Proper Plugin Protocols

Ciera Jaspan
Doctoral Candidate in Software Engineering
Job Talk @ University of San Francisco

Carnegie Mellon
isr institute for SOFTWARE RESEARCH

Have you used a software framework lately?

- Swing/AWT
- Java Servlets
- ASP.NET
- EJB
- Spring
- Ruby on Rails
- iPhone app framework

...and did you have trouble?

What's the problem here?

```
public void duplicate(Collection<String> coll) {
    Iterator<String> itr = coll.iterator();
    String str;

    while (itr.hasNext()) {
        str = itr.next();
        coll.add(str);
    }
}
```


Compile-time checking of framework errors

- I won't make the problem go away

```
public void duplicate(Collection<String> coll) {
    Iterator<String> itr = coll.iterator();
    String str;

    while (itr.hasNext()) {
        str = itr.next();
        coll.add(str);
    }
}
```

Compile-time checking of framework errors

- I won't make the problem go away
- I will help you find your errors at compile time

```
public void duplicate(Collection<String> coll) {
    Iterator<String> itr = coll.iterator();
    String str;

    while (itr.hasNext()) {
        str = itr.next();
        coll.add(str);
    }
}
```


Today's talk

- A motivating example from ASP.NET
- Collaboration constraints
- Specifying with relationships
- Statically analyzing code to find defects
- Tradeoffs in cost-effectiveness
- Implemented as FUSION, an Eclipse plugin
- Teaching and research interests

Motivating example: DropDownList (ASP.NET)

- Can add drop down lists to a web page
- Can change the selection programmatically
- Only one item is selected at a time

Make: -None Selected- Model: -None Selected-

Motivating example: DropDownList (ASP.NET)

- Can add drop down lists to a web page
- Can change the selection programmatically
- Only one item is selected at a time

Make: -None Selected- Model: -None Selected-

-None Selected-
Prius
Corolla
Neon
Viper
Ram
Civic
Accord
Insight
Exploder
Crown Victoria
Taurus

Motivating example: DropDownList (ASP.NET)

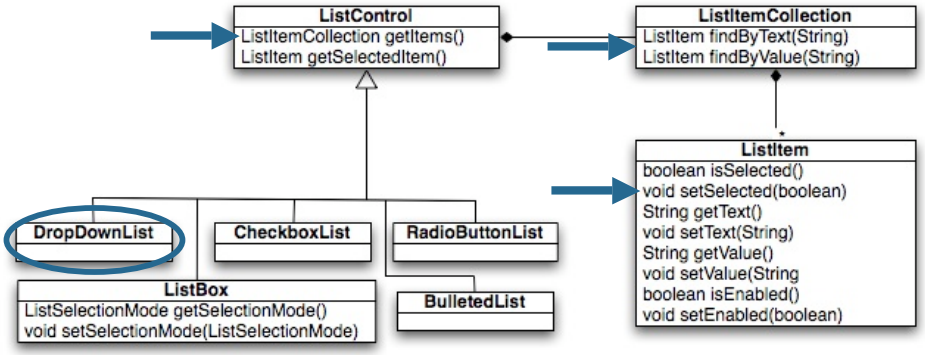
- Can add drop down lists to a web page
- Can change the selection programmatically
- Only one item is selected at a time

Let's change the selection

```
String searchTerm = …;

DropDownList ctrl = getControl("makeList");
```

Class diagram



That's a lot of calls just to change the selection…

Let's change the selection

```
String searchTerm = …;
ListItem newItem;
DropDownList ctrl = getControl("makeList");



newItem = ctrl.getItems().findByValue(searchTerm);
newItem.setSelected(true);
```

No Plugin Code!

Cannot have multiple items selected in a DropDo

Stack Trace:

```
[HttpException (0x80004005): Cannot have multiple items selected in a DropDownList.]
   System.Web.UI.WebControls.DropDownList.VerifyMultiSelect() +133
   System.Web.UI.WebControls.ListControl.RenderContents(HtmlTextWriter writer) +206
   System.Web.UI.WebControls.WebControl.Render(HtmlTextWriter writer) +43
   System.Web.UI.Control.RenderControlInternal(HtmlTextWriter writer, ControlAdapter adapter) +74
   System.Web.UI.Control.RenderControl(HtmlTextWriter writer, ControlAdapter adapter) +291
```

Correct code

```
String searchTerm = …;
ListItem newItem, oldItem;
DropDownList ctrl = getControl("makeList");

oldItem = ctrl.getSelectedItem();
oldItem.setSelected(false);

newItem = ctrl.getItems().findByValue(searchTerm);
newItem.setSelected(true);
```


Can we switch these?

```
String searchTerm = …;
ListItem newItem, oldItem;
DropDownList ctrl = getControl("makeList");

newItem = ctrl.getItems().findByValue(searchTerm);
newItem.setSelected(true);

oldItem = ctrl.getSelectedItem();
oldItem.setSelected(false);
```

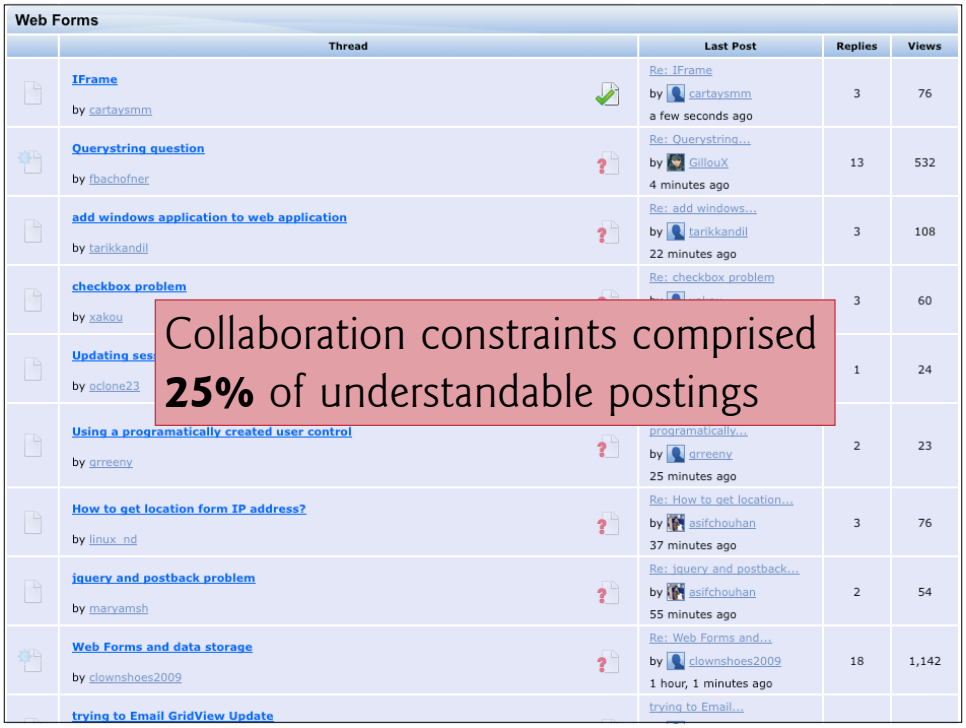
Correct code

```
String searchTerm = …;
ListItem newItem, oldItem;
DropDownList ctrl = getControl("makeList");

oldItem = ctrl.getSelectedItem();
oldItem.setSelected(false);

newItem = ctrl.getItems().findByValue(searchTerm);
newItem.setSelected(true);
```

Collaboration Constraint
A constraint on how several objects may interact



Common properties of these constraints

- **Multiple objects**
  - Frequently 2-5 objects, and identities matter
- **Temporal**
  - Must de-select currently selected `ListItem` before selecting a new one
  - Ordering of operations and code paths matter
- **Extrinsic nature**
  - `DropDownList` constrains methods of `ListItem` (without `ListItem` knowing it!)
  - Very different from a class invariant
- **Non-code artifacts**
  - Objects defined in ASPX, XML, JSP…
  - Constraints span code and non-code files

Today's talk

- A motivating example from ASP.NET
- Collaboration constraints
- **Specifying with relationships**
- **Statically analyzing code to find defects**
- Tradeoffs in cost-effectiveness
- Implemented as FUSION, an Eclipse plugin
- Teaching and research interests

A very simple collaboration

```
public class ListItemCollection {

   public void add(ListItem item);

   public void remove(ListItem item);
}
```

Removes item from this

Adds item to this

*Define a **relationship** type:*
`Item(ListItem, ListItemCollection)`

Specify with a relationship

```
public class ListItemCollection {

   public void add(ListItem item);

   public void remove(ListItem item);
}
```

Removes `Item(item, target)`

Adds `Item(item, target)`

Relationships

- Collaboration constraint
  - A constraint on how multiple objects may interact
- Relationship
  - A named, typed tuple on multiple objects
  - Describes how they are associated

```
Item(ListItem, ListItemCollection)
```


More relationships

```
public class ListControl {
   public ListItem getSelectedItem();
}
```

Adds `Child(result, target)` *and* `Selected(result)`

```
public class ListItemCollection {
   public boolean contains(ListItem item);
}
```

Adds or removes `Item(item, target)`

Relationship Effects

- Describe **what information is learned** as an effect of the framework operation
- Write them in code as annotations
- Framework developer defines the relations
  - **No pre-defined semantics**
  - The analysis doesn't know what "Item" means
- Framework developer annotates the API
  - No specifications on plugin code

Annotating the API

```
public class ListItemCollection {
   @Item({"item", "target"})
   public void add(ListItem item);

   @Item(value = {"item", "target"},
         effect = Effect.REMOVE)
   public void remove(ListItem item);
}
```

```
public class ListControl {
   @Child({"result", "target"})
   @Selected({"result"})
   public ListItem getSelectedItem();
}
```

Wildcards and tests

```
public class ListItemCollection {
    @Item(value = {"*", "target"},
          effect = Effect.REMOVE)
    public void clear();

   @Item({"item", "target"},
          effect = Effect.TEST,
          test = "result")
    public boolean contains(ListItem item);
}
```


Relationships effects provide semantic context

Will show context between {}

```
String searchTerm = …;
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl = getControl("makeList");
{}
oldItem = ctrl.getSelectedItem();

oldItem.setSelected(false);

coll = ctrl.getItems();


newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```

Relationships effects provide semantic context

```
String searchTerm = …;
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl = getControl("makeList");
{}
oldItem = ctrl.getSelectedItem();
{Child(oldItem, ctrl), Selected(oldItem)}
oldItem.setSelected(false);

coll = ctrl.getItems();


newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```


Relationships effects provide semantic context

```
String searchTerm = …;
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl = getControl("makeList");
{}
oldItem = ctrl.getSelectedItem();
{Child(oldItem, ctrl), Selected(oldItem)}
oldItem.setSelected(false);
{Child(oldItem, ctrl), !Selected(oldItem)}
coll = ctrl.getItems();


newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```

Relationships effects provide semantic context

```
String searchTerm = …;
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl = getControl("makeList");
{}
oldItem = ctrl.getSelectedItem();
{Child(oldItem, ctrl), Selected(oldItem)}
oldItem.setSelected(false);
{Child(oldItem, ctrl), !Selected(oldItem)}
coll = ctrl.getItems();
{Child(oldItem, ctrl), !Selected(oldItem),
 ItemList(coll, ctrl)}
newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```


Relationships don't have to come from code

- Also exist in non-code artifacts
- ASPX, JSP, XML, Java Properties files
- Extract using a query language
- FUSION processes declarative files first
  - Uses these for the starting context
  - Currently supporting XML files with XQuery

Can write constraints on the context

A constraint has four parts:

Operation
The operation we are constraining

Trigger predicate
Predicate logic over relationships
When to constrain the operation

Requires predicate
Predicate logic over relationships
Must be true if the constraint is triggered

Effect list
List of relationship effects
Will be applied if the constraint is triggered

Constraints as annotations

```
@Constraint(name = "Deselect Old",




)
public class DropDownList {
}
```

Constraints as annotations

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",




)
public class DropDownList {
}
```


Constraints as annotations

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",
   trg = "sel == false and Child(target, ctrl) and
          ctrl instanceof DropDownList",


)
public class DropDownList {
}
```

Constraints as annotations

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",
   trg = "sel == false and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "Selected(target)",

)
public class DropDownList {
}
```


Constraints as annotations

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",
   trg = "sel == false and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "Selected(target)",
   eff = {"!CorrectlySelected(ctrl)"}
)
public class DropDownList {
}
```

Constraints as annotations

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",
   trg = "sel == false and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "Selected(target)",
   eff = {"!CorrectlySelected(ctrl)"}
)
@Constraint(name = "Select New",
   op = "ListItem.setSelected(sel)",
   trg = "sel == true and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "!CorrectlySelected(ctrl)",
   eff = {"CorrectlySelected(ctrl)"}
)
public class DropDownList {
}
```


First example

```
ListItem newItem;
ListItemCollection coll;
DropDownList ctrl;

coll = ctrl.getItems();
newItem = coll.findByValue(searchTerm
{Child(newItem, ctrl)}
newItem.setSelected(true);
```

Fails!
trg *matches, but don't know if* ***req*** *is true!*

```
@Constraint(name = "Select New",
   op = "ListItem.setSelected(sel)",
   trg = "sel == true and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "!CorrectlySelected(ctrl)",
   eff = {"CorrectlySelected(ctrl)"}
)
```

Second example

```
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl;

oldItem = ctrl.getSelectedItem();
{Selected(oldItem), Child(oldItem, ctrl)}
oldItem.setSelected(false);

coll = ctrl.getItems();

newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```

Doesn't apply ***trg*** *doesn't match*

```
@Constraint(name = "Select New",
   op = "ListItem.setSelected(sel)",
   trg = "sel == true and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "!CorrectlySelected(ctrl)",
   eff = {"CorrectlySelected(ctrl)"}
)
```

Second example

```
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl;

oldItem = ctrl.getSelectedItem();
{Selected(oldItem), Child(oldItem, ctrl)}
oldItem.setSelected(false);

coll = ctrl.getItems();

newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```

Passes ***trg*** *matches and* ***req*** *is true.*

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",
   trg = "sel == false and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "Selected(target)",
   eff = {"!CorrectlySelected(ctrl)"}
)
```

Second example

```
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl;

oldItem = ctrl.getSelectedItem();
{Selected(oldItem), Child(oldItem, ctrl)}
oldItem.setSelected(false);
{!Selected(oldItem), Child(oldItem, ctrl), !CorrectlySelected(ctrl)}
coll = ctrl.getItems();

newItem = coll.findByValue(searchTerm);

newItem.setSelected(true);
```

Passes
trg *matches so*
eff *applies*

```
@Constraint(name = "Deselect Old",
   op = "ListItem.setSelected(sel)",
   trg = "sel == false and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "Selected(target)",
   eff = {"!CorrectlySelected(ctrl)"}
)
```

Second example

```
ListItem newItem, oldItem;
ListItemCollection coll;
DropDownList ctrl;

oldItem = ctrl.getSelectedItem();
{Selected(oldItem), Child(oldItem, ctrl)}
oldItem.setSelected(false);
{!Selected(oldItem), Child(oldItem, ctrl), !CorrectlySelected(ctrl)}
coll = ctrl.getItems();
{…, !CorrectlySelected(ctrl)}
newItem = coll.findByValue(searchTerm);
{…, Child(newSel, ctrl), !CorrectlySelected(ctrl)}
newItem.setSelected(true);
```

Passes!
trg *matches and*
req *is true.*

```
@Constraint(name = "Select New",
   op = "ListItem.setSelected(sel)",
   trg = "sel == true and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "!CorrectlySelected(ctrl)",
   eff = {"CorrectlySelected(ctrl)"}
)
```

Today's talk

- A motivating example from ASP.NET
- Collaboration constraints
- Specifying with relationships
- Statically analyzing code to find defects
- **Tradeoffs in cost-effectiveness**
- Implemented as FUSION, an Eclipse plugin
- Teaching and research interests

An adoptable approach

- Specifications must be
  - Written by framework developers only
  - Incremental
- My tool should
  - Analyze plugin code only
  - Direct developers to root cause of the error
  - Provide cost-effective results

Cost-effectiveness and precision

Precision: how good are the results?

	Error exists	Error does not exist
Analysis reports error	**True Positive**	**False Positive**
Analysis doesn't report error	**False Negative**	True Negative

Cost-effectiveness and precision

Sound: Guarantees it finds **all** possible defects (of a class of defects)

	Error exists	Error does not exist
Analysis reports error	**True Positive**	**False Positive**
Analysis doesn't report error	**False Negative**	True Negative

Cost-effectiveness and precision

Complete: Guarantees it finds **only** real defects

	Error exists	Error does not exist
Analysis reports error	True Positive	False Positive
Analysis doesn't report error	False Negative	True Negative

How do we analyze this?

```
ListItem newItem;
ListItemCollection coll;
DropDownList ctrl;

coll = ctrl.getItems();
newItem = coll.findByValue(searchTerm);
{Child(newItem, ctrl)}
newItem.setSelected(tru
```

Don't know whether `CorrectlySelected(ctrl)` *is true or false! It is* ***unknown***.

```
@Constraint(name = "Select New",
   op = "ListItem.setSelected(sel)",
   trg = "sel == true and Child(target, ctrl) and
          ctrl instanceof DropDownList",
   req = "!CorrectlySelected(ctrl)",
   eff = {"CorrectlySelected(ctrl)"}
)
```

Soundness and completeness

- Sound: Finds **all** real defects

	Constraint **applies** when trigger is…	Constraint **passes** when requires is…
Sound		

Soundness and completeness

- Sound: Finds **all** real defects

	Constraint **applies** when trigger is…	Constraint **passes** when requires is…
Sound	True or Unknown	

Soundness and completeness

- Complete: Finds **only** real defects

	Constraint **applies** when trigger is…	Constraint **passes** when requires is…
Sound	True or Unknown	True
Complete		

Soundness and completeness

- Complete: Finds **only** real defects

	Constraint **applies** when trigger is…	Constraint **passes** when requires is…
Sound	True or Unknown	True
Complete	True	

Soundness and completeness

- Sound: Finds **all** real defects
- Complete: Finds **only** real defects

	Constraint **applies** when trigger is…	Constraint **passes** when requires is…
Sound	True or Unknown	True
Complete	True	True or Unknown

The pragmatic variant

- Doesn't apply constraint unless it's sure
- When it is, insists the requires predicate be true

	Constraint **applies** when trigger is…	Constraint **passes** when requires is…
Sound	True or Unknown	True
Complete	True	True or Unknown
Pragmatic	True	True

Cost-effectiveness

- **Three variants** of the analysis
  - Sound (no false negatives)
  - Complete (no false positives)
  - Pragmatic (balance of both)
- How do we know they are sound/complete?
  - Formal semantics of core language in [Jaspan09]
  - Extended semantics and proofs in [JaspanTR08]
- Which will perform better on real code?

Real Case Study

- Specifying the Spring Web Application framework
  - Large and popular industry framework
  - Java and XML
- Pulling examples from developer help forums
  - Classifying the type of example
  - Specifying the constraint
  - Demonstrating that FUSION finds the bug!
- Deep analysis of the three variants
  - Which is better on real code?
  - What causes the differences in the results?

Other challenges of the work

- A new, more general definition of "software framework"
- Analysis of industry frameworks to support the definition
- Tracking object identity across language boundaries
- Handling aliasing in the static analysis
- Handling broken behavioral subtyping

Can discuss these in depth at a later time

Today's talk

- A motivating example from ASP.NET
- Collaboration constraints
- Specifying with relationships
- Statically analyzing code to find defects
- Tradeoffs in cost-effectiveness
- **Implemented as FUSION, an Eclipse plugin**
- Teaching and research interests

Live Demo!

- The DropDownList example
- Iterators
  - Check for hasNext() before next()
  - Check for concurrent modification

Major contributions of this research

- An expanded definition of **software frameworks** (beyond OO)
- A deep understanding of **collaboration constraints**
- **Relationships** as an abstraction to specify collaboration constraints
- An investigation of **precision and cost-effectiveness** in static analyses

Today's talk

- A motivating example from ASP.NET
- Collaboration constraints
- Specifying with relationships
- Statically analyzing code to find defects
- Tradeoffs in cost-effectiveness
- Implemented as FUSION, an Eclipse plugin
- **Teaching and research interests**

Other interests: Adoptable tools

- Human-readable error messages [Jaspan08]
  - Error Reporting Logic (ERL)
  - Translates **first-order predicate logic** into **readable English**
  - Validated with user studies on complex nested predicates

```
TRUE IMPLIES HasNext(target) and
CollIterator(target, coll)
```

HasNext(itr) must be true.

Other interests: Adoptable tools

- Human-readable error messages [Jaspan08]
  - Error Reporting Logic (ERL)
  - Translates **first-order predicate logic** into **readable English**
  - Validated with user studies on complex nested predicates

```
TRUE IMPLIES HasNext(target) and
CollIterator(target, coll)
```

HasNext(itr) must be true.
Try checking itr.hasNext() first?

Other interests: Adoptable tools

- Human-readable error messages [Jaspan08]
  - Error Reporting Logic (ERL)
  - Translates **first-order predicate logic** into **readable English**
  - Validated with user studies on complex nested predicates
- Visualizations of relationships at each line
- Dynamically inferring collaboration constraints

Other interests: Deployment configurations

- Complex configurations to use frameworks (Rails, Eclipse, Spring)
- Filesystem and network configurations
- Differences between dev, test, and production environments
- Experience from eBay and LEVEL: this is a large source of errors!
- **Can we automatically detect these with relationships?**

Other interests: Software Design

- Framework Design
  - How can we design better frameworks?
  - How can we implement usable frameworks?
- Quality Attributes and Design
  - Can we statically determine the quality attributes of our design before implementation?
  - Can we statically change the quality attributes of the system?
- Teaching Design
  - How should we study good software designs?
  - What can we learn from failed system designs?

Courses I can teach

- **System design**
  - Can't just check off all the coding errors
  - Many failures are at the design level
  - Study engineering design and failures and practice it!
- **Large system development and maintenance**
  - How do you start in a 200KLOC-2MLOC project?
  - Work on a real large project (Red Hat and Mozilla want you!)
- **Quality assurance techniques**
  - Not enough trained QA engineers
  - Not just testing: static analysis, dynamic analysis, code reviews, statistical beta testing…

Student Take-aways

- Collaboration constraint
  - A extrinsic constraint across multiple objects
- Static analysis
  - Soundness: finds all defects
  - Completeness: finds only defects
- Precision and cost-effectiveness
  - False positives cost developer time
  - False negatives cost loss of confidence