

# *Architecture Extraction and Modeling for Object Oriented Sources*

Prof. Rushikesh K Joshi  
& Shakeb Sagheer  
IIT Bombay

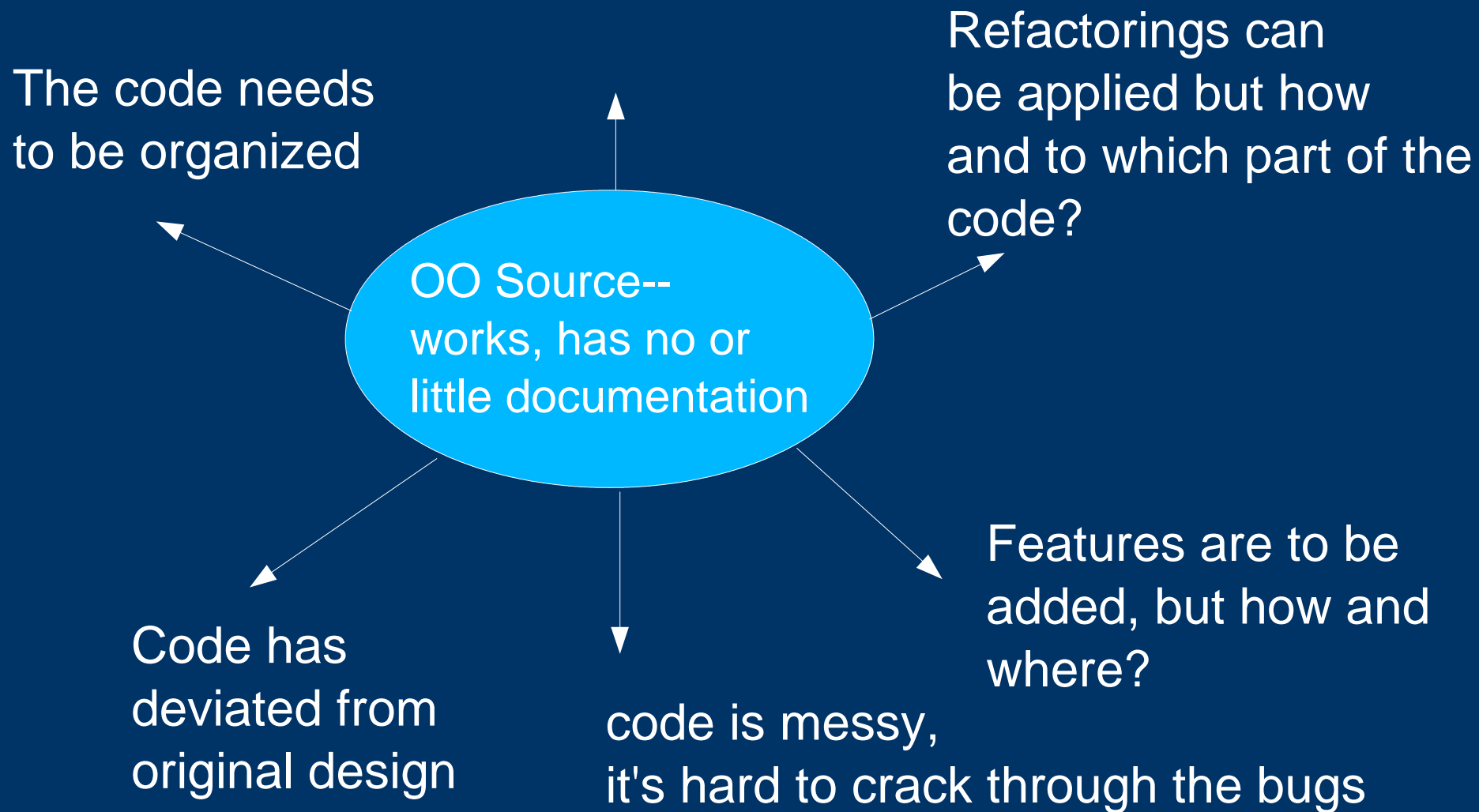
*Architects often come into an environment with very little documentation, and have to create architectural models from existing code before they can proceed with re-engineering of the application. In this talk, we will describe an ongoing work about techniques for building models from object oriented code with the help of a case study.*

*opengroup conference, Bandra, Feb. 27, 2007*

---

---

# *Why extract/recover architecture and models from sources?*



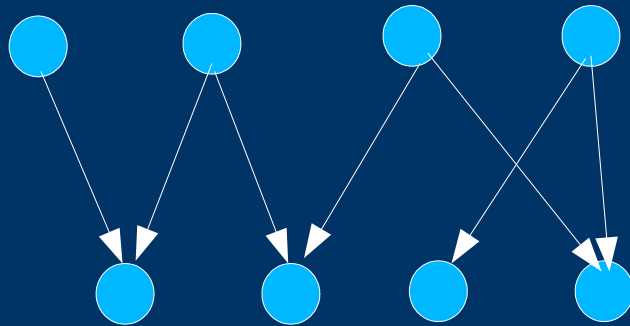
# Approach

- Individual Class Level ✓
- Class Interaction/Coupling Level ✓
- Class Relationships
- Class Groups/Architectural Styling ✓
- File Interactions ✓
- Objects/Components
- Processes
- Deployment/Networking



# Class level Models

- Cohesion Analysis
  - access graphs
  - concept analysis



Access graphs

vertices: functions and variables  
edges: R/W accesses, calls

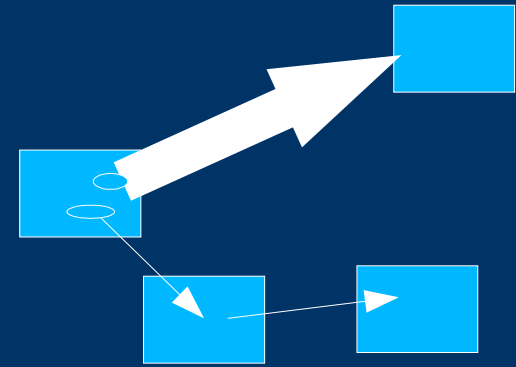
cohesion analysis can be performed.

\* Need to integrate cohesion results with coupling

---

---

# Class level Models



- Interaction/Coupling Analysis
  - Are some methods coupled heavily with other classes than with their container classes?
  - Coupling metrics can reveal the affinity
    - [CSMR 2006 paper]
  - challenges
    - Automatic refactoring: which is the right class for a given method? But during adjustments, abstractions should not be violated.
    - Microscopic analysis for identifying candidate members for restructuring



# *Class level models*

- Relationships
  - inheritance, aggregation, association, generalization, dependencies etc.
  - use existing tools to get a base diagram
    - refine it further
  - challenges
    - Semantics of relations are often not taken into account
      - e.g. how to infer aggregation? (part-whole semantics)
    - Multiplicity of association relation



# *Groups of Classes*

- Which classes together form a logical group?
- Knowledge of architectural styling
  - MVC, Layers, C/S, P/P FDP
- File groupings/packaging
- Design patterns
- Partalogy



# *(Source) File Level Interactions*

- What type of components contained in each file
  - What type of connectors/semantics of interactions among the components
  - types of source files: classes, jsp, js, html, ...
  - Member function calls
  - Object instantiation
  - Calls to servlets
  - JSP references..
- 
-



# *Executable representations of Architectural models*

- Component/connector paradigm
- Capture Architectural scenarios, events
- Timelines/Sequencing
- Kinds of connectors, first class connectors
  - A java+aspects based implementation is under development
- Ontology for semantics of architectural primitives



# *A Case Study: Java Pet Store*

- Java Pet Store 2.0 Reference Application is a sample J2EE application developed by Sun Microsystems.
  - Web application to model a pet store.
  - Uses Java Server Pages (JSP's) for client interactions and a back-end java functionality to serve requests.
  - Key design pattern used is Model-View-Controller (MVC) architecture.
- 
-

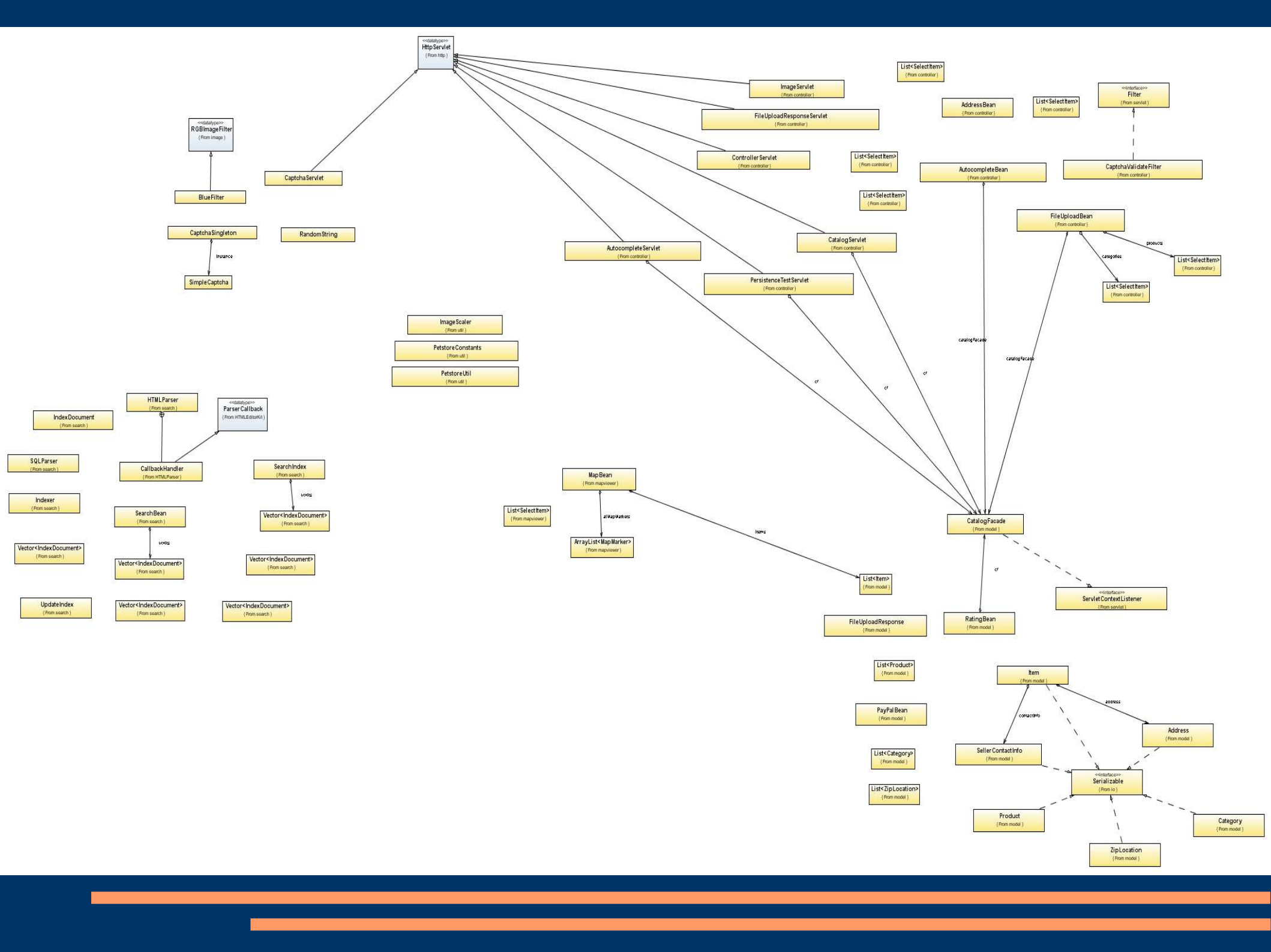
# *Java Pet Store Raw (automatically extracted) Class Diagram*

- Gives a static view of class level architecture
- Describes system classes, their attributes and the relationship between classes
- Class diagram given here was produced using Sun Microsystems Java Studio

Enterprise 8 SDK.

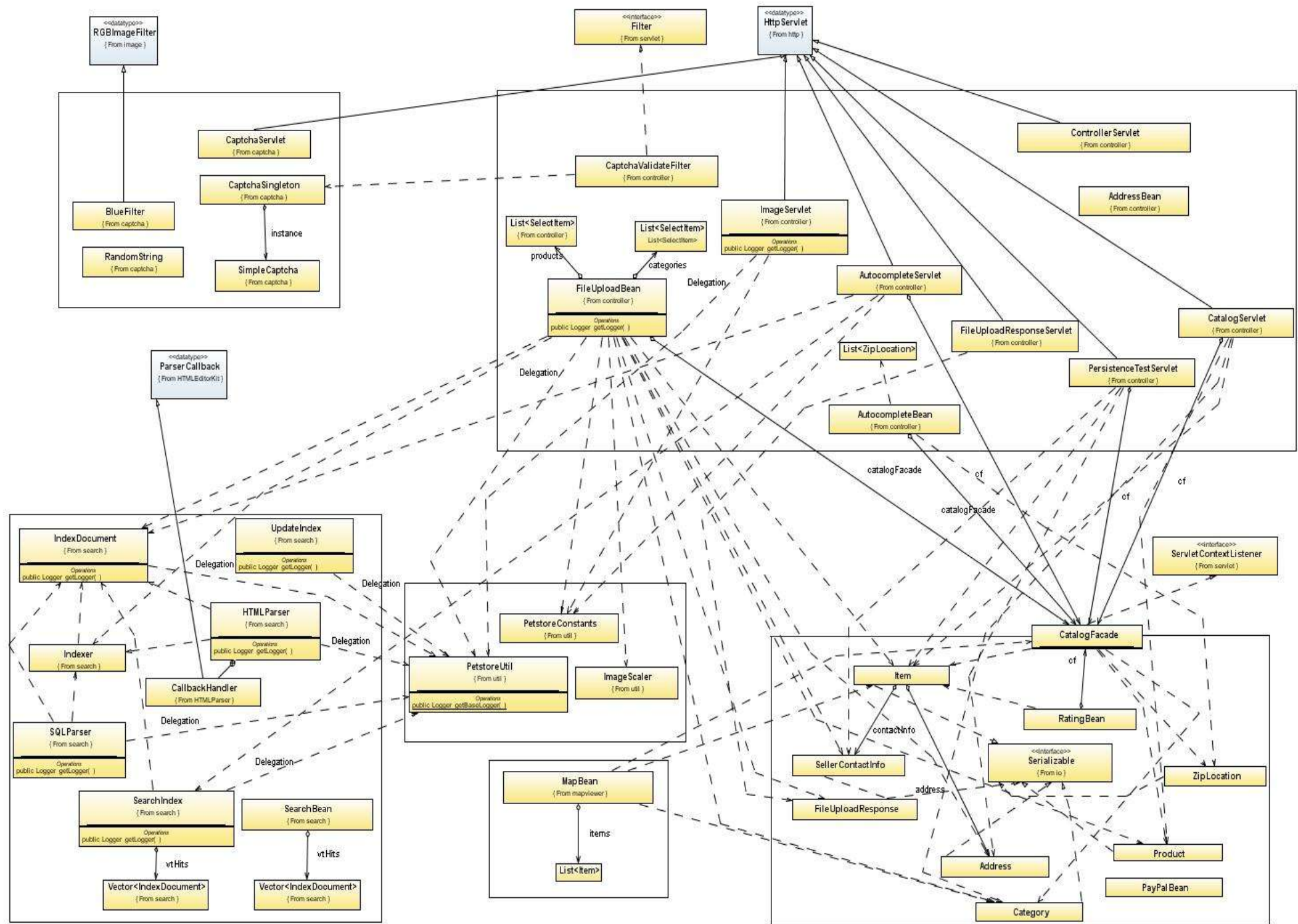
---

---



# *Java Pet Store: Class Groups based on packages*

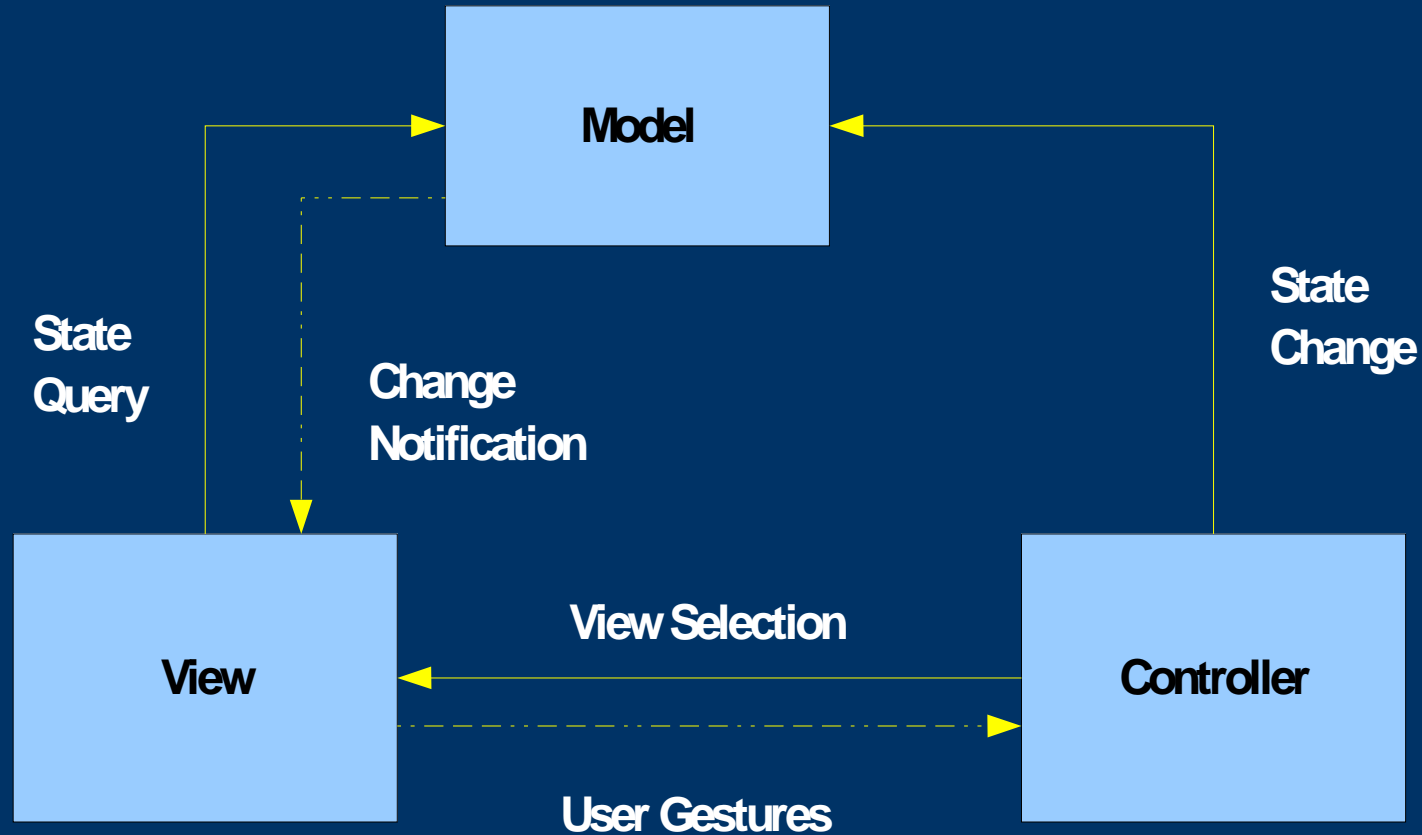
- Components manually grouped to show existing packages.
  - Rectangles drawn to denote package boundaries.
  - Pre-existing 'model' and 'controller' packages point towards MVC modeling.
  - Add dependencies (non association/aggregation)
  - Update with Aggregation Analysis, ..
  - May still be incomplete in terms of full architectural styling (e.g. jsp files don't get included)
- 
-



# *MVC Architecture used in Petstore*

- Application divided into three layers: Model, View and Controller
  - View
    - User Interface
    - HTML pages, JSP's.
  - Model
    - Represents the structure of data
    - Performs application-specific operations on data
  - Controller
    - Translates user actions into application function calls on model
    - Selects appropriate view
- 
-

# MVC Architecture





# *Java Pet Store MVC Architecture*

- Files divided among View, Model, Controller and Utility components.
  - View consists of the JSP's.
  - Model and Controller have same contents as 'model' and 'controller' packages resp.
  - Model uses a facade design pattern
    - CatalogFacade.java acts as facade while handling requests
- 
-

# *Java Pet Store MVC Architecture*

- Utility contains the remaining classes.
- Classification of files into Model-View-Controller components gives an idea about the functionality
  - but not about interaction semantics



# *File/Package Level Interaction Architecture (FLIA/PLIA)*

- Gives a view of how files/packages are related and how source components in them interact.
  - A link from file/package A to file/package B indicates message/data passing from A to B.
  - Types of data interchange between files (from the point of view of Java Pet Store):
    - Object invocations
    - Servlet Interaction
    - JSP references
- 
-

## *FLIA : Type of links*

- Object invocations : Using features of classes by instantiating objects.
- Servlet Interaction : Sending data to servlets and receiving response.
- jsp references: Passing requests/parameters to JSPs or HTML files.



## *FLIA : Link Parameters*

- Set of parameters associated with each type of link
  - Provide information about the degree of association between the two connected components.
- Each link labelled with a tuple of values for these parameters.



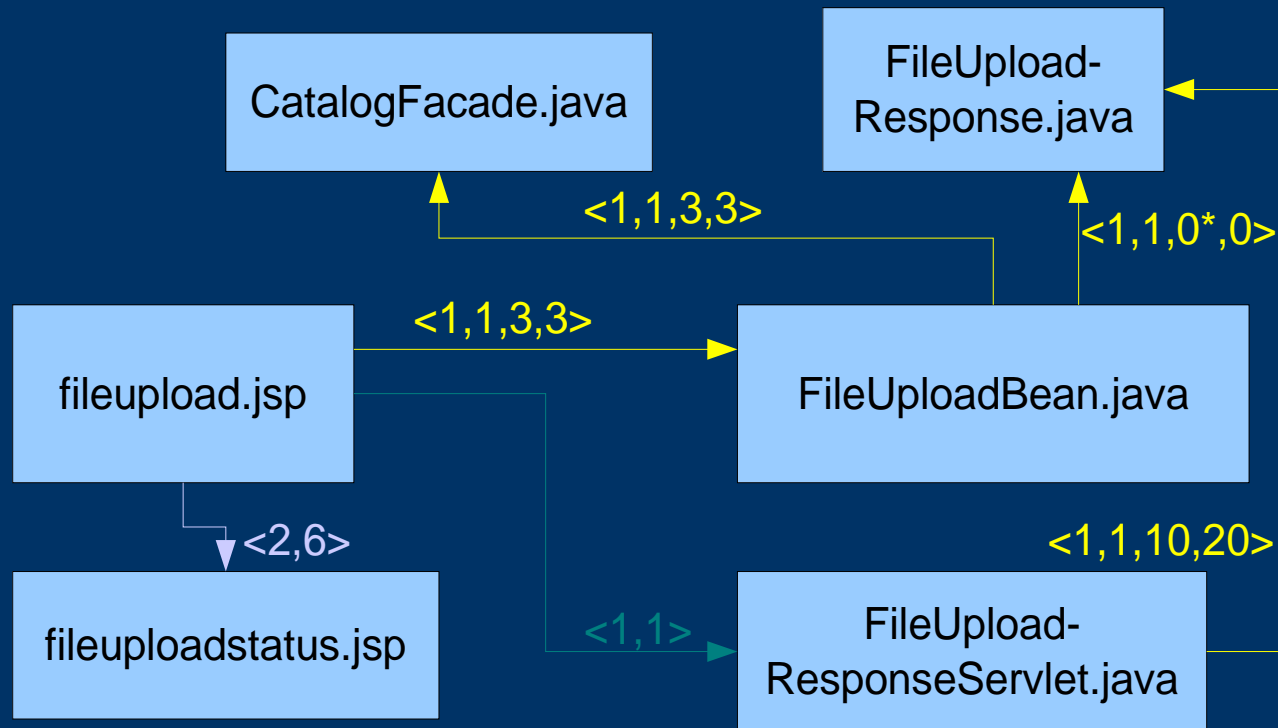
# *FLIA : Link Parameters*

- Object Invocation : <no\_of\_classes, no\_of\_objects, no\_of\_features\_accessed, no\_of\_times\_features\_accessed>
    - no\_of\_classes : Number of classes of the target file instantiated.
    - no\_of\_objects : Number of objects of the target classes initialized.
    - no\_of\_features\_accessed : Number of features of the target classes accessed.
    - no\_of\_times\_features\_accessed : Number of times features of the target class accessed.
- 
-

## *FLIA : Link Parameters*

- Servlet Interaction : <no\_of\_requests, no\_of\_invocations>
    - no\_of\_request : Number of times a request was send to the servlet
    - no\_of\_invocations : Number of methods of the target invoked.
  - JSP References: <no\_of\_times, no\_of\_parameters\_passed>
    - no\_of\_times : Number of times the parameters are passed.
    - no\_of\_parameters\_passed : Number of parameters passed to the jsp/html.
- 
-

# File Level Interaction Scenario in Java Pet Store



—▶ Object Invocations

—▶ Servlet Interaction

—▶ JSP References



# Summary

- An approach towards extraction of models from sources
- Mixed approach:
  - use of analysis techniques
  - use of manual intervention/available knowledge
- Multifaceted analysis

Acknowledgements: Part of the work is being carried out by Shakeb Sagheer towards his B.Tech Project, which is funded through a grant from IBM CAS.

---

---