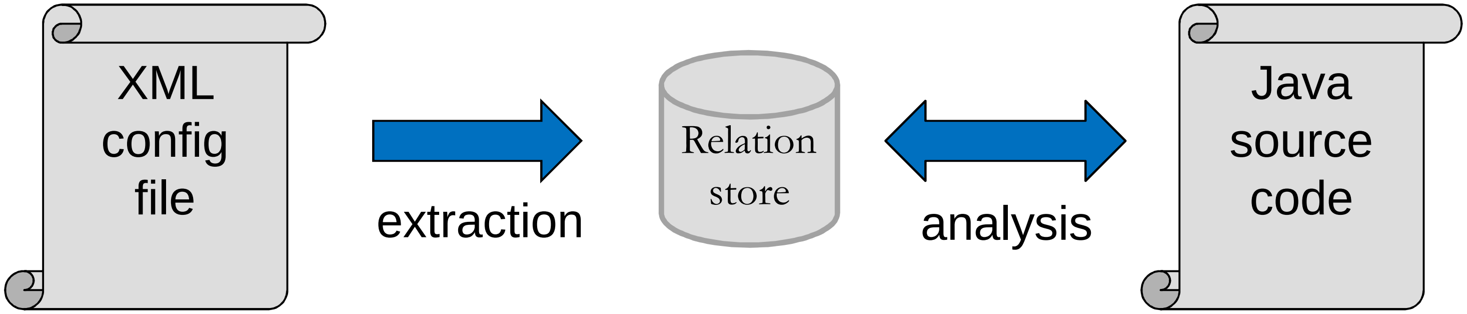
- How to make architecture part of everyday development?

Example: Architecture in Industry Frameworks

- Framework config files describe structure, properties
  - Web app frameworks (Spring, Rails)
    - Structure, security
  - Mobile frameworks (e.g. Android)
    - Event communication, UI flow, security

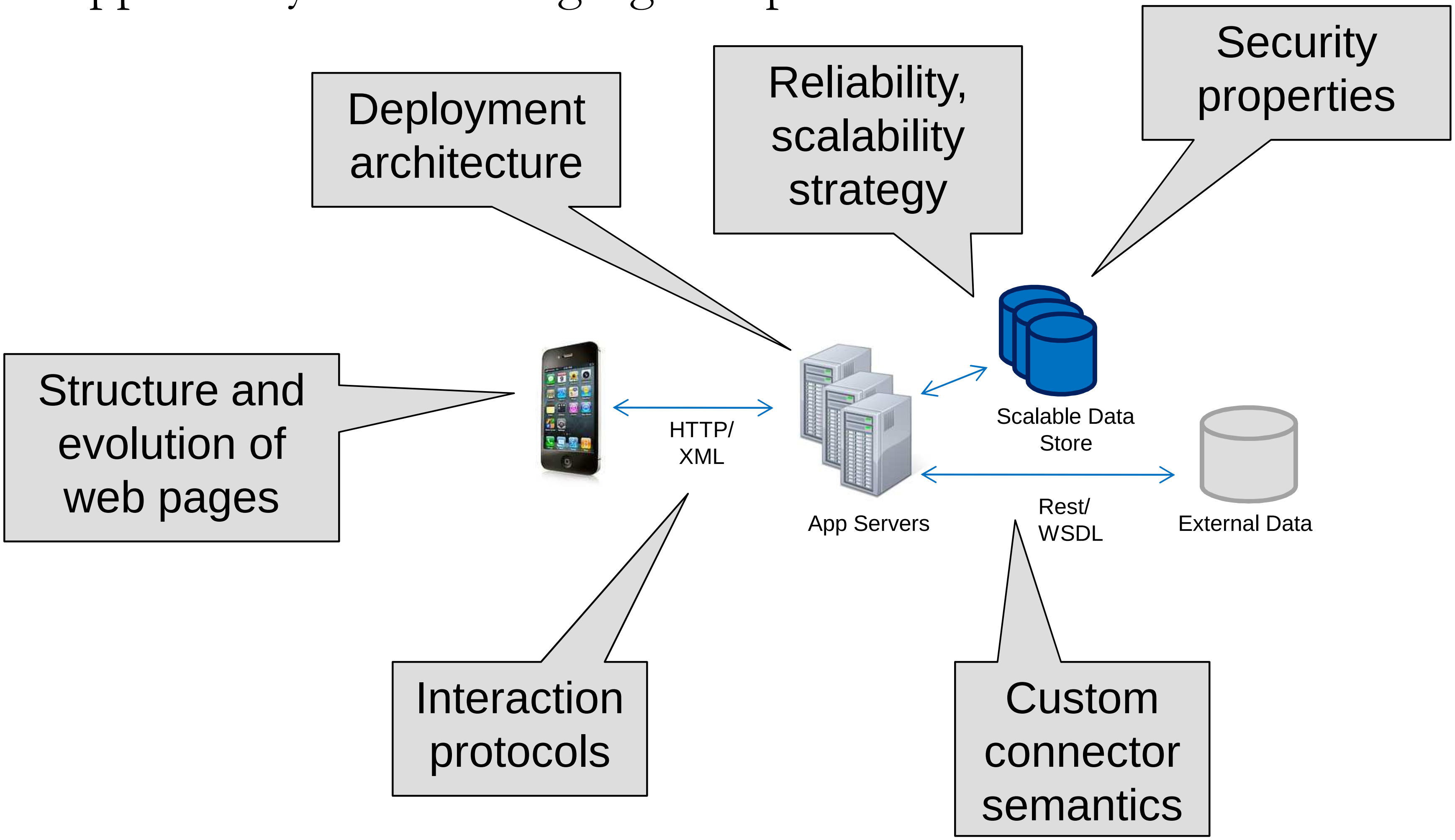
- Can we check consistency?
  - Framework-specific tools exist—do they generalize?

- FUSION tool at CMU/Cal Poly Pomona [C. Jaspan thesis, 2011]



Current Work: Mobile Web App Architecture

- Opportunity for new language adoption



Deployment architecture

Reliability, scalability strategy

Security properties

Structure and evolution of web pages

Interaction protocols

Custom connector semantics

Connecting Architecture to Implementation

- 10 years later, we have made progress
  - Making architectural verification more practical
  - Support for new kinds of synthesis, analysis
  - Domains such as mobility, scientific computing

- Many opportunities to have impact in practice!
  - Configuration as architecture
  - Emerging systems (web, mobile)
  - Exposing architecture in code

Acknowledgments

- Contributors to ArchJava
  - Marwan Abi-Antoun
  - Kevin Bierhoff
  - Wesley Coelho
  - Sangjin Han
  - Valentin Kostadinov
  - Nagi Nahas
  - Vibha Sazawal
  - Tony Tseng
- Contributors to other work described
  - Marwan Abi-Antoun
  - Jeff Barnes
  - Ciera Jaspan
- The many who inspired the work
  - Cited above, and in the paper