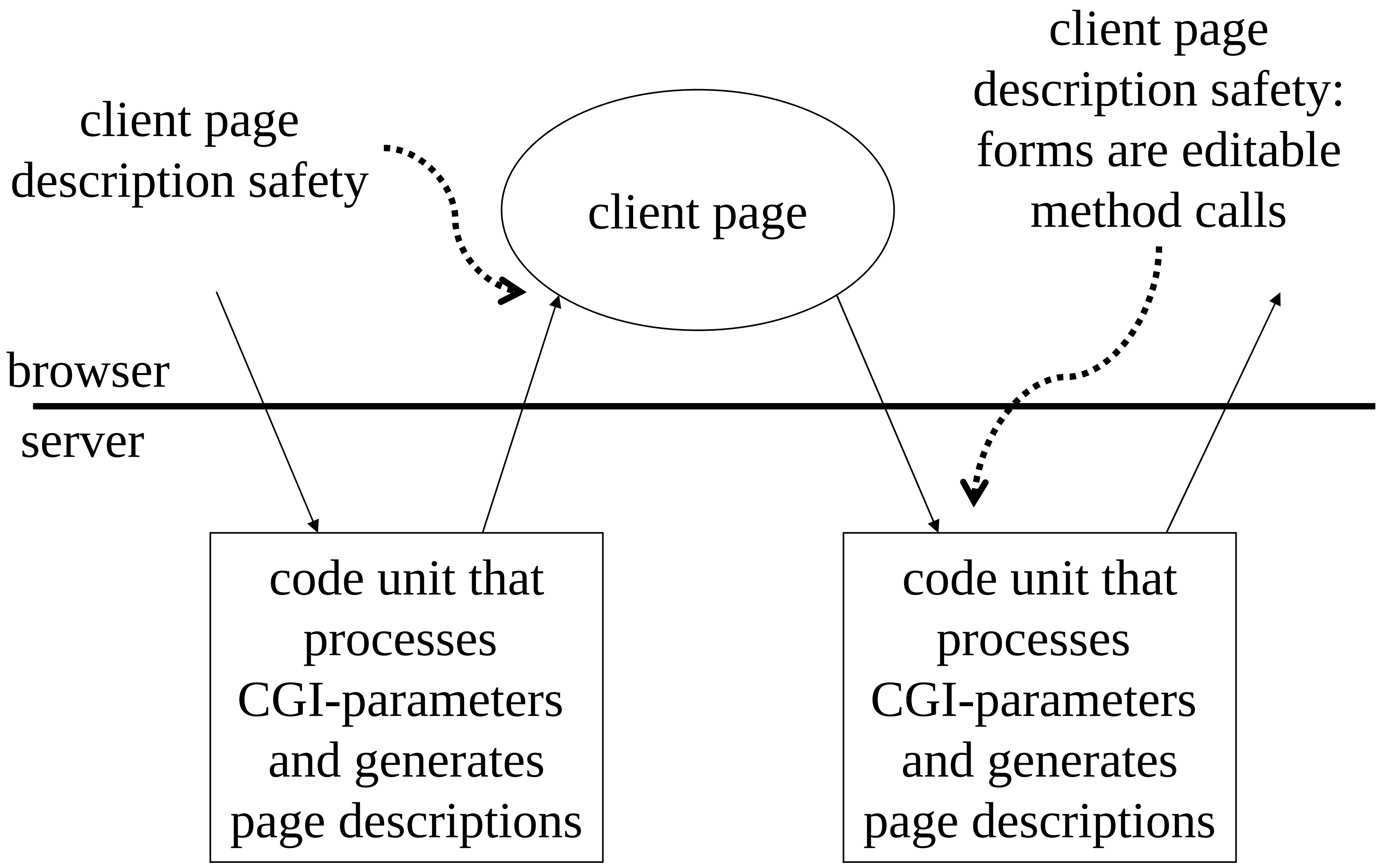
The Core NSP Type System

Dirk Draheim
Freie Universität Berlin

Gerald Weber
The University of Auckland

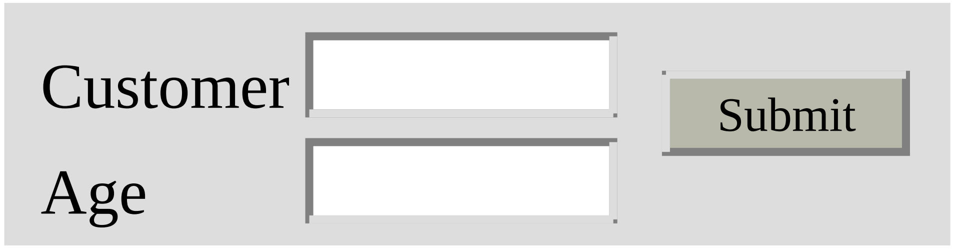
WMR 2006
Bari

Server Pages Safety Problems


client page description safety
client page
client page description safety: forms are editable method calls
browser
server
code unit that processes CGI-parameters and generates page descriptions
code unit that processes CGI-parameters and generates page descriptions

JSP Counter Example

```
<FORM ACTION="http://www.x.net/NewCustomer.jsp" method="GET">
  <%
  for (int i=0; i<j; i++) {%>
  <INPUT TYPE="TEXT" NAME="customer" SIZE="20"> <%}%>
  <INPUT TYPE="TEXT" NAME="age" SIZE="20">
  <%
  if (d==0) {%>
</FORM> <%}%>
```



NewCustomer.jsp

```
String name;
int age;
name = request.getParameter("foobar");
name = request.getParameter("customer");
try {
age = new Integer(request.getParameter("age")).intValue();
} catch (IllegalArgumentException _e){}
```

NSP – Parameterised Server Pages

```
<nsp name="Registration"><head>...</head><body>
  <form callee="NewCustomer">
    <input widget="String" param="customer"></input>
    <input widget="int" param="age"></input>
    <submit></submit>
  </form>
</body></nsp>

<nsp name="NewCustomer"><head><title>...</title></head>
  <param name="customer" type="String"/>
  <param name="age" type="int"/>
  <java>import myBusinessModel.CustomerBase;</java>
  <body>
    <java>
      CustomerBase.createCustomer(customer,age);
    </java>
    <redirect callee="Somewhere"></redirect>
</body></nsp>
```

NSP Features

- Parameterised server pages
- Support for complex types in forms
- Exchanging objects across web interaction
- Higher-order server pages
- Statically ensured client page description safety
- Statically ensured client page type safety
- No unresolved links
- Active controls
- Unifying client-side and server-side calls

NSP Type System

- Core NSP
- Core NSP Grammar
- Core NSP Types
- Core NSP Subtyping
- Type Operator: signature connection
- Core NSP Typing
- Theorem: Core NSP type checking is decidable
  - Core NSP is explicitly typed
  - Recursive subtyping is decidable

Core NSP Grammar

```
system ::= page | system system
page ::= <nsp name="id"> websig-core </nsp>
websig-core ::= param websig-core | webcall | include
param ::= <param name="id" type="parameter-type"/>
webcall ::= <html> head body </html>
head ::= <head><title> strings </title></head>
strings ::= ε |  string strings
body ::= <body> dynamic </body>
include ::= <include> dynamic </include>

string ::= s ∈ String      id ::= l ∈ Label
parameter-type ::=   t ∈ T ∪ P
supported-type ::=   t ∈ B_supported

dynamic  ::=   dynamic dynamic | ε | string
  | ul | li |  table | tr | td
  | call |  form | object | hidden |  submit
  | input | checkbox | select | option
  | expression | code
```

Core NSP Types

- Programming language types **T**
  - basic types **B** (primitive and supported)
  - type variables **V** (including type constants)
  - array types **A**, record types **R** = **Label** $\rightarrow_{\mathbf{part}}$**T**
  - recursive types **Y** = $\{\mu\, X\,.\, R \mid X \in \mathbf{V}, R \in \mathbf{R}\}$
- Server page types
  - page types **P** = $\{w \rightarrow r \mid w \in \mathbf{W}, r \in \mathbf{C} \cup \mathbf{D}\}$
  - web signatures **W** = **Label** $\rightarrow_{\mathbf{part}}(\mathbf{T} \cup \mathbf{P})$
  - complete web page **C** = { } *complete web page type*
  - document fragment types **D** = $\mathbf{L} \times \mathbf{W}$
  - layout types **L** = $\mathbf{E} \times \mathbf{F}$
  - element types **E**=$\{\circ$ **,•,TR,TD,LI,OP**$\}$*neutral doc.t., output t.,etc.*
  - form occurrences **F** = $\{\Downarrow,\Uparrow,\Updownarrow\}$ *inside f.t., outside f.t., neutral f.t.*
  - system types **S** = $\{\Diamond\}$ *well type*

Core NSP Typing – Selected Rules I

- $d \in \textbf{string} \Rightarrow d:((\bullet, \Updownarrow),\emptyset)$
- $e:T \Rightarrow$ <hidden param=“l“>e</hidden>$:((\circ, \Downarrow),\{l \mapsto T\})$
- $T \in \mathbf{B}_{\text{supported}} \Rightarrow$ <input type=“T“ param=“l“/>$:((\bullet, \Downarrow),\{l \mapsto T\})$
- </submit>$:((\bullet, \Downarrow), \emptyset)$
- $l:w\rightarrow$, $d:((e, \Downarrow), v)$, $v<w \Rightarrow$ <form callee=“l“>d</form>$:((e, \Uparrow), \emptyset)$
- $l:w\rightarrow D$, $as:v$, $v<w \Rightarrow$ <call callee=“l“>as</form>$:D$
- $d:((\circ$ *or* $\bullet,F),w) \Rightarrow$ <li>d</li>: $((\mathbf{LI},F),w)$
- $d:((\mathbf{LI}$ *or* $\circ,F),w) \Rightarrow$ <ul>d</ul>: $((\bullet,F),w)$
- $d_1:(L_1,w_1)$, $d_2:(L_2,w_2)$, $\mathit{def}(\mathit{lub}(L_1,L_2))$, $\mathit{def}\,(w_1 \otimes w_2) \Rightarrow$
  $d_1\, d_2: (\mathit{lub}(L_1,L_2), w_1 \otimes w_2)$

Core NSP Typing – Selected Rules II

- d:D, d∈**dynamic** ⇒ <include>d</include>:∅→D
- d:((• *or* ∘**,** ⇑ or ⇕), ∅), t∈**string,** d∈**dynamic** ⇒ <html><head><title>t</title></head><body>d</body></html>: ∅→
- l:T, c:w→D**,** l ∉ *dom*(w) ⇒ <param name="l" type="T" >c: (w ∪ {l ↦ T}) → D
- l:P, c:P, c ∈**websig-core** ⇒ <nsp name="l"> c </nsp>: ◇

Core NSP Subtyping – Establishing Rules –

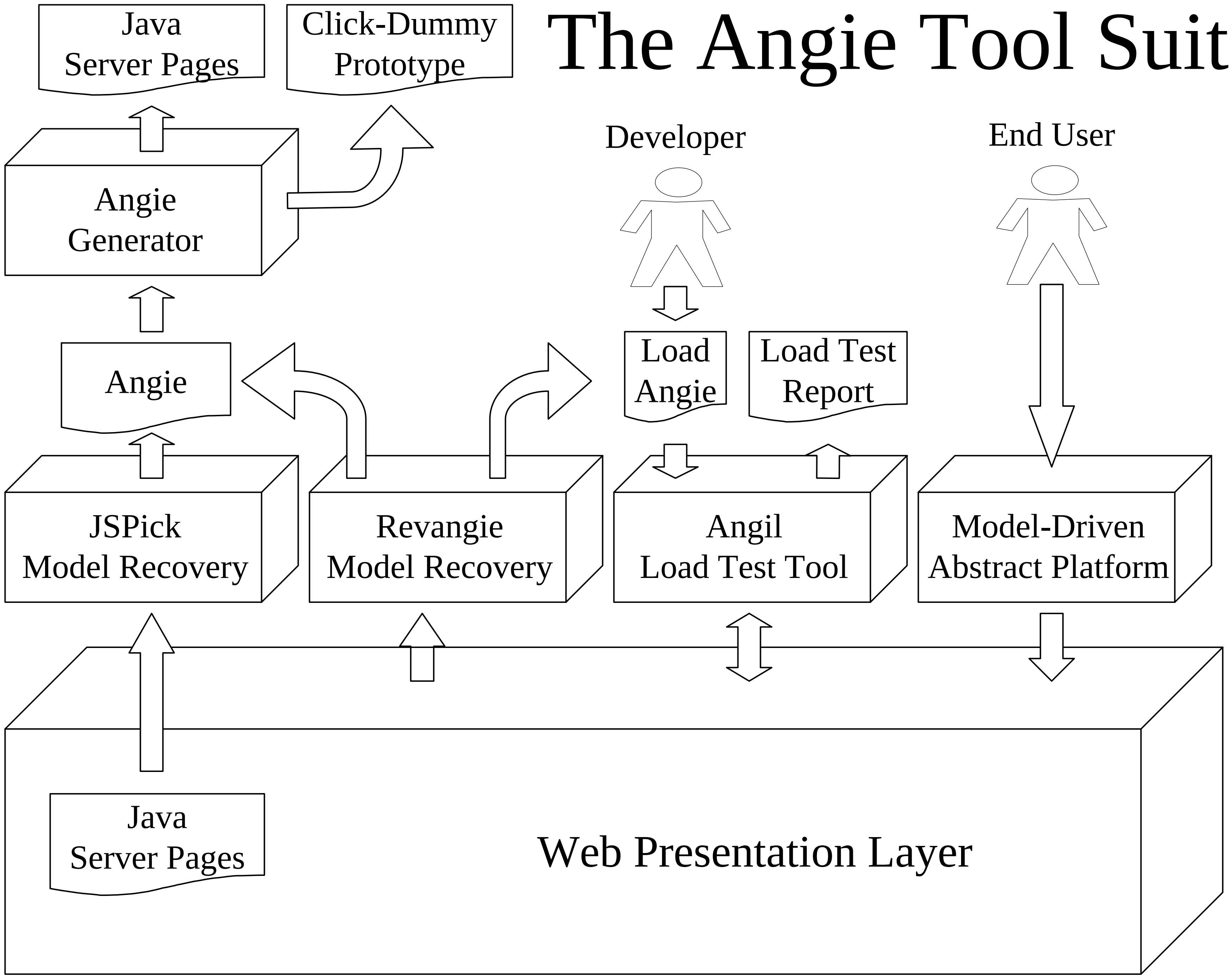
- $T < \text{array of } T$

- $T_j \notin (\mathbf{B}_{\text{primitive}} \cup \mathbf{P}),\ j \in 1..n \Rightarrow$
  $\{l_i \mapsto T_i\}^{i \in 1..j-1,j+1,..n} < \{l_i \mapsto T_i\}^{\mathbf{i \in 1..n}}$

- $\circ$ **< •,** $\circ$ **< TR,** $\circ$ **< TD,** $\circ$ **< LI,** $\circ$ **< OP**

- $\Updownarrow < \Downarrow,\ \Updownarrow < \Uparrow$

The Angie Tool Suite



Conclusion

- NSP is based on a well-understood system metaphor
- NSP ensures CPDS and CPTS at compile time
- NSP supports complex types in forms
- NSP improves web-based application architecture
- NSP seamlessly integrates with form-oriented analysis
- NSP has a convenient formal type system