Early stage work

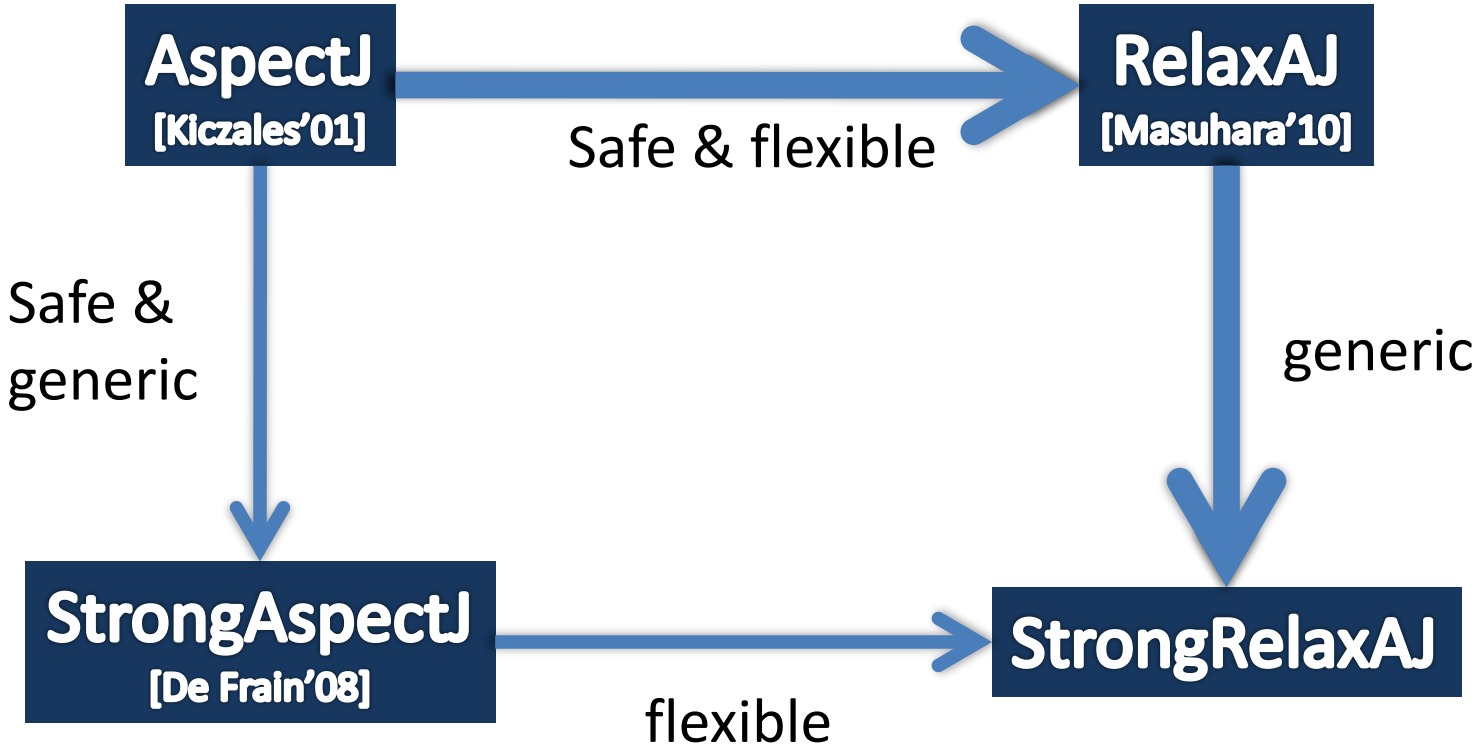
StrongRelaxAJ: integrating adaptability of RelaxAJ and expressiveness of StrongAspectJ

Tomoyuki Aotani

Manabu Touyama and Hidehiko Masuhara

University of Tokyo, Japan

Background: improving type-safety and expressiveness of around advice

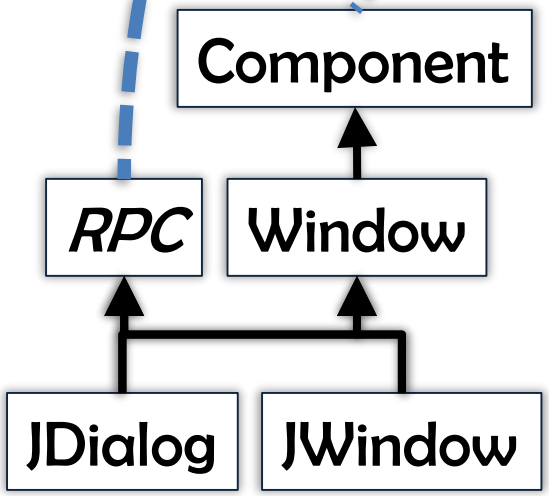


Background – AspectJ is less flexible: changing a JDialog to a JWindow

```
void showPreview(Frame mainWin){
  JDialog popup=
                  new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(close);
  popup.setVisible(true);
}
```

Replace with JWindow:

```
JWindow popup=
    new JWindow(mainWin);
```




```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  new JWindow(f);
}
```

Wrong

Background – AspectJ is less flexible: **JDialog** cannot be replaced due to types

- *forall Rs. Ra <= Rs* must be hold
  - *Rs*: return type of a join point shadow
  - *Ra*: return type of around advice

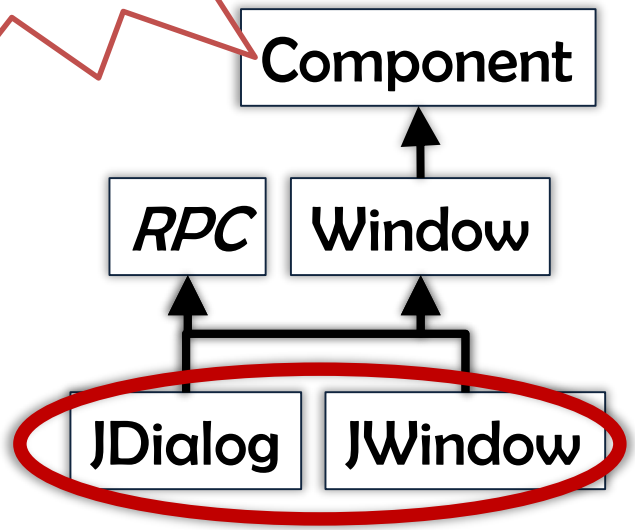
Must be a subtype of **JDialog** in AspectJ

Not satisfied!

```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  new JWindow(f);
}
```

apply

```
new JDialog(mainWin);
```



Background – RelaxAJ[Masuhara'10]: accepting any type if safely used

- *forall Rs. Ra <= Rs* is NOT required
  - *Rs*: return type of a join point shadow
  - *Ra*: return type of around advice
- Instead, *Ra* must be a subtype of all the *usage types* of the returned value
  - receiver's type of a method call
  - field's type of a field set

Background – RelaxAJ[Masuhara'10]: accepting any type if safely used

```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  return new JWindow(mainWin);
}
```

collect usage types of returned value

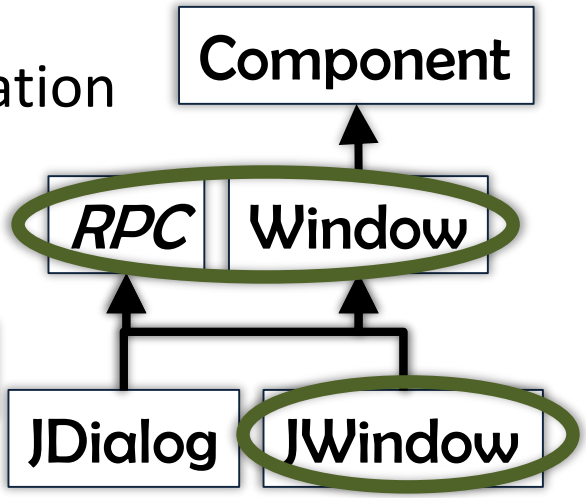
RelaxAJ compiler → ***OK!***

```
void showPreview(Frame mainWin){
  JDialog popup=
        new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane()
                    .add(close);
  popup.setVisible(true);
}
```

used as RPC

used as Component

check relation



Problems of RelaxAJ

- *Lack of expressiveness*:
  return type of around advice must be a single class
- *Strange typing*:
  signature of proceed is the same to the one of around advice
  - Same to AspectJ

Problems of RelaxAJ

- *Lack of expressiveness*:
  return type of around advice must be a single class
- *Strange typing*:
  signature of proceed is the same to the one of around advice
  - Same to AspectJ

Extended example: replacing a **JDialog** with a **JWindow** conditionally

```
void showPreview(Frame mainWin){
  JDialog popup=
                  new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(close);
  popup.setVisible(true);
}
```

Replace with JWindow:

```
JWindow popup=
    new JWindow(mainWin);
```

if DECORATE *is false*

Wrong

```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  if(DECORATE) return proceed(f);
  else return new JWindow(f);
}
```

Extended example: replacing a **JDialog** with a **JWindow** conditionally

```
void showPreview(Frame mainWin){
  JDialog popup=
                new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(close);
  popup.setVisible(true);
}
```

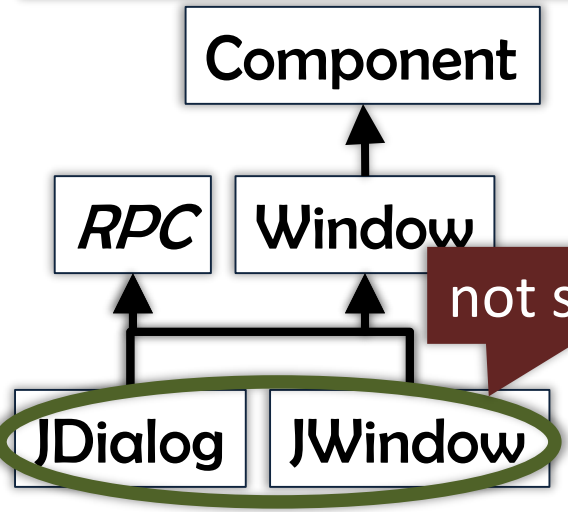
Replace with JWindow:

```
JWindow popup=
    new JWindow(mainWin);
```

if DECORATE *is false*

returns JDialog
➔ not JWindow

Wrong




```
JWindow around(Frame f):
  call(JDialog.new(Frame)) &&args(f){
  if(DECORATE) return proceed(f);
  else return new JWindow(f);
}
```

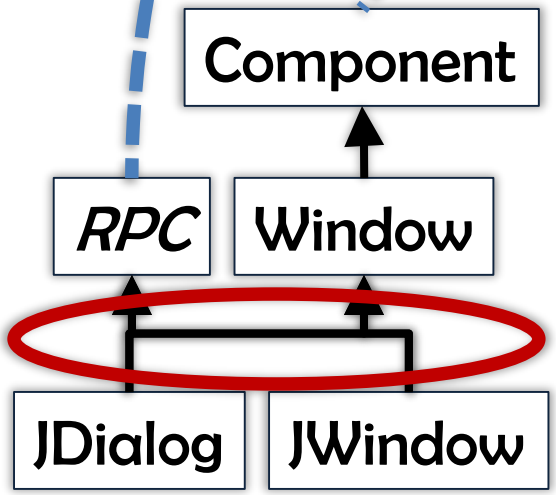
What can be the return type?

Problem 1: no suitable return type for the advice

```
void showPreview(Frame mainWin){
  JDialog popup=
                new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(close);
  popup.setVisible(true);
}
```

Return type T should be:

- T <: **RPC**
- T <: **Window**
- **JWindow** <: T
- **JDialog** <: T




```
T around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  if(DECORATE) return proceed(f);
  ...ow(f);
}
```

No such type!

Problems of RelaxAJ

- *Lack of expressiveness*:
  return type of around advice must be a single class
- *Strange typing*:
  signature of proceed is the same to the one of around advice
  - Same to AspectJ

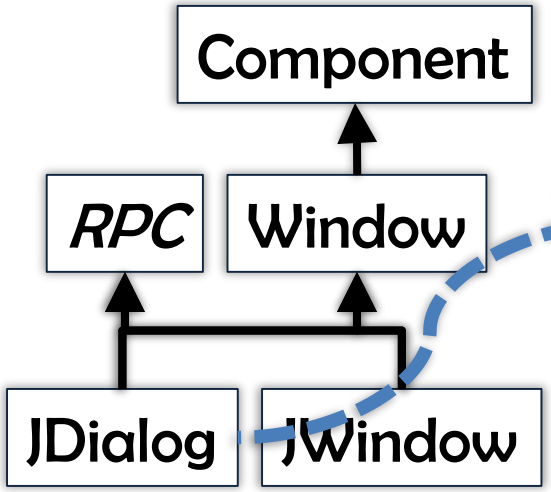
Extended example: making a JDialog modal

```
void showPreview(Frame mainWin){
  JDialog popup=
                  new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(clase);
  popup.setVisible(true);
}
```

Make the JDialog modal and use JWindow as a popup

```
JDialog d=
  new JDialog(mainWin);
d.setModal(true);
JWindow popup=
    new JWin                    Win);
```

Wrong




```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  proceed(f).setModal(true);
  return new JWindow(f);
}
```

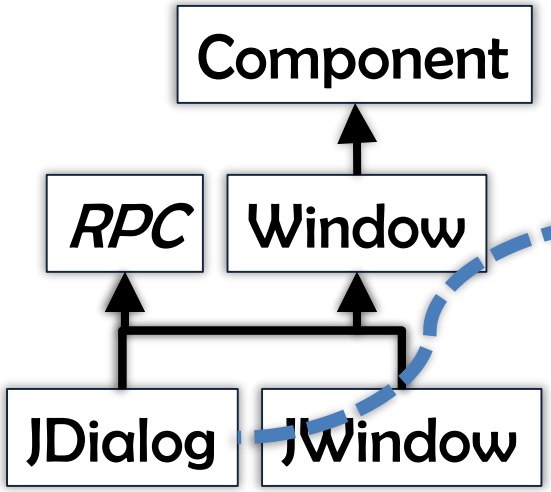
Problem 2: return types of around advice and its proceed are the same

```
void showPreview(Frame mainWin){
  JDialog popup=
                new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(clase);
  popup.setVisible(true);
}
```

Make the JDialog modal and use JWindow as a popup

```
JDialog d=
  new JDialog(mainWin);
d.setModal(true);
JWindow popup=
    new JW                    Win);
```

Wrong




```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f){
  proceed(f).setModal(true);
  return new JWindow(f);
}
```

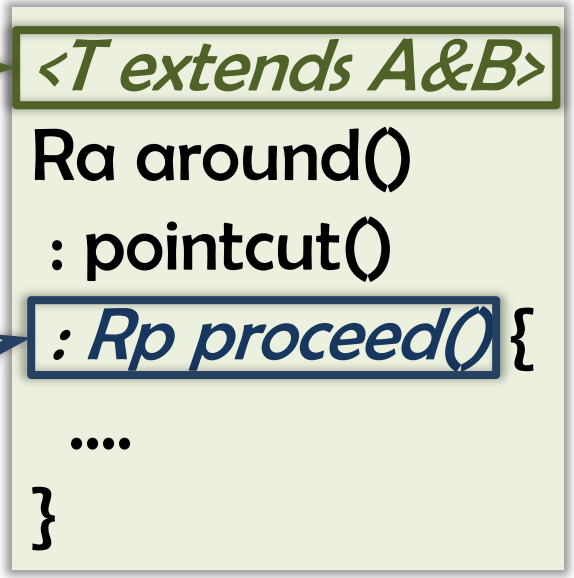
Sig. of proceed:
Frame -> JWindow

But it never returns JWindow!

Our solution: StrongRelaxAJ

- Extending RelaxAJ with
  - *Bounded type variables*: representing "some type that can be used as type A and B"
  - *Explicit signature of proceed*: specifying the return type of proceed

Advice in StrongRelaxAJ:



- These features are found in StrongAspectJ
  - StrongRelaxAJ may= StrongAspectJ + RelaxAJ

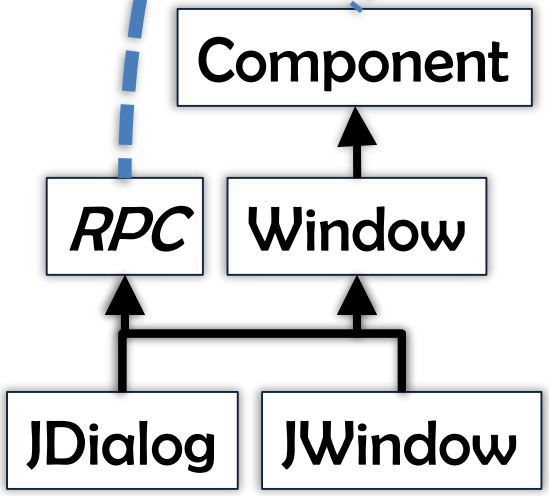
Type variables: representing “some type that can be used as type A and B”

```
void showPreview(Frame mainWin){
  JDialog popup=
                 new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(close);
  popup.setVisible(true);
}
```

Replace with JWindow:

```
JWindow popup=
    new JWindow(mainWin);
```

if DECORATE *is false*



Some type that be used as Component and RPC

```
<T extends Component & RPC>
T around(Frame f):
 call(JWindow.new(Frame))&&args(f){
  if(!DECORATE) return proceed(f);
  else return new JWindow(f);
}
```

JDialog as T

JWindow as T

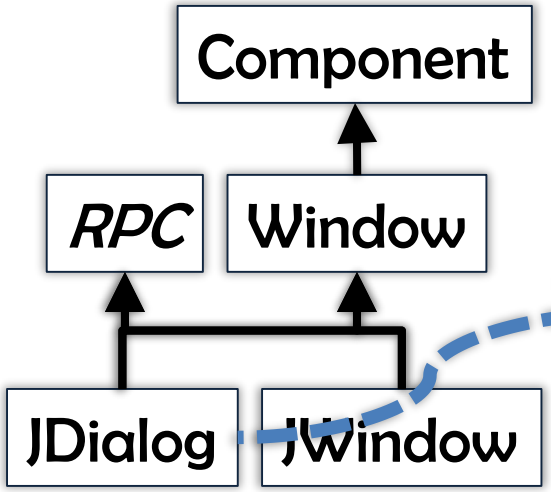
Explicit signature of proceed: specifying the return type of proceed

```
void showPreview(Frame mainWin){
  JDialog popup=
              new JDialog(mainWin);
  JButton close=new JButton("close");
  popup.getContentPane().add(clase);
  popup.setVisible(true);
}
```

OK!

Make the JDialog modal and use JWindow as a popup

```
JDialog d=
  new JDialog(mainWin);
d.setModal(true);
JWindow popup=
    new JWindow(mainWin);
```




```
JWindow around(Frame f):
  call(JDialog.new(Frame))&&args(f)
 : JDialog proceed(Frame){
  proceed(f).setModal(true);
  return new JWindow(f);
}
```

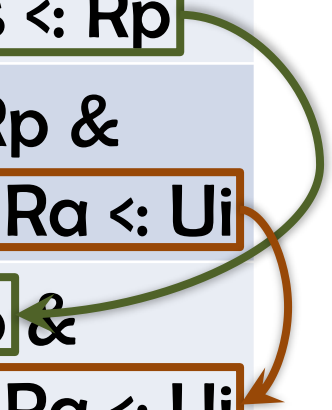
Sig. of proceed: Frame -> JDialog

Relationship between the return types of advice and proceed

```
o=(Rs)e;
((U1)o).m1();
((U2)o).m2();
```

```
Ra around():
   match(Rs e)
   : Rp proceed(){
  ...}
```

language	relationship
AspectJ	Ra == Rp <: Rs
StrongAspectJ	Ra <: Rs <: Rp
RelaxAJ	Ra == Rp & forall i. Ra <: Ui
StrongRelaxAJ	Rs <: Rp & forall i. Ra <: Ui



- Rs: ret. type of shadow
- Ra: ret. type of advice
- Ui: usage type of advice return

Experiments: # of application chances in applications

- Counted the number of variables that is used as more than 2 types in Shimple (SSA) by using Soot[Vallée-Rai'99]

	total	usage type > 1	%
jEdit	42450	72	0.17
JHotDraw (DrawApp)	4668	2	0.04
jython	41980	60	0.14
antlr	8807	4	0.05
freemind	27436	17	0.06

A few, but not none!

Conclusions and future work

- StrongRelaxAJ:
  an extension to RelaxAJ with
  - Bounded type variables
  - Explicit signature of proceed
- A few chances to apply StrongRelaxAJ aspects according to the result of preliminaly experiments
- Future work includes
  - Completing type-checking rule
  - Discussing type-safety formally
  - Mining useful examples from real applications