GrassRoots:

Socially-Driven Web Sites for the Masses

Frank Uyeda

Diwaker Gupta, Amin Vahdat, George Varghese

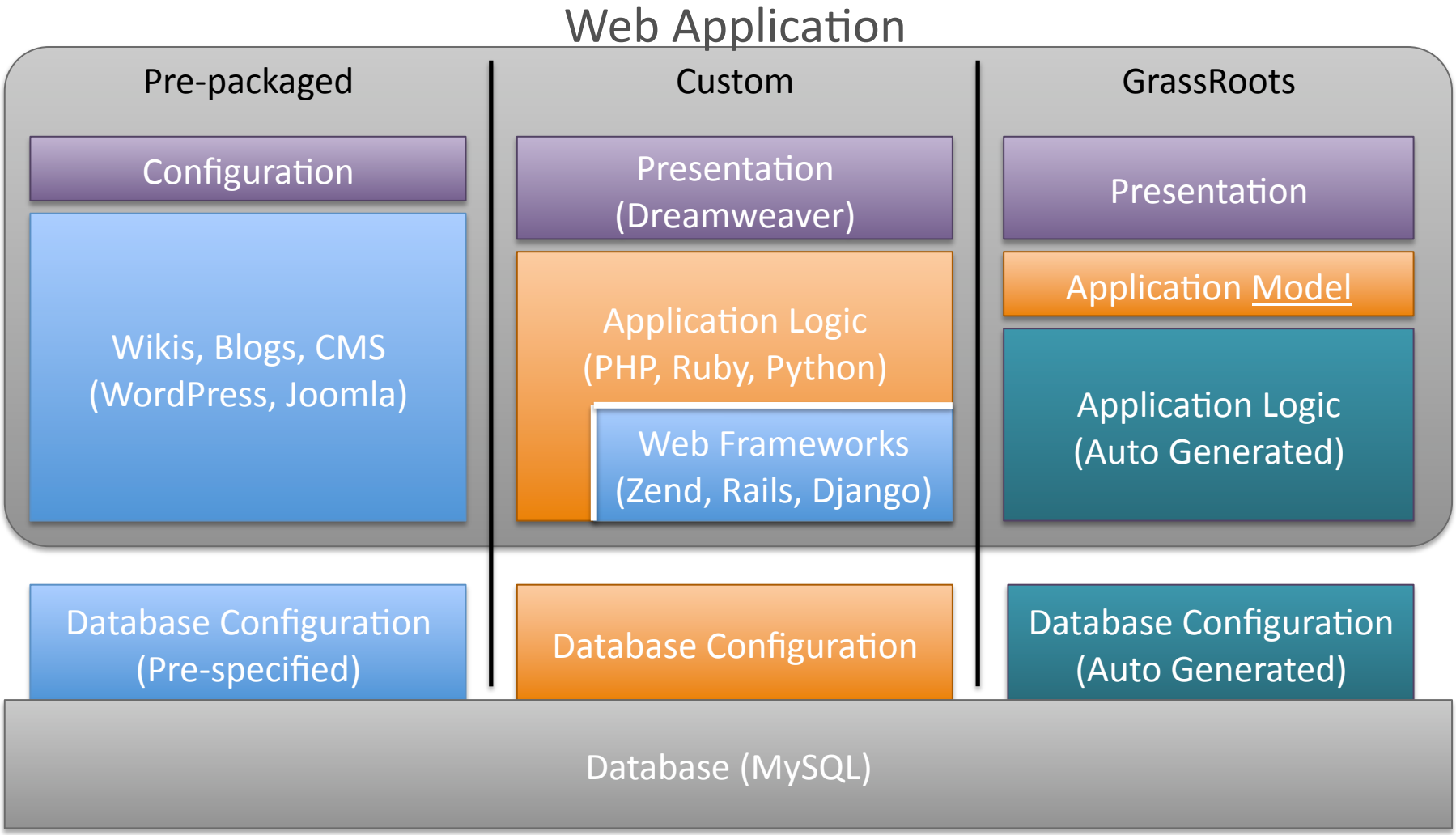
University of California, San Diego

UCSDCSE
Computer Science and Engineering

This Talk in a Nutshell

- Grass-roots communities wish to have websites that allow them to submit and flexibly search for data.
- These communities require tools that are simpler than those currently available (e.g. Apache, PHP, MySQL).
- To build such tools we need models for:
  - data objects, page layout, and, most importantly, search.
- We have instantiated these model in an easy to use language called GrassRoots, and compiler called GR.

Web Application Development



Outline

- Motivation
- Modeling
- Results

The Spread of the Social Web

- Big social networks -- international phenomenon.
  - North America: Facebook (250M)
  - South America / India: Orkut (67M users)

- Not just "social networks"
  - YouTube (video sharing), Digg (social bookmarking)

- Growing interest in smaller sites specialized by industry, enterprise and communities.
  - Wellpoint (insurance)
  - Cisco (company specific)

Who Wants Social Sites?

- Lots of people!
- Example: Physics Researchers
  - Want to share data sets, tag interesting features
- Example: Digital Artists
  - Want to share data visualization programs & collaborate
- Example: Local Parents & Baby-sitters
  - Want job postings, referral network

- Require database & application logic.

These communities lack resources and expertise.
Need cheap, easy-to-use tools!

Limitations of Existing Tools

- Difficult to prototype new ideas
  - Not clear which web development framework will work best
  - Require knowledge of database schemas, web programming languages, design techniques

  Addressed in this talk
- Significant time and expertise needed to develop an operational site
  - Large, complex code base
  - Integration with user management, access control and web API's
  - Engineer for security and privacy
- General techniques to scale are unknown
  - Performance tuning is "black magic"
  - Hire a consultant

Opportunity: Different But Similar

Site	Objects	Search
Flickr	Images	Keyword, Tags, Comparison (geo-tags)
YouTube	Video	Keyword, Tags
Last.fm	Audio	Tags, Structural
Del.icio.us	URLs	Tags
Digg	URLs	Taxonomy, Keyword
Craigslist	Listings (Image + Text)	Taxonomy, Keyword, Comparison
Wikipedia	Articles (Text)	Keyword, Structural
Facebook	User Profile (Image + Text)	Structural, Tags

The Essence: Search & Submit

- Users create and upload content
- Content organized/ranked based user input
- Search based on:
  - Associated keywords (e.g., Del.icio.us tags)
  - Structural relationships (e.g., friends in Facebook)
  - Taxonomy / Hierarchy (e.g., categories in Digg)
  - Comparison / Proximity (e.g., geo-tagging in Flickr)

Approach: Add Abstraction

- Model: Create an abstract model for community driven web sites.
- Specify: Allow developers to express an abstract site model in the GrassRoots language.
- Compile: The GrassRoots compiler generates web code and configures storage.

Outline

- Motivation
- Modeling
- Results

New Insights Behind Our Model

- Insight 1 (Layout): Pages are composed of panes that are populated by search results.
- Insight 2 (Navigation): All navigation is search.
- Insight 3 (Search): Graph-based search with attribute filtering covers existing social search mechanisms.

Data Modeling: like UML

- GrassRoots objects:
  - High-level types (e.g., video, image, text),
  - Composite types like C-structs
  - Built in attributes: taggable, commentable
- Relationships as Graphs
  - General Graph (e.g. friends in Facebook)
  - Directed Graph (e.g. YouTube Subscribers)
  - Hierarchy / Tree (e.g. Craigslist categories)

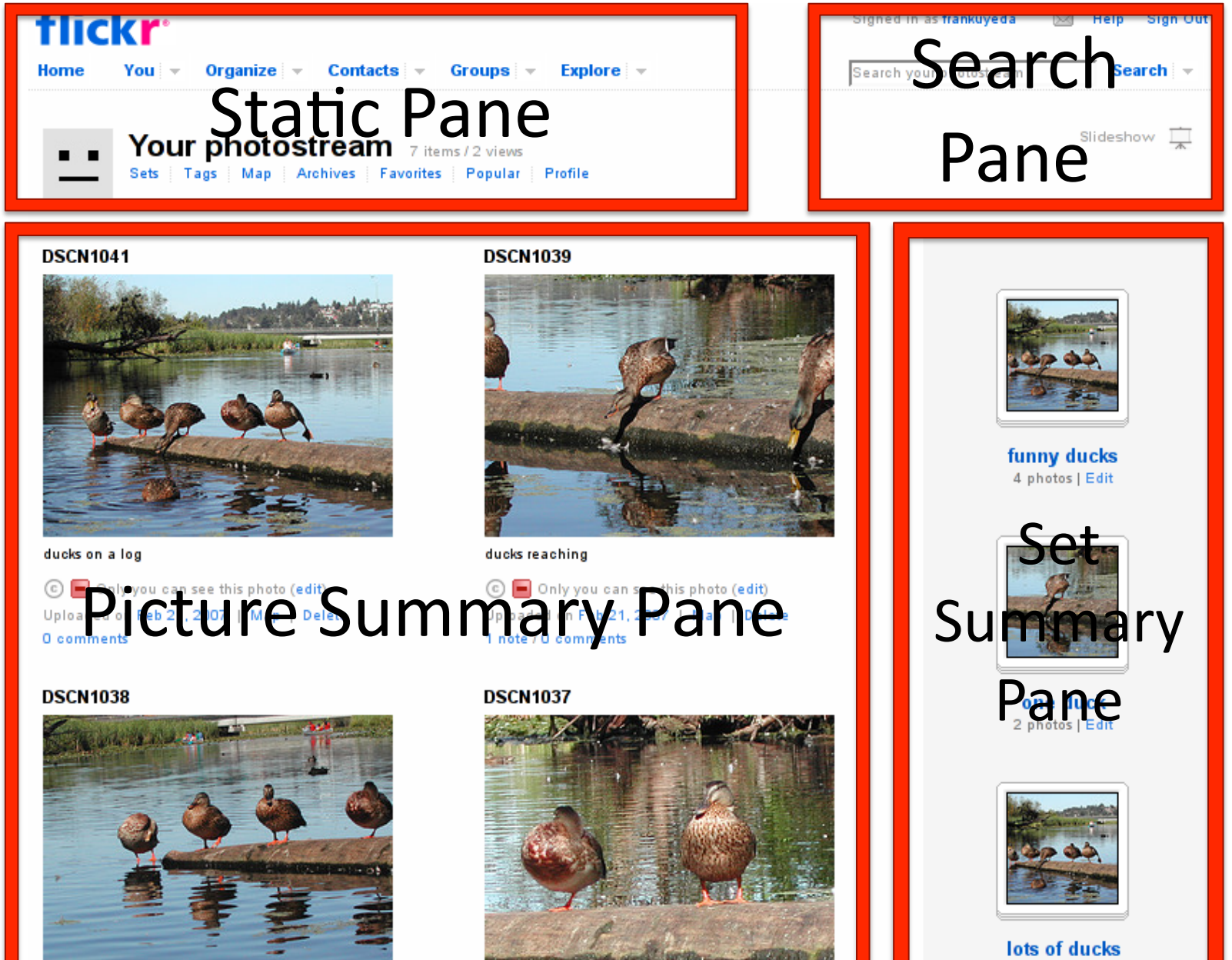
Running Example: Flickr.com

What is 

?

- Community photo-sharing website
- Users associate with each other
- Images organized using:
  - Keyword tags
  - User “photo sets”
  - Group “pools”

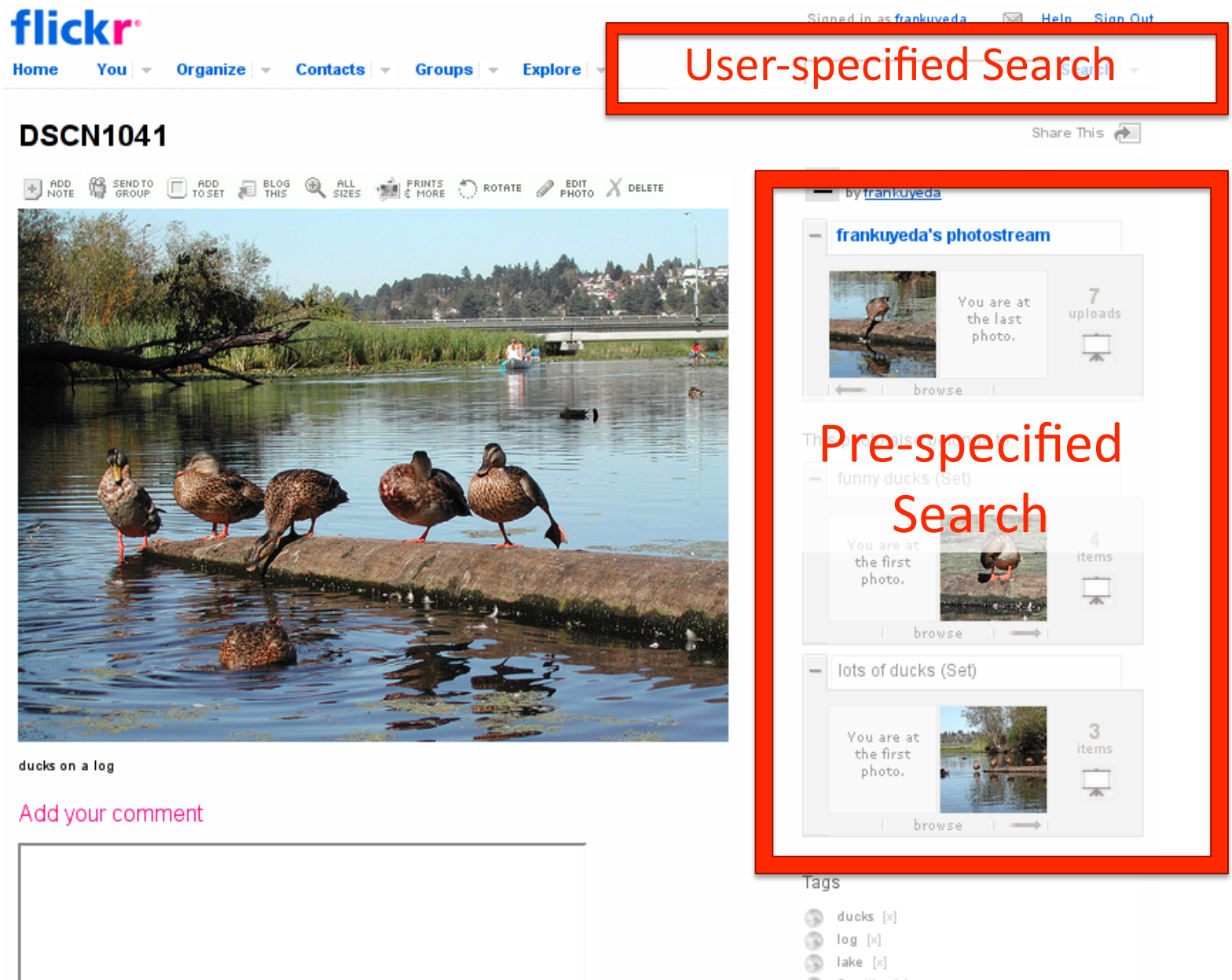
Insight 1: Page Layout Model



Formalizing Insight 1

- A page is composed of one or more panes
  - Panes are populated by embedded searches
- Pane
  - A region within the Page
  - Handles the input and output of a particular data collection, or displays static content
  - Defined once, and reused across many pages
  - Pane aesthetics customized with CSS

Insight 2: Navigation is Search



Formalizing Insight 2

- Navigation associates clicking on a data object to a page and search parameters.
  - Pages embed searches.
  - the “linkto” keyword provides parameters to searches.

- Syntax:

  [object in pane] -> linkto [page]([params]);

- Example: Clicking on a user’s name displays all pictures owned by that user.

  user -> linkto all_users_pictures(user);

Insight 3: Graph-based Search Model

Hierarchy

Attribute Filter

Ordering

Formalizing Insight 3

SELECT <collection>
[FROM <structural relation>]
[WHERE <filter condition> …]
[ORDER BY <ranking function> …]

- Structural relations:
  - Graphs: neighbor,
  - Tree: subtree, parent, children
- Filter conditions:
  - matches, contains, greater than, between, within distance, tagged by
- Ranking:
  - Combination of attributes or graph properties (e.g. node degree)

Outline

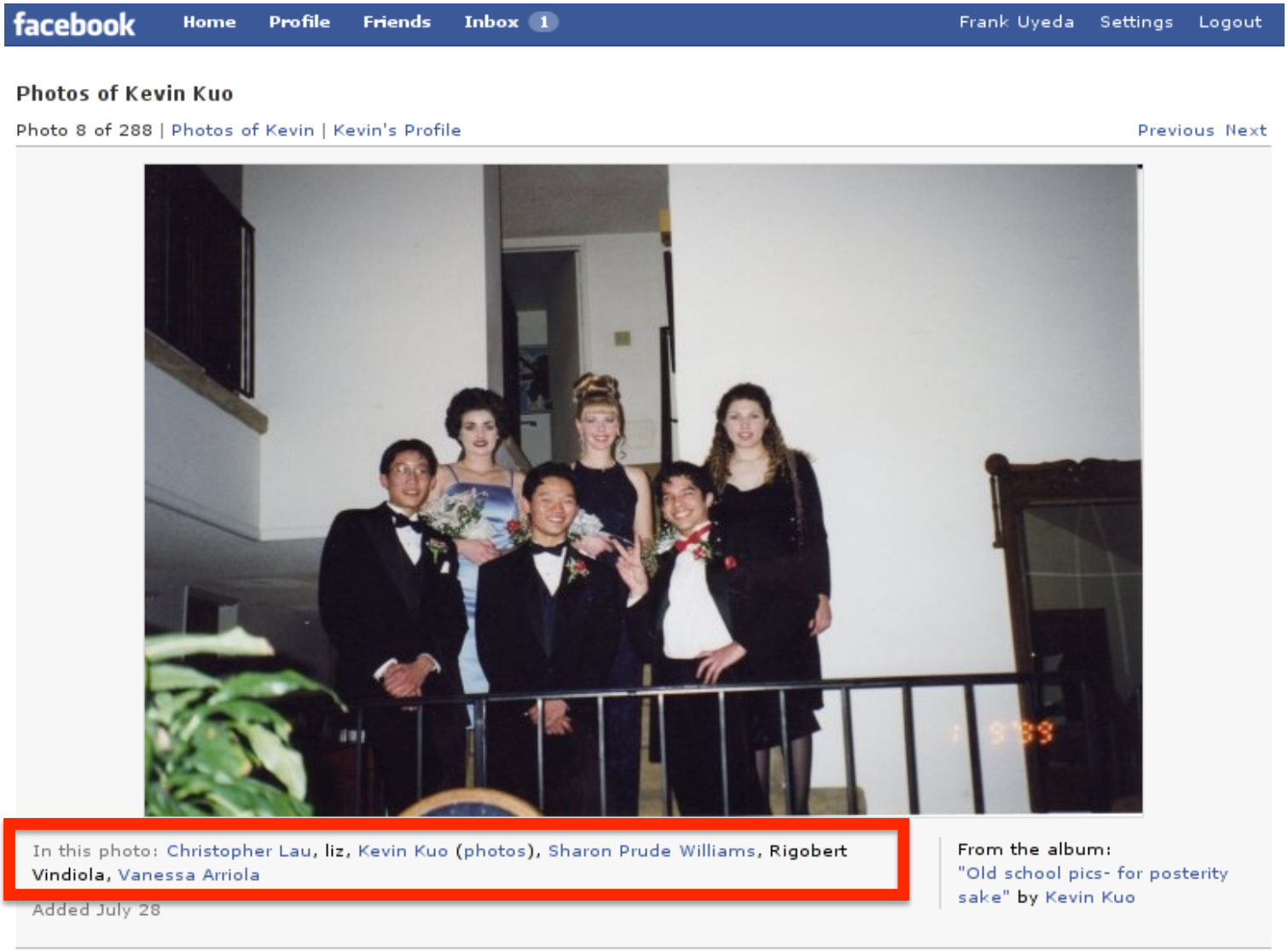
- Motivation
- Modeling
- Results

Abstraction Allow Powerful Modifications

- Claim: Small changes to the specification provide important features at low cost.
- Example:
  - Flickr tags photos with keywords
  - Facebook tags photos with Users

- How do we change our Flickr specification to incorporate User tagging?

Facebook User Tagging



Adding User Tags to Flickr

```
COMPOSITE Picture {
      IMAGE     pic;
      TEXT      pic_title;
      TEXT      pic_description;
} (taggable, taggable by USER, commentable);

PAGE pic_detail( Picture p ) {
      Detail(Picture) main : LOOKUP Picture p;
}

Detail Pane : Picture{
      _owner -> linkto user_profile( _owner );
      _tag -> linkto tag_result( _tag );
      _tag.USER -> linkto user_profile( _tag.USER );
       "add tag" -> linkto add_tag( this );
       "add to set" -> linkto add_to_set( this );
      "add to group" -> linkto add_to_group( this );
}
```

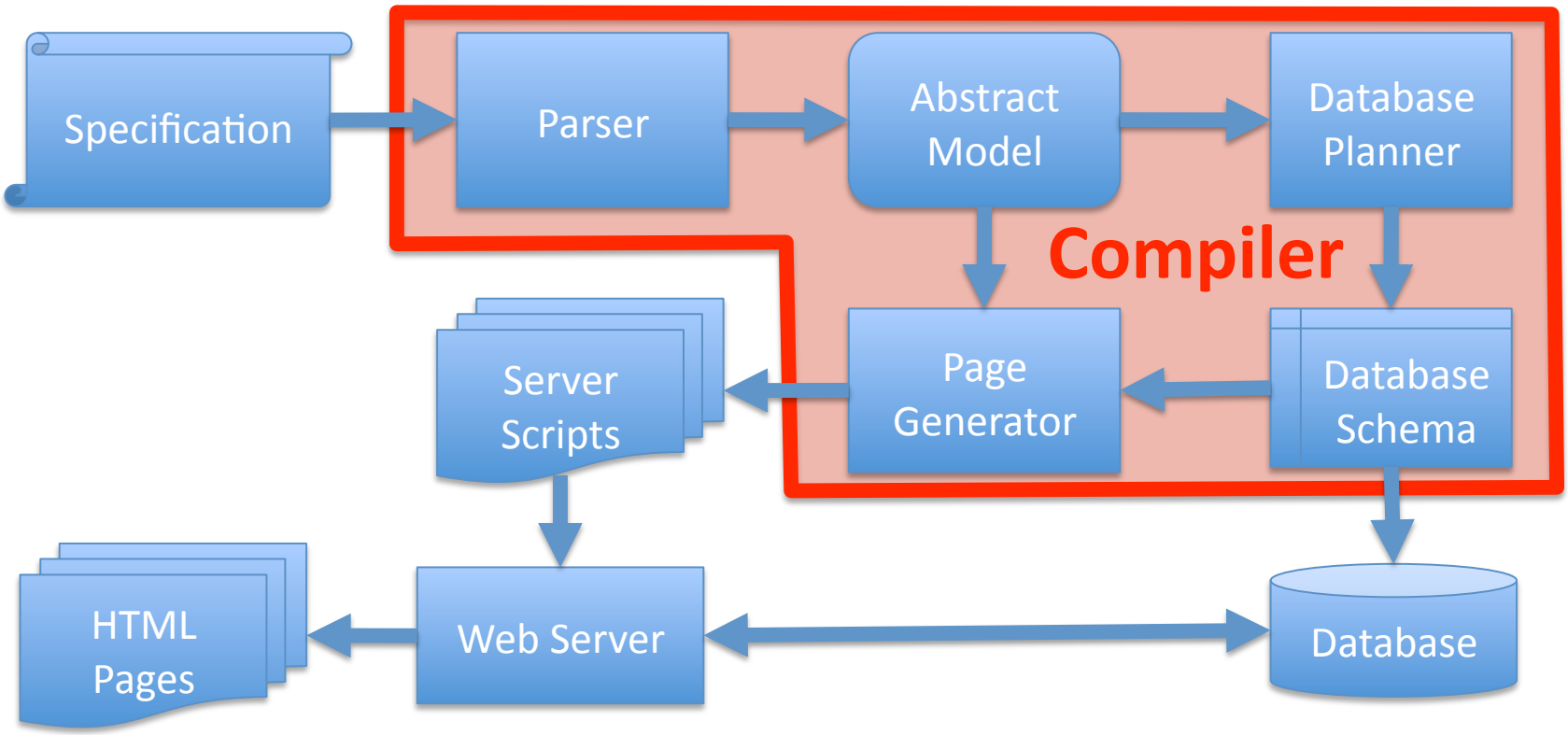
Compiled Flickr Page with User Tags

name: Bob
Tags: **value:** log **value:** ducks
Tags: **name:** Joe **name:** Bob
Comments:

User Tags

pic_description: i like ducks
add to group
add to set
add tag

GR Compiler Implementation



- Implemented in Java (~15K lines of code)
- Page generation in various languages.
  - Currently supports PHP

GR vs. Ruby on Rails: Abstraction and Performance

Implemented Flickr-like site using Rails plug-ins & GR

- Code Complexity
  - 50 lines of code across 19 files (vs. 180 lines in 1 file)
- Picture retrieval throughput:
  - Grassroots gives 2x max throughput.
    - Grassroots only generates necessary code
    - Ruby has large call tree.
- Tag search throughput – 20 most recent
  - 500:1 performance difference, favoring GrassRoots.
  - Suspect poor SQL queries & failure to parallelize.

Summary

- Need better tools for constructing social sites.
- Leverage the commonalities among sites.
- We provide abstractions to ease development.
  - Pages are composed of panes
  - All navigation is search
  - Graph-based search with attribute filtering
- Abstractions provide flexibility & opportunity for optimization.