Repository-Based J2EE Development

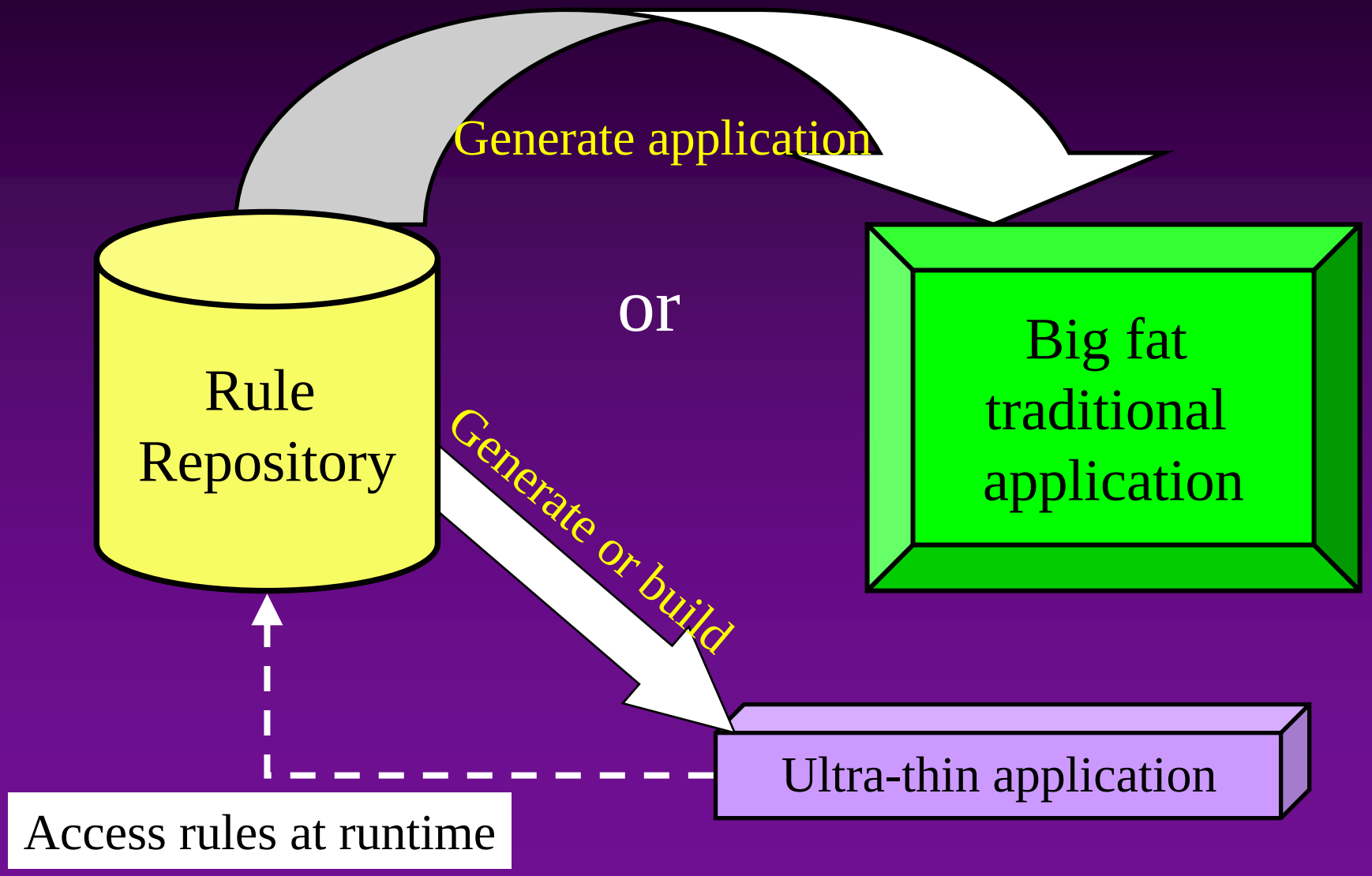
Dr. Paul Dorsey
Dulcian, Inc.
www.dulcian.com

NYOUG - March 9, 2006

The Problem

- Current web applications development product environment is not ideal:
  - Many components
  - Difficult to learn
  - Resulting systems are less robust than old client/server systems.
- J2EE environment is in constant flux:
  - 2 years ago – JavaServer pages (JSPs)
  - Today – JSP/Struts
  - Next year – JavaServer Faces (JSFs)?
  - What about EJB3, BPEL, Web Services?

The Solution



The Challenges

- Hard
  - Repository/grammar is hard to design.
  - Figure out what to generate.
  - Determine how the application will look.
- Easy
  - Create the generator

Advantages of Repository-Based Development

- 99% generated stuff
- Very rapid development
- Easy to port systems
  - to new user interface technology
  - to new user interface standard
  - to new database technology
  - to new area of system
- Very easy to add or change rules
  - Text repository is easy to search.
- Self-documenting
  - English translation of rules
  - Report on repository

Specifying the system

"The articulation of the rules is independent of the implementation of the rules."

System Logical Specification

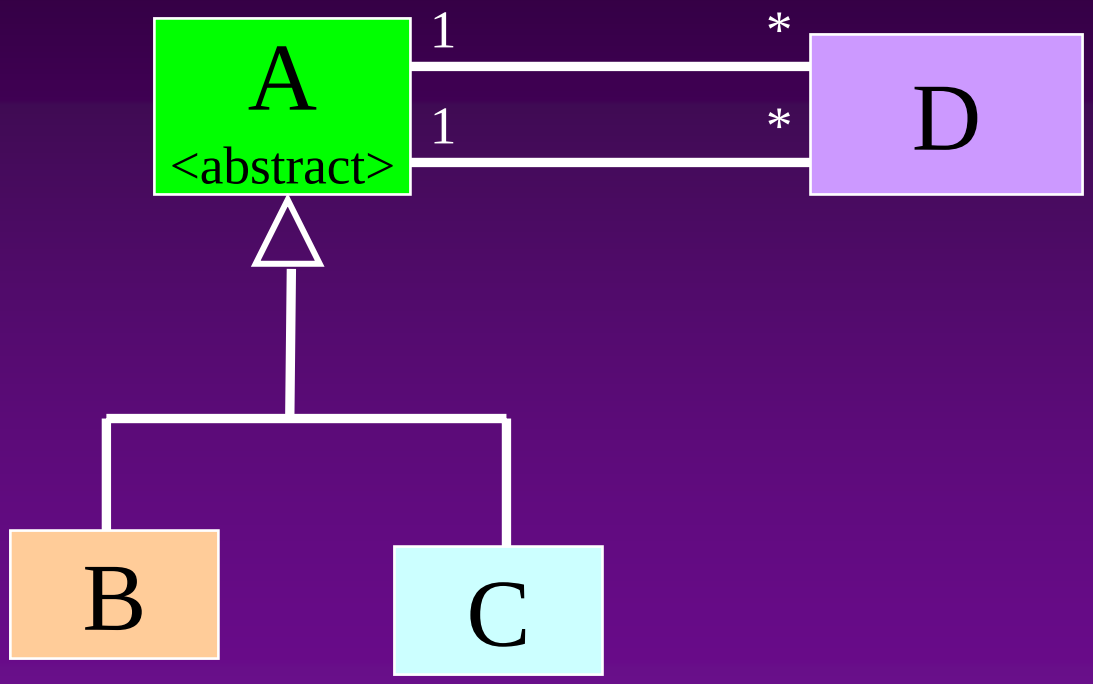
- Object
  - Structure
  - Process
  - Data validation
- User Interface
  - Model
    - Structure
    - Binding
  - View
    - Structure
    - Logical rules
  - Controller
    - Logical page flow

System Physical Specification

- Database
  - Tables
  - Views
  - Packages
- User Interface
  - Model
  - View
  - Controller

Logical vs. Physical

- Limit specification at physical level to the essentials
  - Not table/column names



  - In D, B_OID must appear twice. One must be renamed
  - Don't maintain two models!

Specifying the rules

- System
  - Logical
    - Object
      - Structure
      - Process
      - Data validation
    - User Interface
      - Model
        - Structure
        - Binding
      - View
        - Structure
        - Logical rules
      - Controller
        - Logical page flow

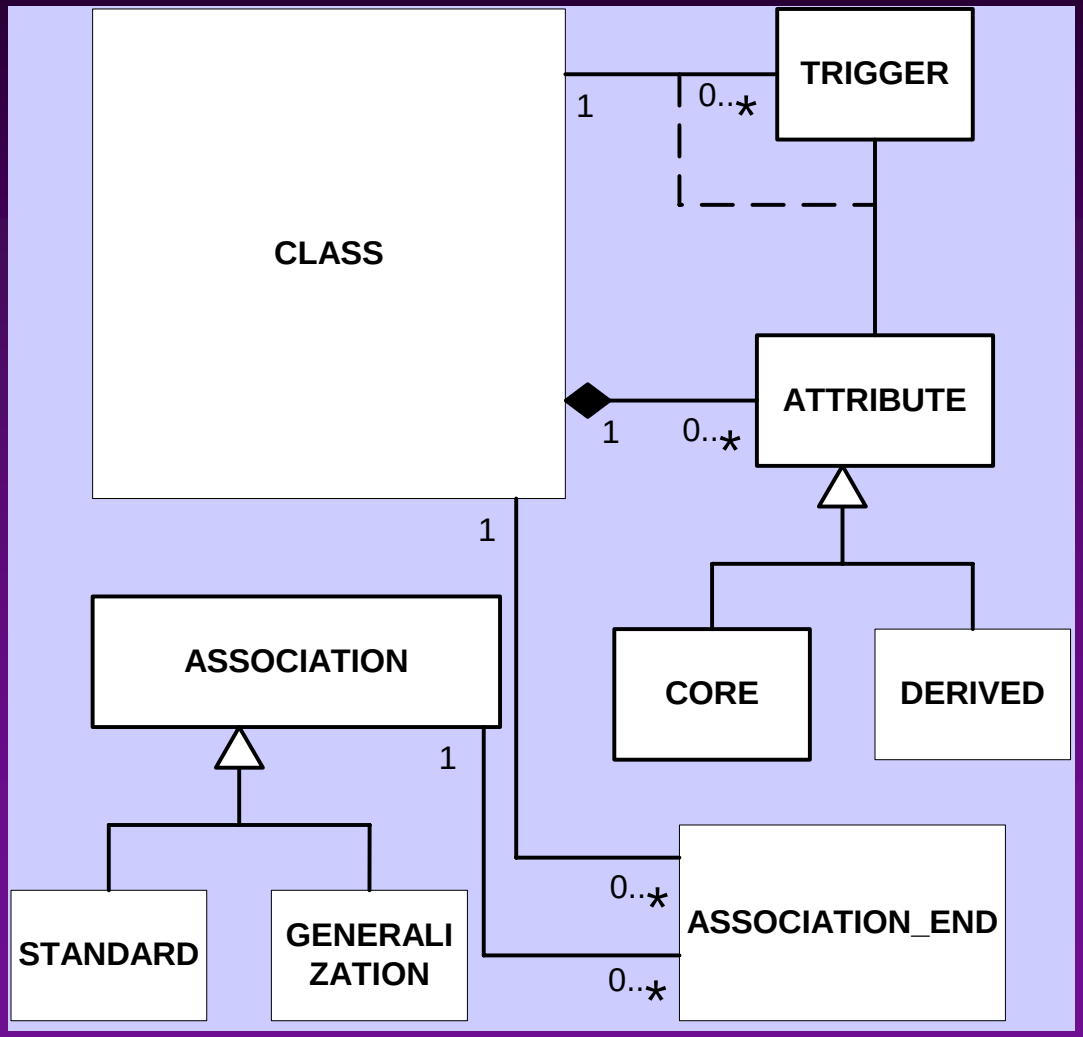
- Specify everything you can at the object level
  - Object level = 80% of the rules
  - UI = 20% of the rules

Object Rules - Structure

- Not just an ERD
- Derived Attributes
- Keywords
  - History
  - Audit
- Logical triggers
  - Post-creation
  - Pre-update

    ⋮
- Inheritance

Structural Rules Repository Meta-Model

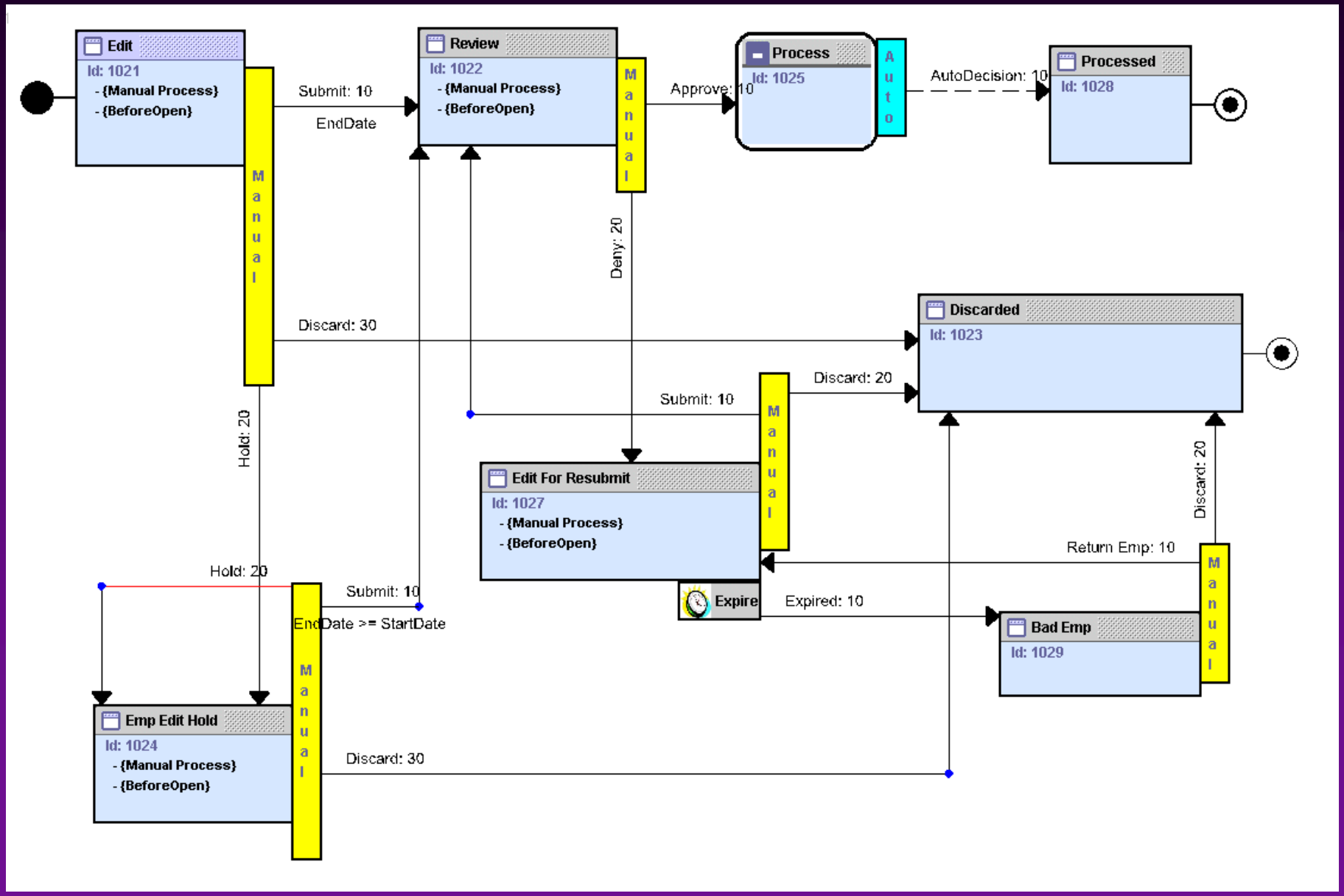


Object Rules - Process

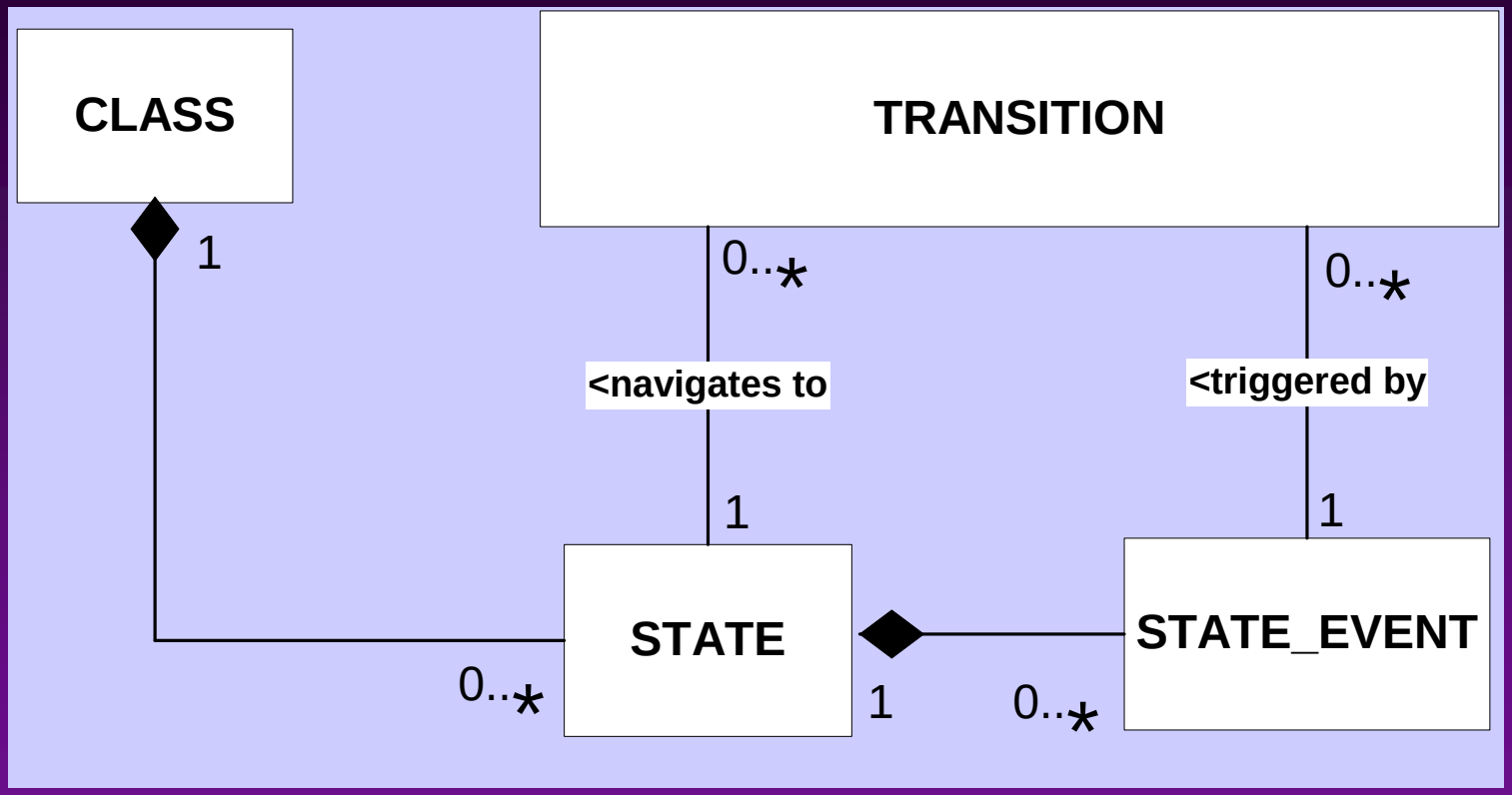
- Not declarative
  - Too many rules
- Not STE or DFD
  - Too many boxes
- Complex state
  - State and state events
- State events (like a database trigger)
  - On-set
  - Expire
  - Manual Process
  - Manual Decision

    •
    •
    •
  - Keeps number of states small

Sample Process Flow



Process Flow Repository



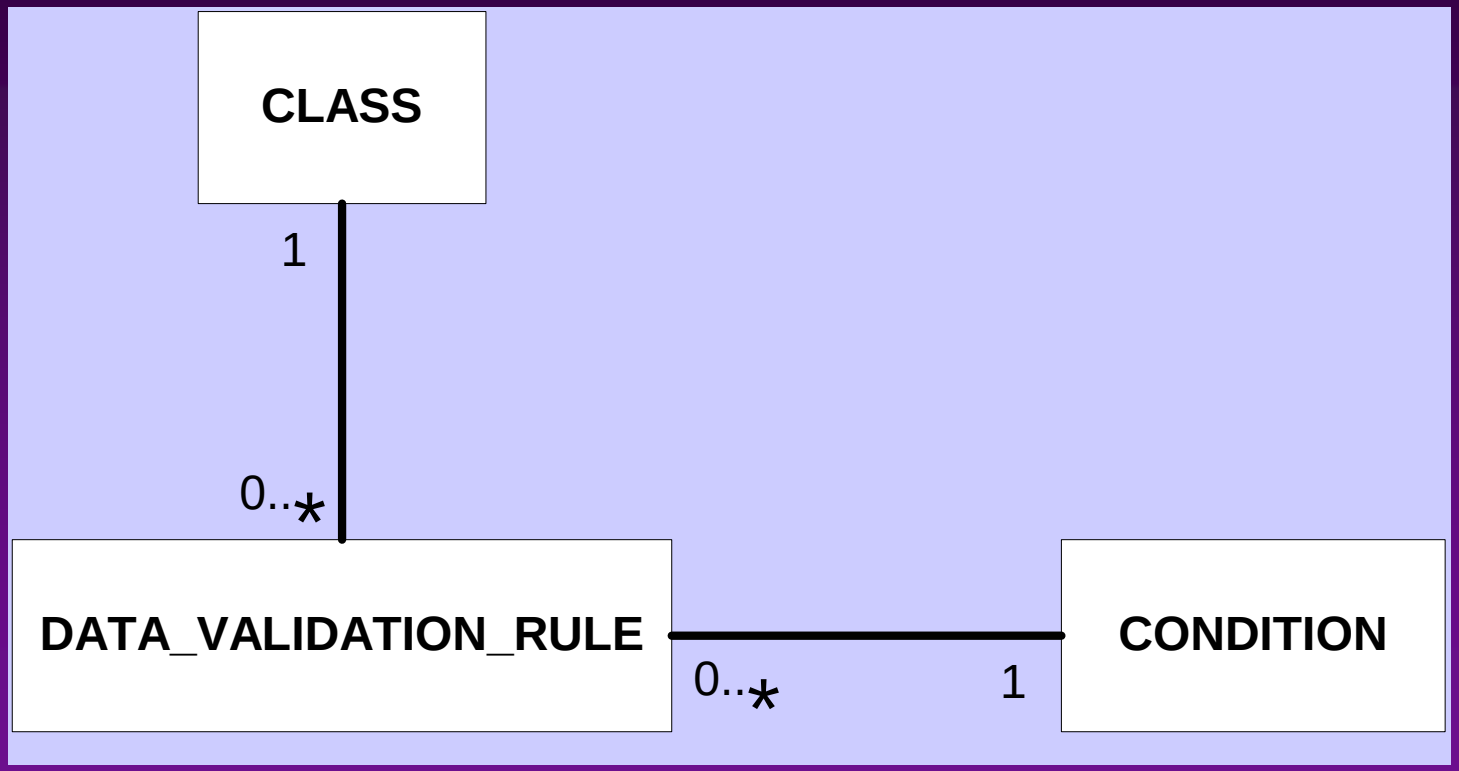
Data Validation Rules

- These rules may always need to be enforced or only contingently enforced based upon some condition or the state of the object.
  - May only require looking at the object being validated or accessing objects in other classes.
  - Rule failure may only trigger a user warning or may prevent data modification entirely.
- The difficulty is creating a grammar to help specify the rules.
  - The solution is to place the rules at the object level but support an Object Constraint Language (OCL)-like syntax that allows you to validate across classes.

```
:_child.emp.count >= 1
```

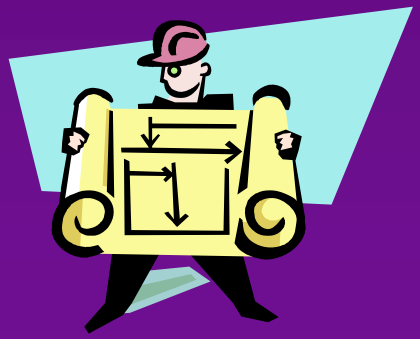
- This grammar can be easily extended to support 99% of all rules encountered.
- Validation rules are often only contingently required.
  - Can be invoked at the object state level and may be contingently executed based upon some condition.

Data Validation Repository



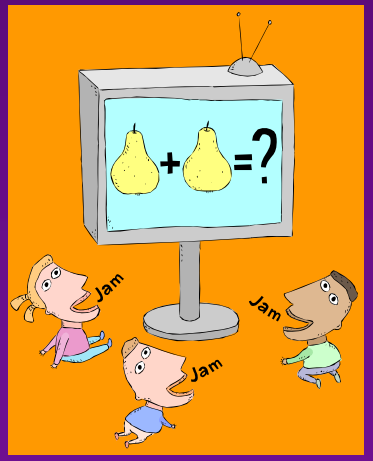
User Interface Rule Types

- Once object rules are collected, some additional rules are required to specify the user interface.
- Use modified version of the Model-View-Controller (MVC) architecture.
- The goal is to define the application independent of any technology or implementation considerations.



Object rules are not enough

- You can generate an application...
  - ...but it won't be usable
- At least minimal additional information is needed.
- Hence.....User interface (UI) rules



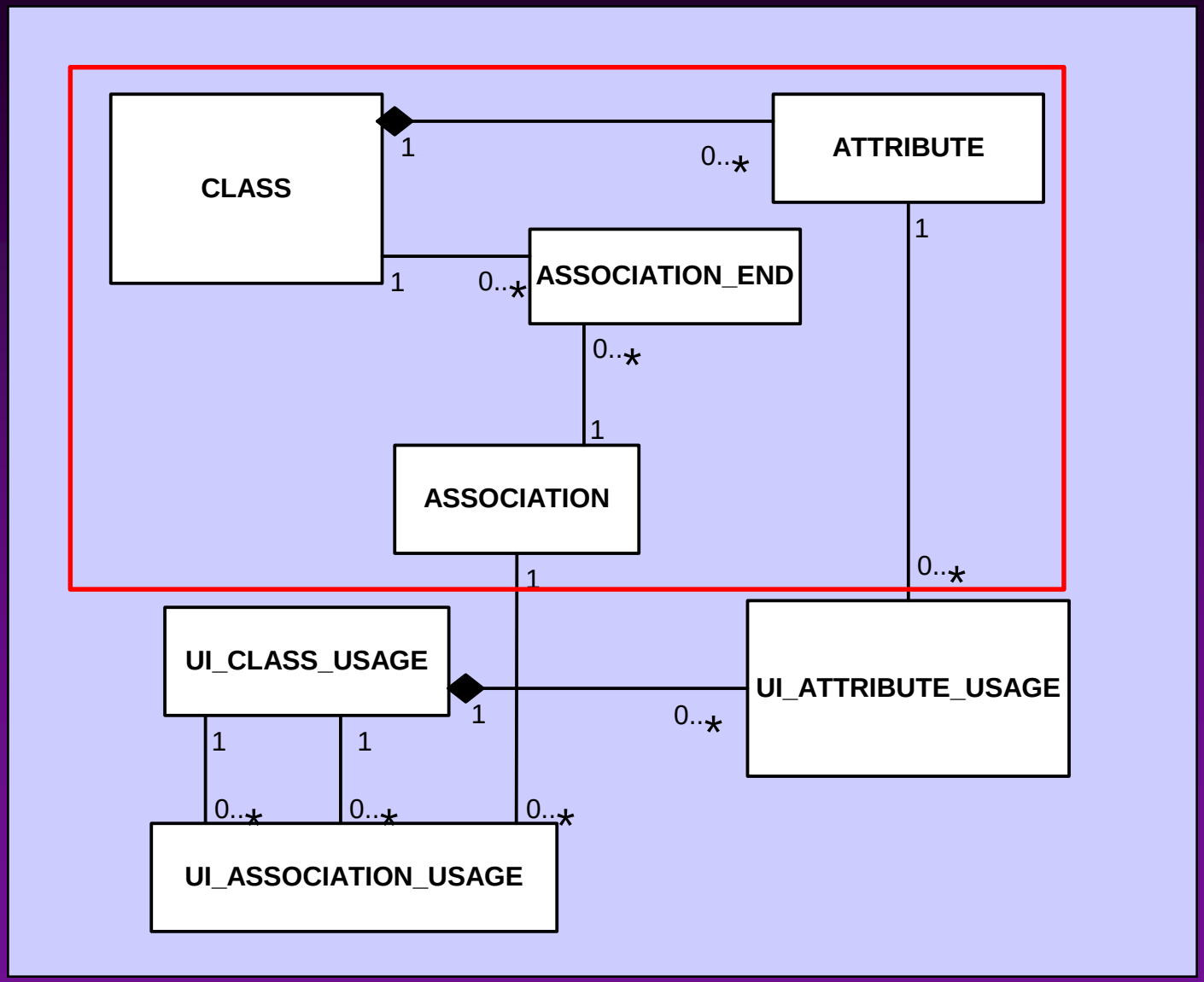
UI Rules

- UI
  - Model
    - Structure & binding
      - Just point to existing classes, attributes, associations
    - Requires parameterized views in the object layer
  - View
    - Structure
      - Items bind to model
      - Items sit in groups
      - Logic – Event-Condition-Action (ECA)
  - View Logic
    - Event-Condition-Action
    - All rules in the database
    - Access at runtime
  - Controller
    - Logical page flow

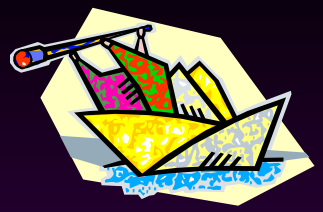
Model Layer

- Model portion of the logical UI rules is not difficult to specify.
  - Classes, attributes and associations have already been defined at the object level.
- Only requirement at the UI level is to select a subset of objects from the object level for use in the UI specification.
- Approach runs counter to the way in which most systems are built.
- Most tools specializing in model development support very sophisticated object specification in the model portion of the UI.
- Approach does not preclude “thick” UI model level for implementation
  - Structure of the UI model should properly be defined at the object level.
- Using this approach:
  - Structural rules at the object level will be quite sophisticated
  - Requires not only standard views, but also views that are dynamically altered or generated based on the values of some passed parameters.
  - UI model specification merely needs to point to existing structural object specifications.

UI Specification Model

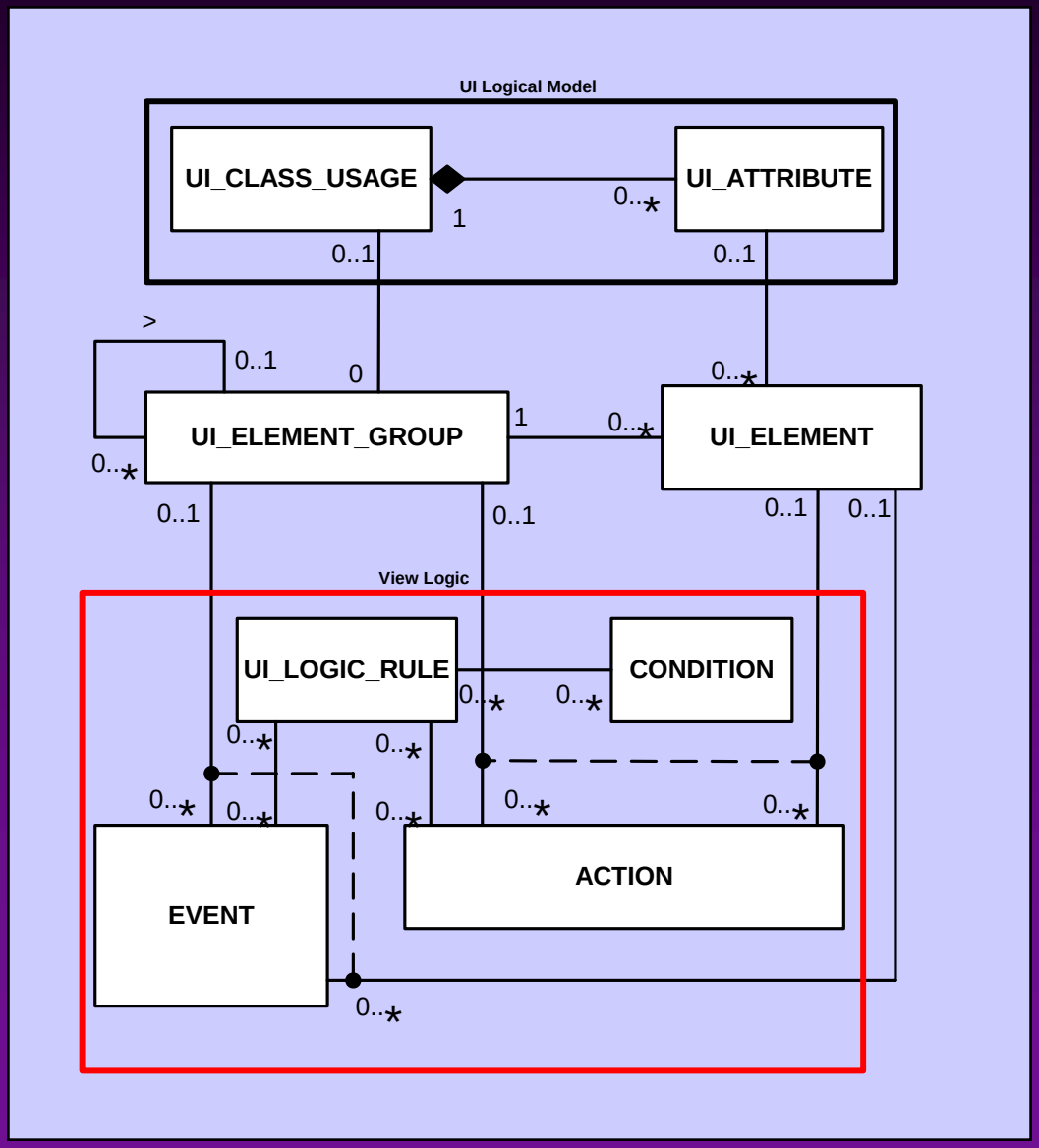


View Layer



- Rules in the view layer of the logical UI are divided:
  - Structural (what are the elements and how are they grouped)
  - Logical (what happens when a screen opens, or a button is pressed)
  - Presentation (how and where the elements are displayed).
- The view layer structural rules are very simple.
  - Define UI elements (fields, buttons, etc.) and how they are grouped and bound to the UI model.
- View layer logical rules are quite complex.
  - Full Event-Condition-Action (ECA) architecture needed to define what happens when events (button press, open an application, etc.) occur.
  - Conditions, actions, and events are defined as reusable objects.

ECA Architecture Model

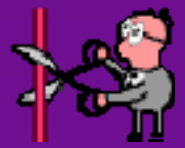


UI Controller

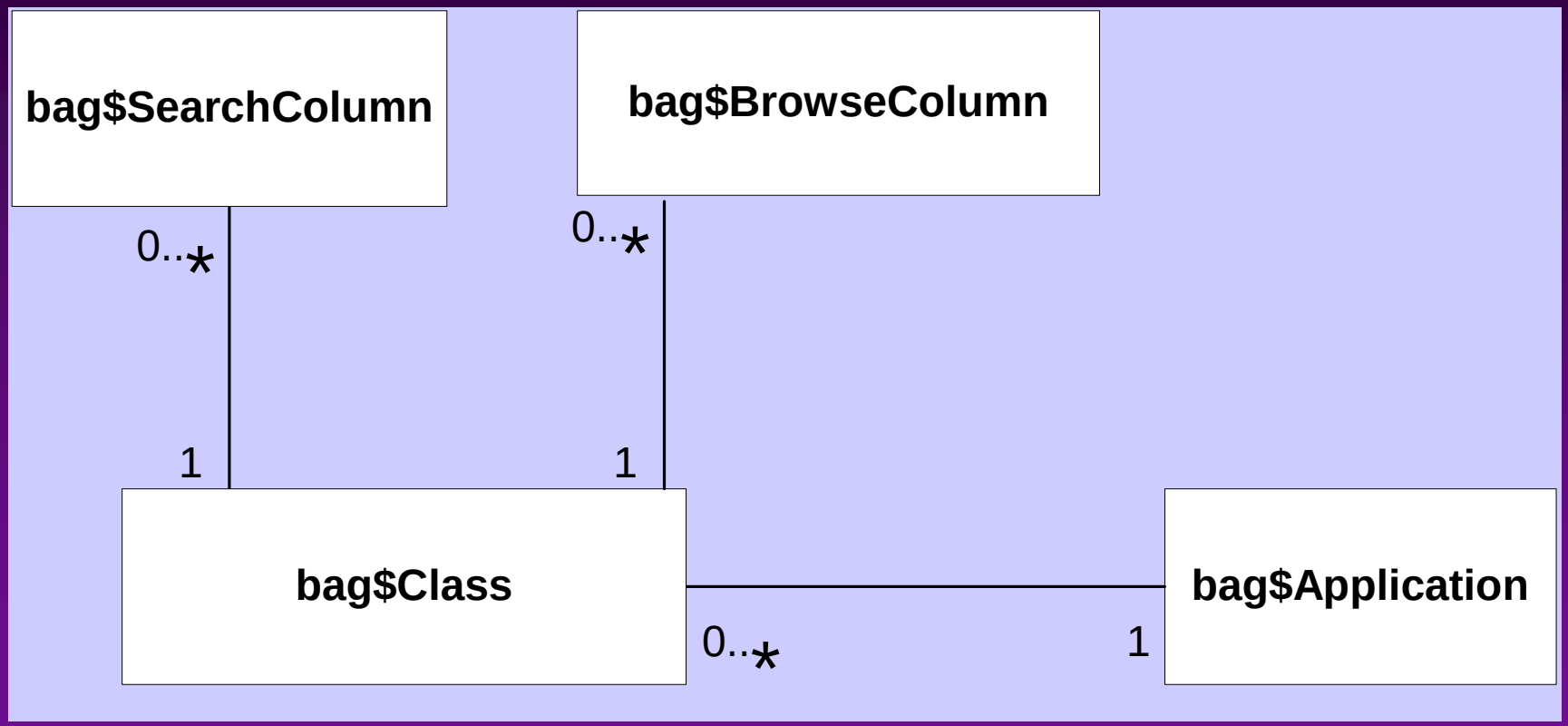
- Logical Page Flow Diagram
- How pages navigate
- What happens between pages
- Same STE as for objects

User Interface Shortcuts

- Standard UI structures should not have to be built over and over again.
  - You can define system elements such as Browse screens that only require a few elements to be specified.
  - All of the logical specification will then be generated automatically.
- Example of elements that must be specified in the Browse screen:
  - Fields that you want to query by (and how they appear)
  - Fields in the display block (and how they will appear)
- The rest is automatic.
- This approach allows you to build the user interface very quickly.
- Specify the browse screen for a particular class, point and click the desired Query By attributes and display and out pops the application.



Browse Screen Model



Generation Decisions

- Thick database
- Application Development Framework – Business Components (ADF BC)
- Limit UI design options
- Use JSP/Struts
- Avoid post-generation modification

Sample Browse Page

Survey - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://192.168.0.104:8988/surveyWS2-context-root/srsurvey/browseData.do

Survey

Search Criteria

Survey OID

Name_TX

Default
Case Sensitive
Exact Match

Search

Create Page [1] Records per page : 10 Set Total number of records : 5

Results

	Survey OID	Survey	Instruction_TX		
◉	7258518	Music Survey		Edit	Remove
○	7258187	Soccer Fan Survey	About likes and dislikes of fans	Edit	Remove
○	7258199	PE Maternal		Edit	Remove
○	7259336	Test Survey		Edit	Remove
○	7259436	yalim survey		Edit	Remove

Go to Associated

Question Group Usage (Question Group used in Survey)

Main page

▶Record ▶Layout Group ▶Multi Choice ▶Multi Record ▶Question Drill To ▶Question Group ▶Question Group Usage ▶Question Usage ▶Record Value ▶Single Response

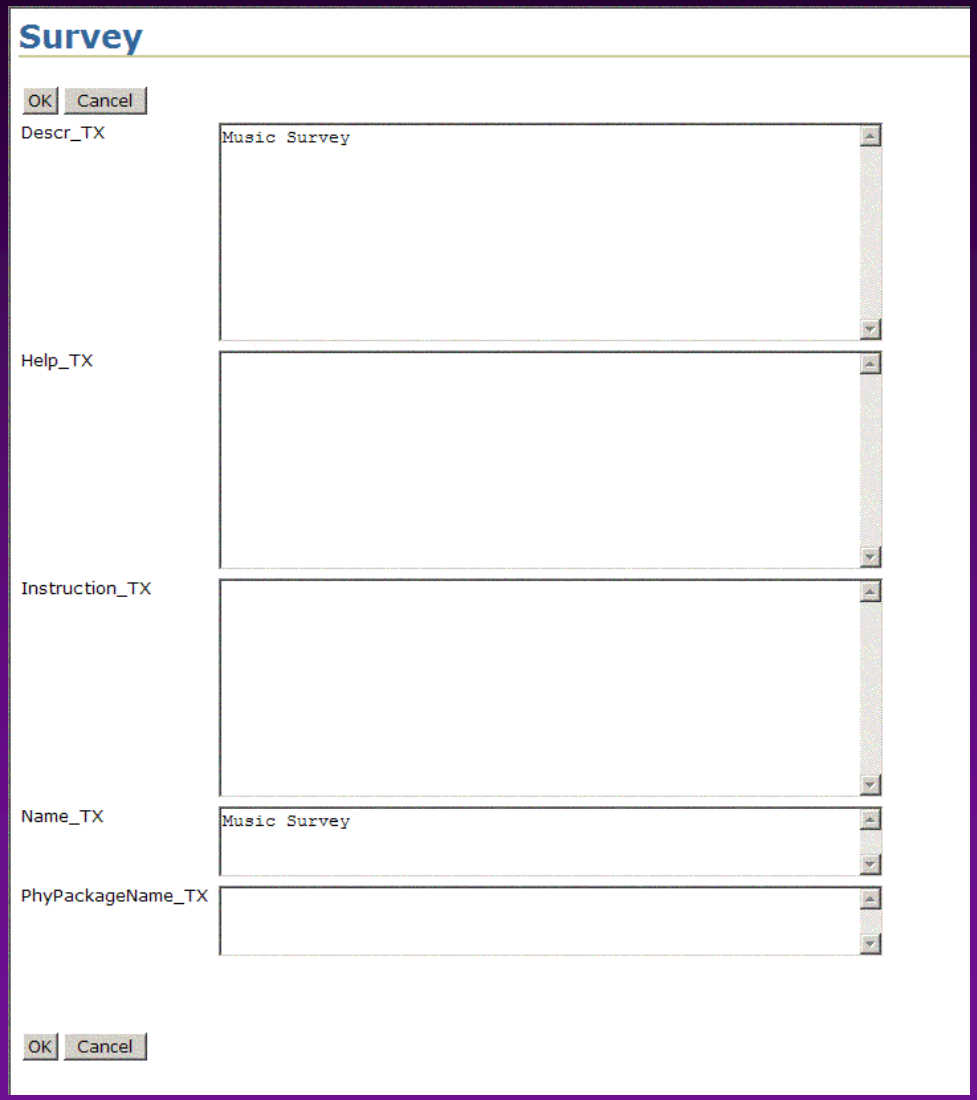
▶Survey ▶Boolean ▶User

Browse Page Sections

- Search Criteria:
  - Search fields generated according to the domain of the column with which they are associated.
- Results:
  - Displays the query results.
  - Includes a navigation bar to quickly locate the desired rows.
- Associations:
  - This section shows links to the master and detail classes.
  - Generated using the associations specified in the UML Data Model.
- Menu:
  - Includes links to all classes in the application.

Sample Edit Screen

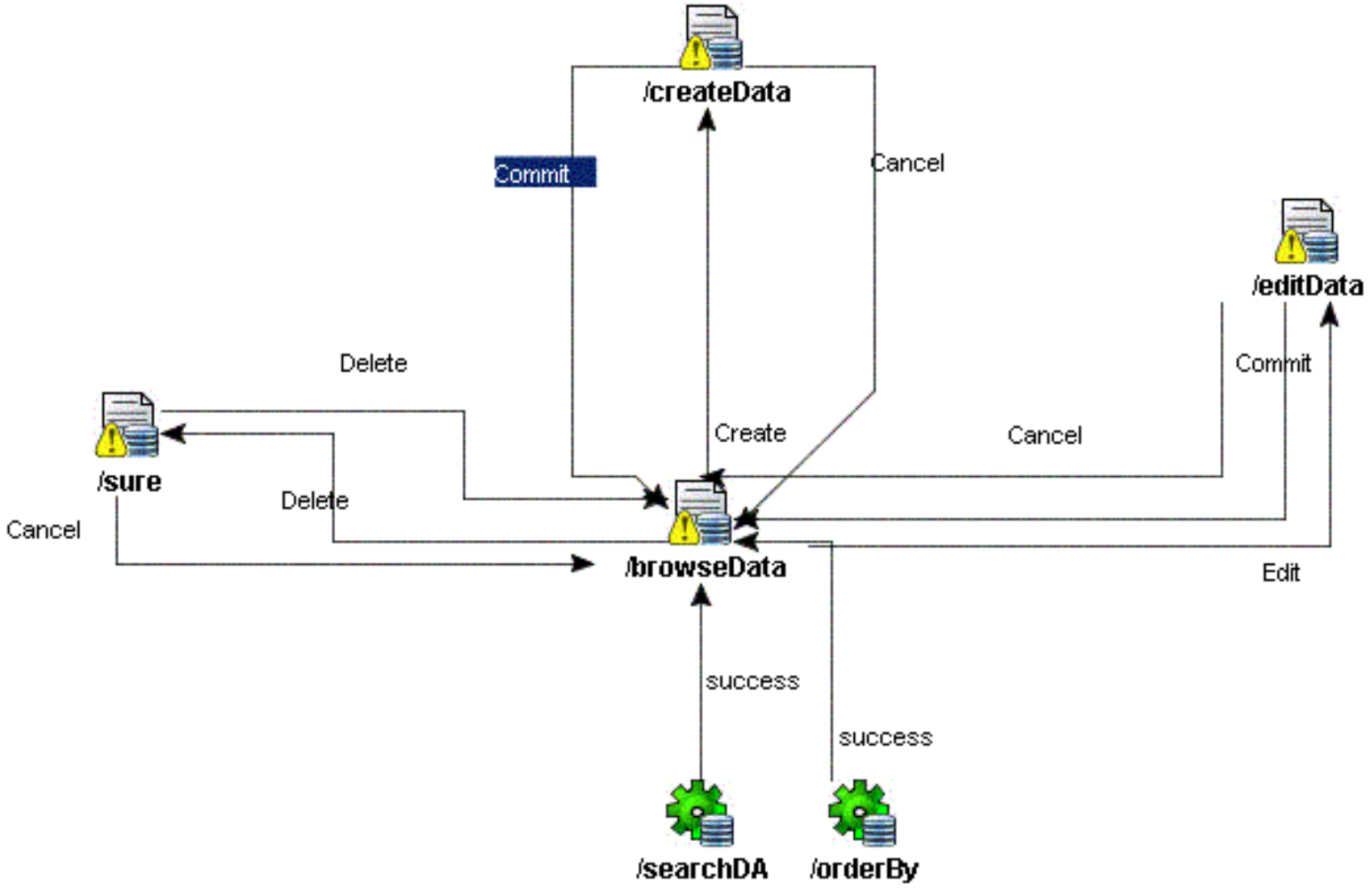
- Fields on the Edit screen appear or disappear depending upon the security settings of the user who is logged in.
- Fields can:
  - be editable
  - be display-only
  - not show at all.
- Size of the fields based on domain settings of associated column in the object model.



The Generator

- Once the logical application is specified, the user calls a database procedure that starts the application generation process.
- Generator is written in PL/SQL.
  - Consists of about 18,000 lines of code
  - Output is a JDeveloper workspace folder in the operating system.
  - Once generation is complete, workspace is zipped and the user can download the zip file onto his/her local machine.
  - Generated workspace uses a small custom tag library (paging functionality used in the browse page) and a code library (for security).

Struts Page Flow Diagram

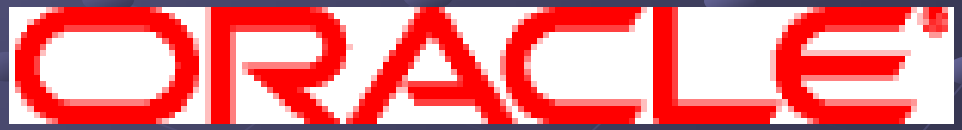


Conclusions

- It is possible to create a complete architecture to describe and generate a full J2EE application.
- Challenges:
  - Creating a repository/grammar to describe the system
  - Deciding on the UI and architecture that you want to generate.
- Writing the repository managers and generators is a relatively simple task.
- Surprise is really how well it all works.
- Using this approach, applications are quickly specified, effortlessly generated, and easily maintained.
- It can be a long road to get it all working, but there is a great pay-off at the end of the process.

The J2EE SIG

Co-Sponsored by:



Chairperson – Dr. Paul Dorsey

About the J2EE SIG

- Mission: To identify and promote best practices in J2EE systems design, development and deployment.
- Look for J2EE SIG presentations and events at national and regional conferences
- Website: www.odtug.com/2005_J2EE.htm
- Join by signing up for the Java-L mailing list:
  - http://www.odtug.com/subscrib.htm

J2EE SIG Member Benefits

- Learn about latest Java technology and hot topics via SIG whitepapers and conference sessions.
- Take advantage of opportunities to co-author Java papers and be published.
- Network with other Java developers.
- Get help with specific technical problems from other SIG members and from Oracle.
- Provide feedback to Oracle on current product enhancements and future product strategies.

Share your Knowledge: Call for Articles/Presentations

- Submit articles, questions, … to

IOUG – The SELECT Journal

select@ioug.org

ODTUG – Technical Journal

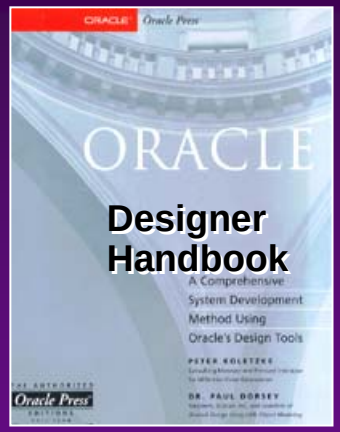
pubs@odtug.com

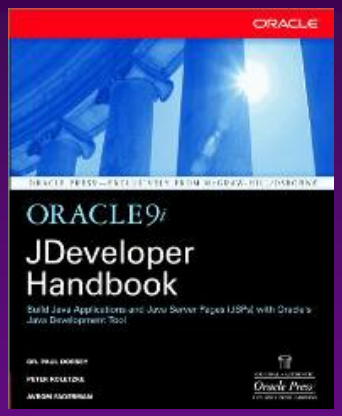
Dulcian's BRIM® Environment

- Full business rules-based development environment
- For Demo
  - Write "BRIM" on business card
- Includes:
  - Working Use Case system
  - "Application" and "Validation Rules" Engines

Contact Information

- Dr. Paul Dorsey – paul_dorsey@dulcian.com
- Dulcian website - www.dulcian.com





Coming in 2006:
Oracle PL/SQL for Dummies