When Frameworks Let You Down

Platform-Imposed Constraints on the Design and Evolution of Domain-Specific Languages

Danny M. Groenewegen, Zef Hemel, Lennart C.L. Kats, Eelco Visser

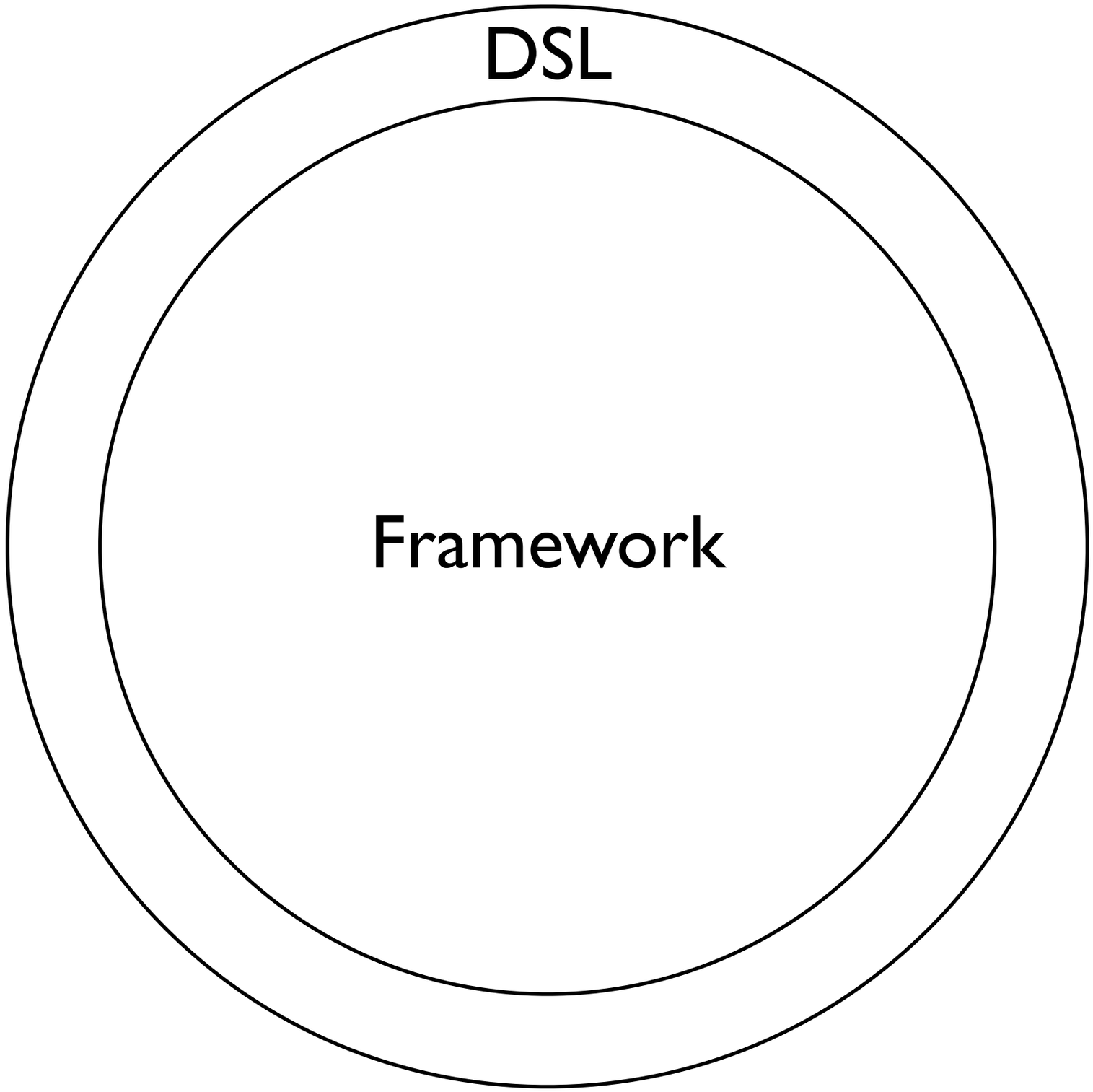
TU Delft

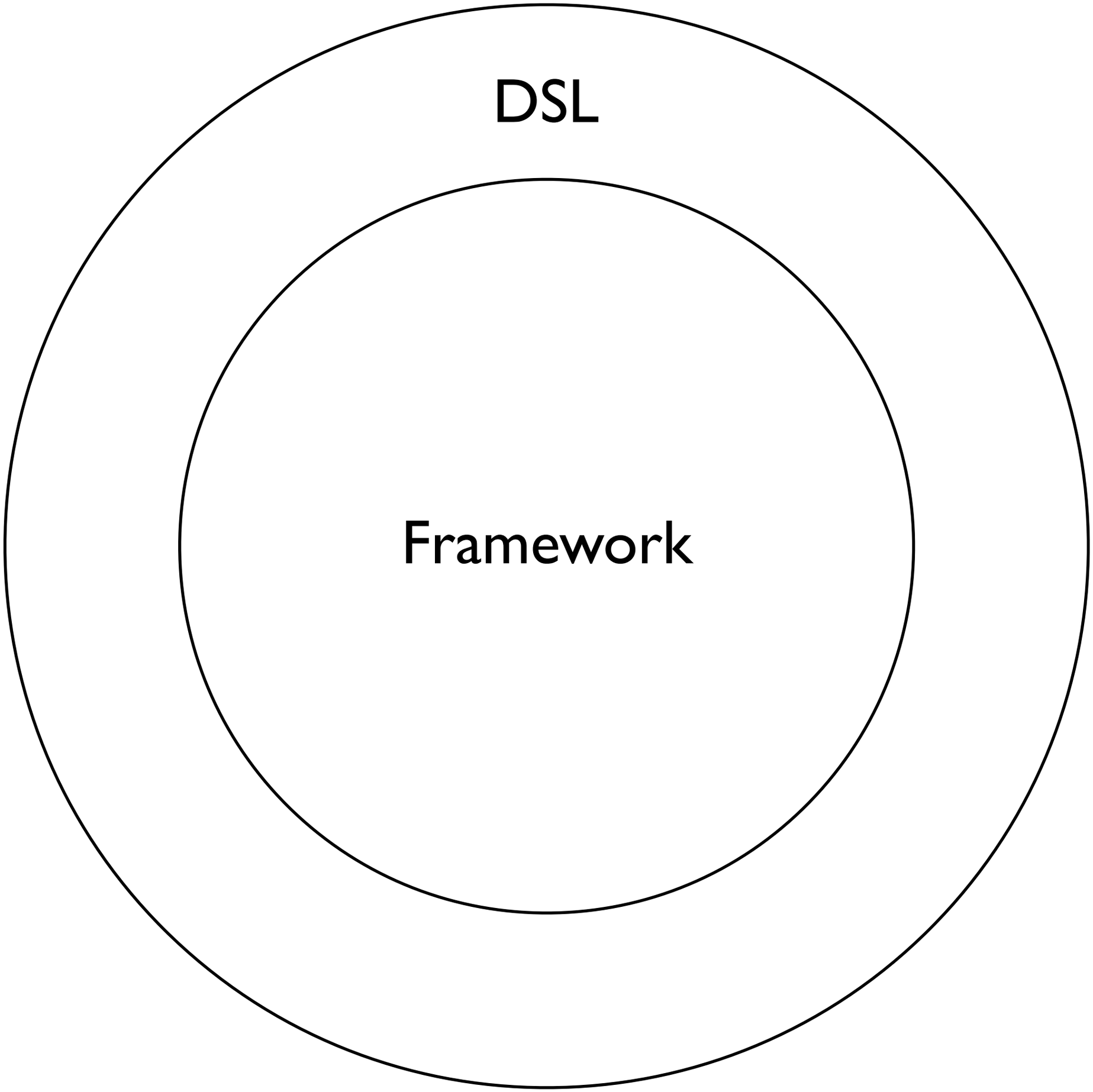
When Frameworks Let You Down

Platform-Imposed Constraints on the Design and Evolution of Domain-Specific Languages

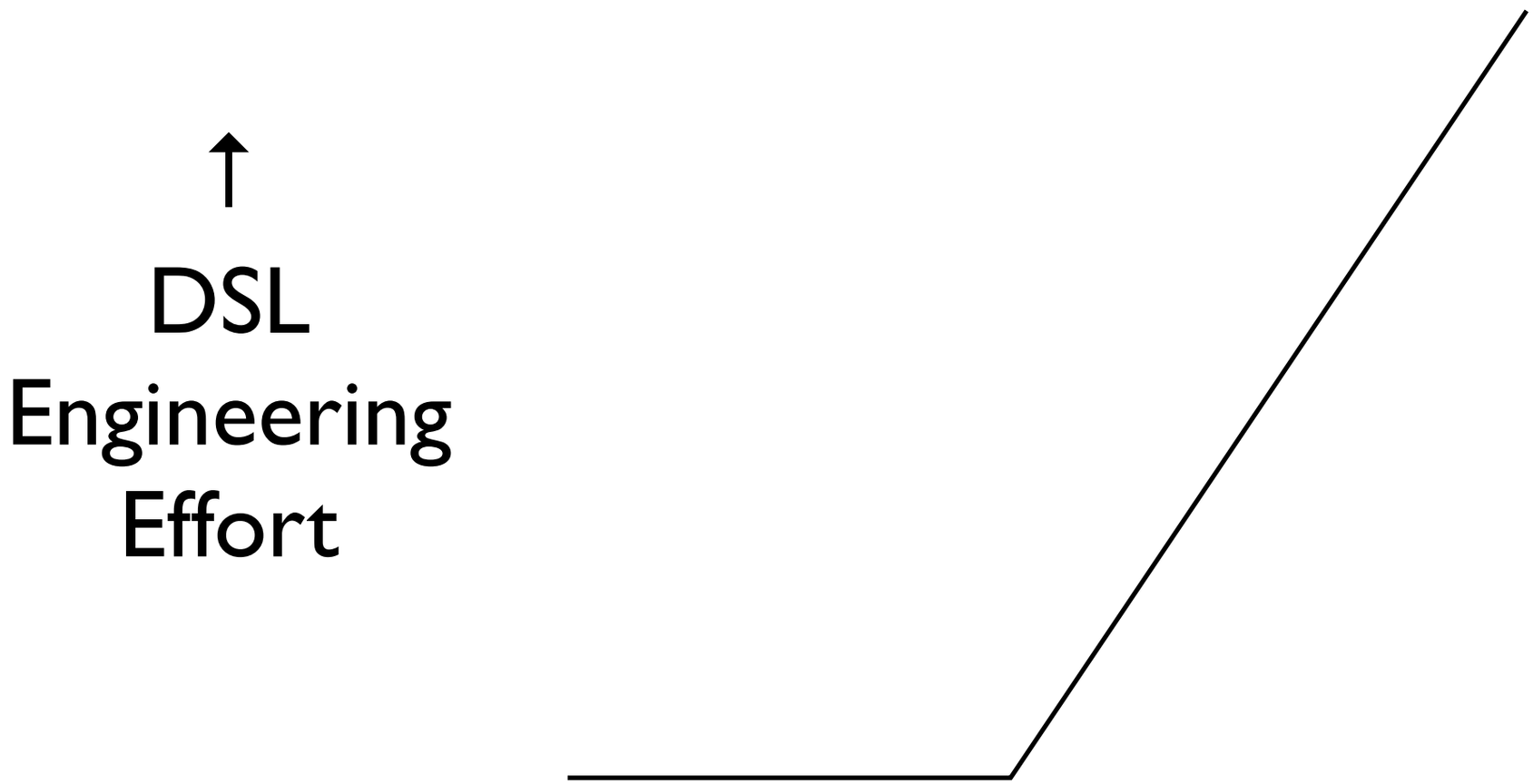
Danny M. Groenewegen, Zef Hemel, Lennart C.L. Kats, Eelco Visser

TU Delft


DSL
Framework


DSL
Framework

DSL based on Framework

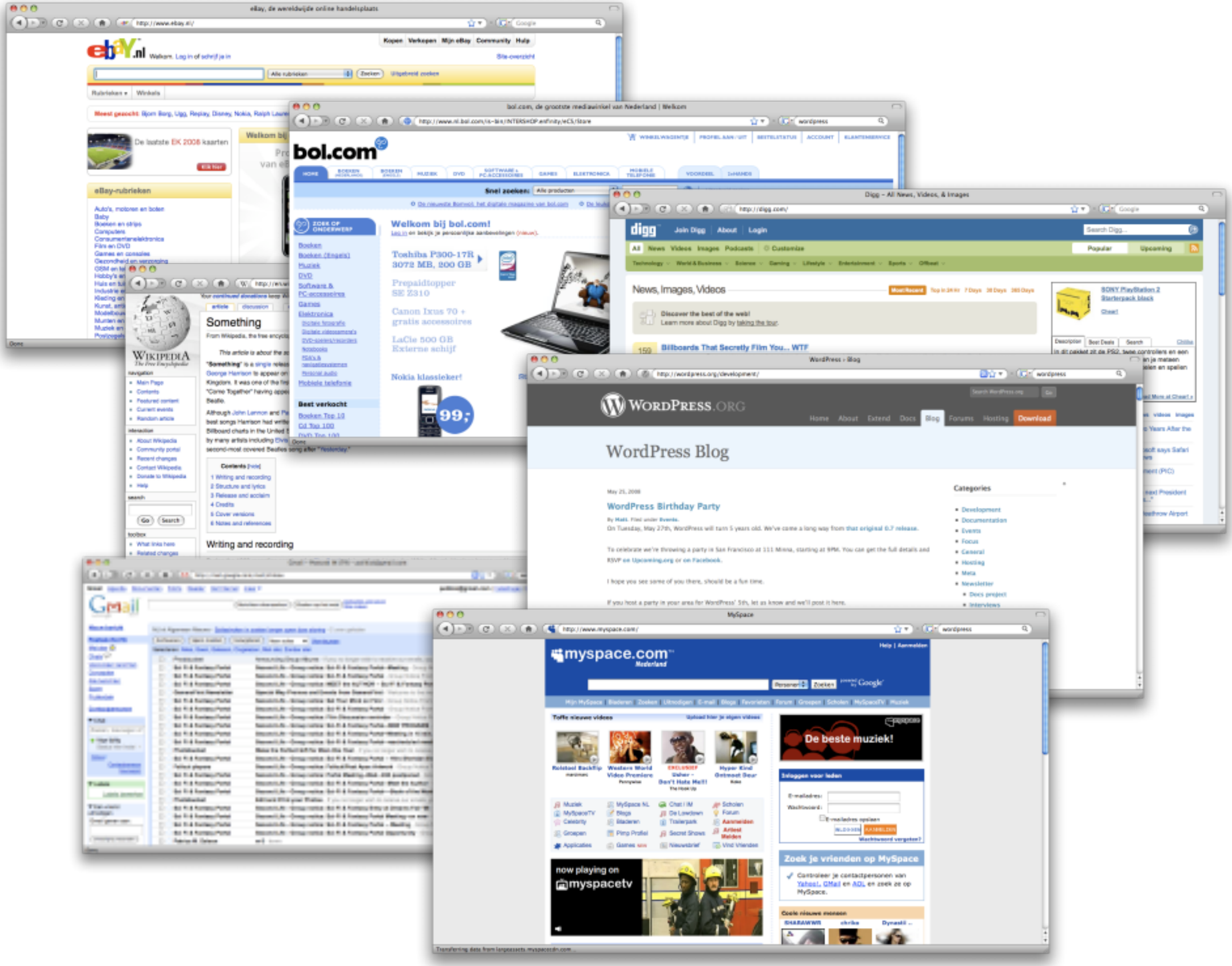


When Frameworks Let You Down

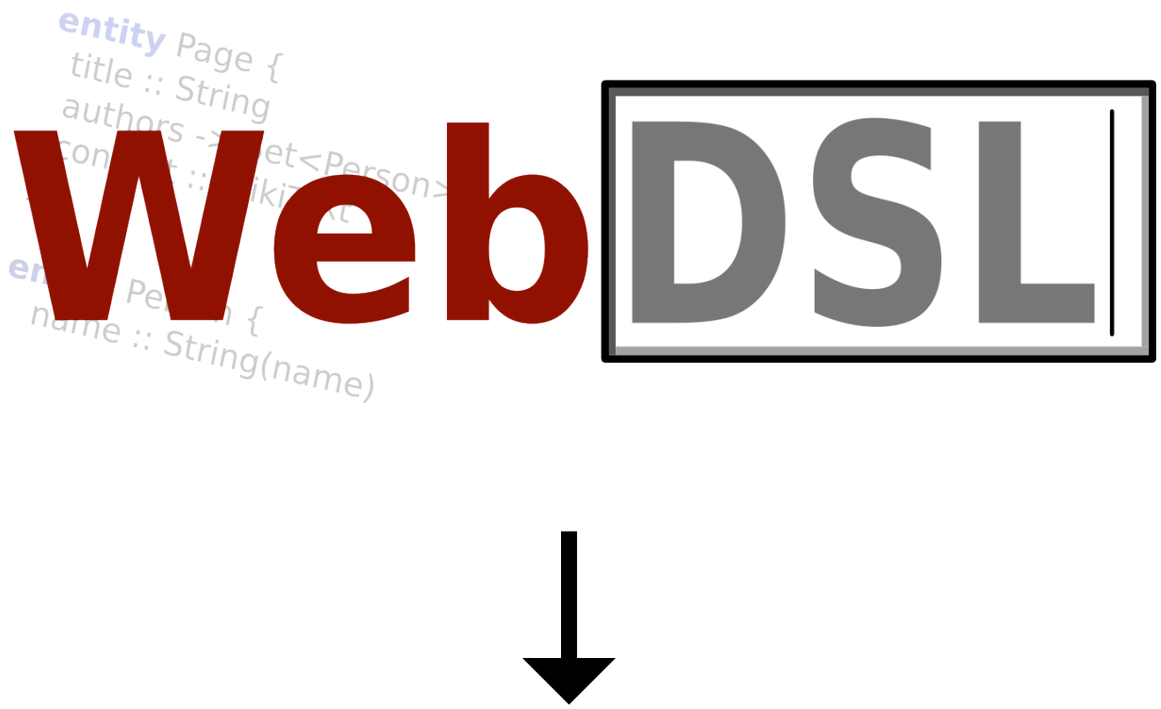
Platform-Imposed Constraints on the Design and Evolution of Domain-Specific Languages

Danny M. Groenewegen, Zef Hemel, Lennart C.L. Kats, Eelco Visser

TU Delft

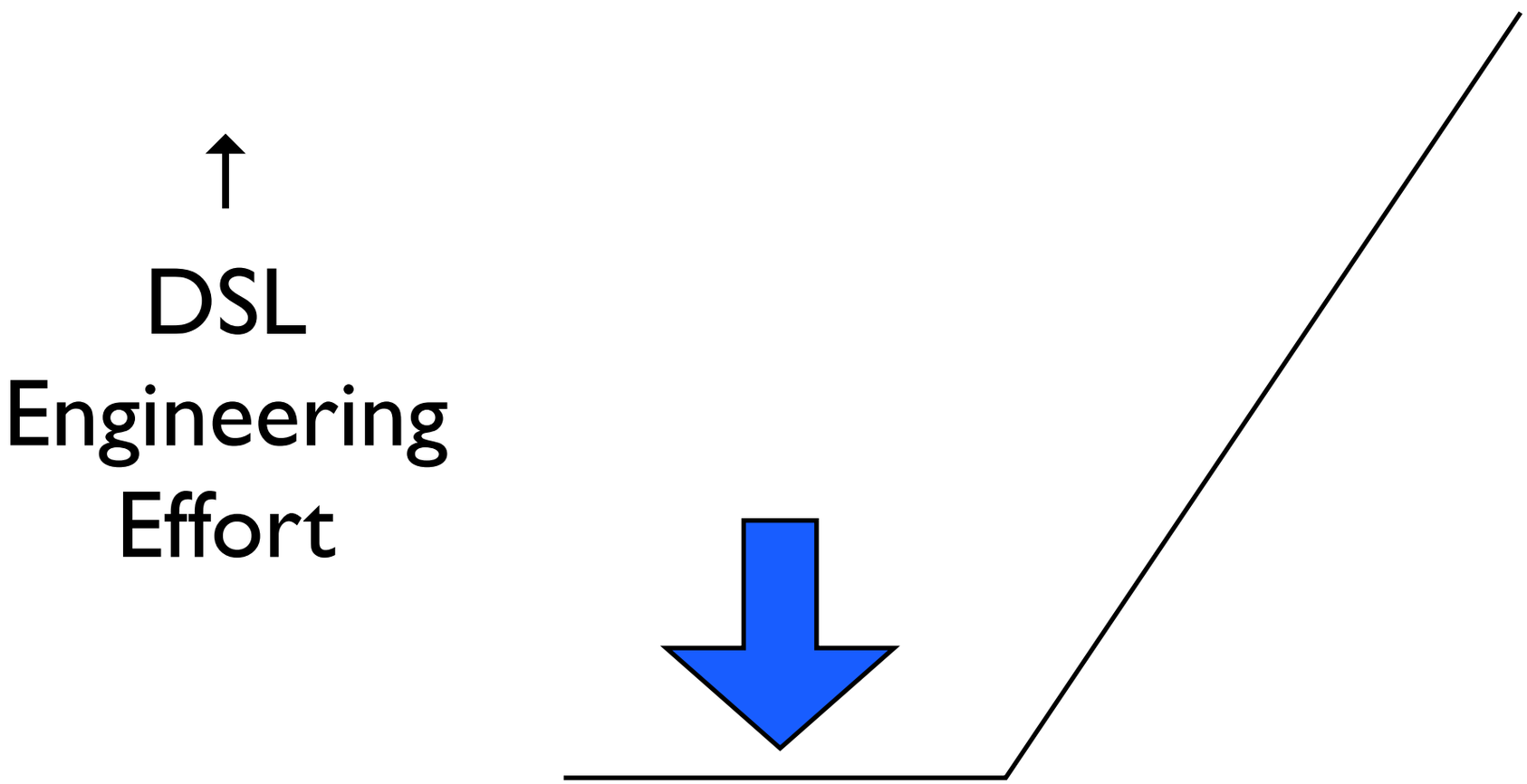

eBay, de wereldwijde online handelsplaats
ebay.nl
bol.com, de grootste mediawinkel van Nederland | Welkom
bol.com
Welkom bij bol.com!
Digg – All News, Videos, & Images
digg
News, Images, Videos
WordPress › Blog
WordPress.ORG
WordPress Blog
WordPress Birthday Party
Something
Writing and recording
Gmail
MySpace
myspace.com
Nederland
De beste muziek!
Zoek je vrienden op MySpace
now playing on myspacetv


entity Page {
title :: String
name :: String(name)
WebDSL


entity Page {
title :: String
authors ->
Person
name :: String(name)
WebDSL


SeamFramework.org

DSL based on Framework



Data Models

```
entity Blog {
  author   -> User (inverse=User.blogs)
  title    :: String
  entries  <> Set<BlogEntry>
}
```

Data Models

```
entity Blog {
  author   -> User (inverse=User.blogs)
  title    :: String
  entries  <> Set<BlogEntry>
}
```

```
@Entity
public class Blog {
   protected String _title = "";
   public String getTitle() {
     return _title;
   }
   public void setTitle(String value) {
     _title = value;
   }
   @ManyToOne
   protected User _author;
   public User getAuthor() {
     return _author;
   }
   public void setAuthor(User author) {
     _title = value;
   }
   @OneToMany(mappedBy="_blog",
              targetEntity=BlogEntry.class)
   @Cascade(...)
   protected List<BlogEntry> _entries;
   public List<BlogEntry> getEntries() {
     return _entries;
   }
   public void
   setEntries(List<BlogEntry> entries) {
     _entries = entries;
   }
}
```

User Interface

```
define page editBlogEntry(e : BlogEntry) {
  form{
    table{
      row{ "Title: " inputString(e.title) }
      row{ "Content: " inputWikiText(e.content) }
      action("Save", save())
    }
  }
  action save() {
    e.save();
    return blogEntry(e);
  }
}
```

User Interface

```
define page editBlogEntry(e : BlogEntry) {
  form{
    table{
      row{ "Title: " inputString(e.title) }
      row{ "Content: " inputWikiText(e.content) }
      action("Save", save())
    }
  }
  action save() {
    e.save();
    return blogEntry(e);
  }
}
```

```
<html> ... <body> ...
<h:form>
<table><tr>
<td>Title:</td><td>
<h:inputText value="#{editBlogEntry.e.title}"/>
</td></tr><tr>
<td>Content:</td><td>
<h:inputTextarea value="#{editBlogEntry.e.content}"/>
</td></tr></table>
<h:actionLink action="#{editBlogEntry.save()}"/>
</h:form>
...
</body> </html>
```

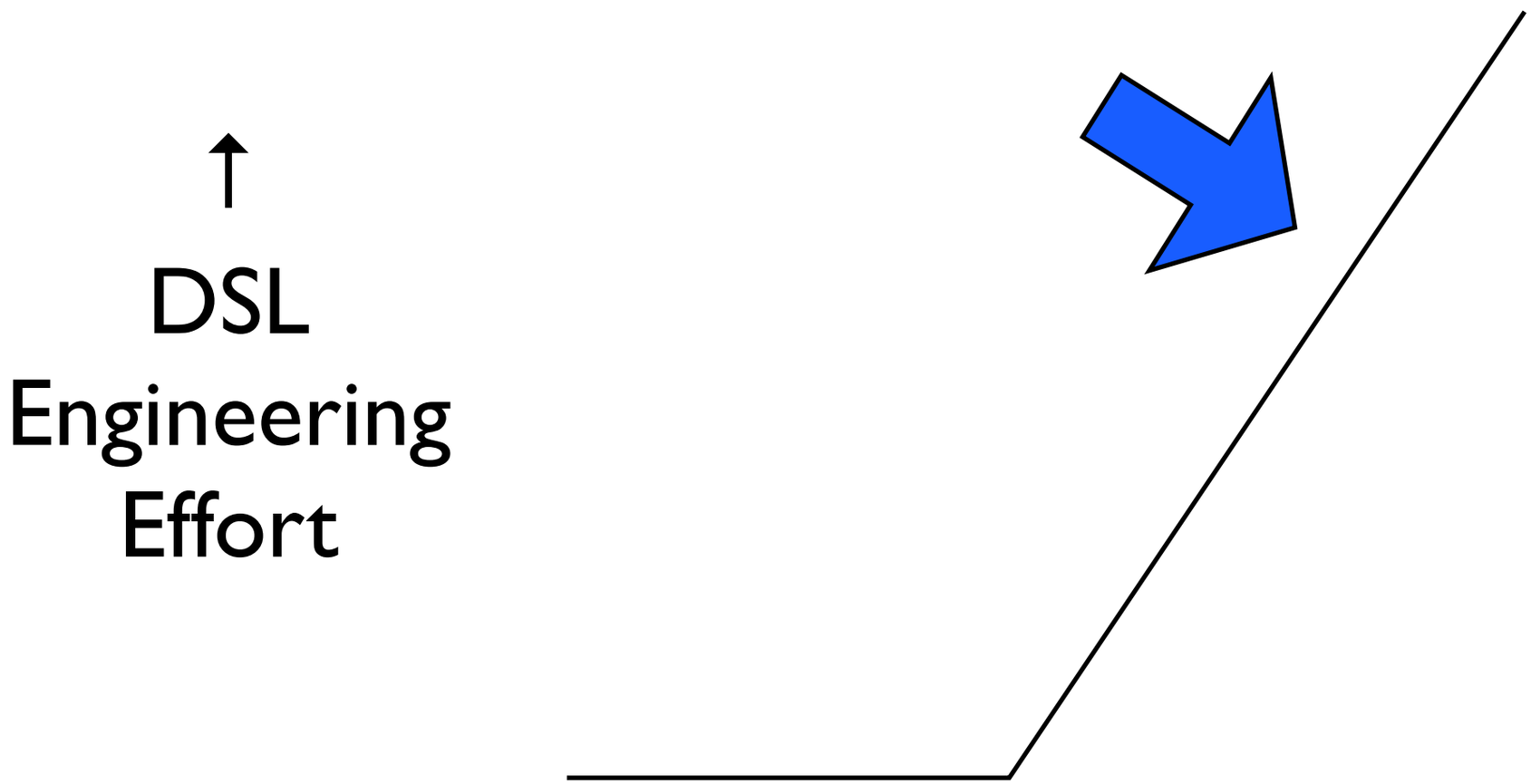
```
@Stateful @Name("editBlogEntry")
public class EditBlogEntry {
    ...
    @In @Out
    private BlogEntry e;

    public void setE(e) {
      this.e = e;
    }

    public BlogEntry getE() {
      return this.e;
    }

    public void save() {
      em.persist(e);
    }
    ...
}
```

Evolution



Examples

Template Mechanism

Access Control

Data Model Modularity

Template Mechanism

```
define page home() {
   header{“My Blog”}
   list {
      listitem { navigate(home()) { “Home” } }
      listitem { navigate(about()) { “About” } }
   }
   spacer
   for(p : Post) {
     output(p.title)
   }
   spacer
   “(C) WebDSL, 2008”
}
```

```
define page about() {
   header{“My Blog”}
   list {
      listitem { navigate(home()) { “Home” } }
      listitem { navigate(about()) { “About” } }
   }
   spacer
   par {
     “This is about a blog”
   }
   spacer
   “(C) WebDSL, 2008”
}
```

Template Mechanism

```
define main() {
   header{“My Blog”}
   list {
      listitem { navigate(home()) { “Home” } }
      listitem { navigate(about()) { “About” } }
   }
   spacer
   body()
   spacer
   “(C) WebDSL, 2008”
}

define page home() {
  main()
  define body() {
    for(p : Post) {
      output(p.title)
    }
  }
}
define page about() {
  main()
  define body() {
    par {
      “This is about a blog”
    }
  }
}
```

Template Mechanism

```
define main() {
   header{“My Blog”}
   list {
      listitem { navigate(home()) { “Home” } }
      listitem { navigate(about()) { “About” } }
   }
   spacer
   body()
   spacer
   “(C) WebDSL, 2008”
}

define page home() {
  main()
  define body() {
    for(p : Post) {
      output(p.title)
    }
  }
}
define page about() {
  main()
  define body() {
    par {
      “This is about a blog”
    }
  }
}
```

Template Mechanism

```
define main() {
   header{“My Blog”}
   list {
      listitem { navigate(home()) { “Home” } }
      listitem { navigate(about()) { “About” } }
   }
   spacer
   body()
   spacer
   “(C) WebDSL, 2008”
}

define page home() {
  main()
   define body() {
     for(p : Post) {
       output(p.title)
     }
   }
}
define page about() {
  main()
   define body() {
     par {
       “This is about a blog”
     }
   }
}
```

Template Mechanism

Mismatch with JSF Facelets

Template Inheritance

Static scope

Access Control

```
module userpages

  define page viewUser(u : User) {
    output(u.name)
    output(u.authored)
  }

  define page userList() {
    for (u : User) {
      navigate(viewUser(u)) { output(u.name) }
    }
  }
```

Access Control

```
module userpages

  define page viewUser(u : User) {
    init {
      if(securityContext.principal != u) {
        goto accessDenied();
      }
    }
    output(u.name)
    output(u.authored)
  }

  define page userList() {
    for (u : User) {
      navigate(viewUser(u)) { output(u.name) }
    }
  }
```

Access Control

```
module userpages

  define page viewUser(u : User) {
    init {
      if(securityContext.principal != u) {
        goto accessDenied();
      }
    }
    output(u.name)
    output(u.authored)
  }

  define page userList() {
    for (u : User) {
      if(securityContext.principal == u) {
        navigate(viewUser(u)) { output(u.name) }
      }
    }
  }
```

Access Control

```
module ac

  rule page viewUser(u:User) {
    principal == u
  }

module userpages

  define page viewUser(u : User) {
    output(u.name)
    output(u.authored)
  }

  define page userList() {
    for (u : User) {
      navigate(viewUser(u)) { output(u.name) }
    }
  }
```

Access Control

Mismatch with Seam Access Control Model

- inflexible built-in policies
- limited to controller

Data Model Modularity

```
module usermanagement

  entity User {
    username :: String
    password :: Secret
  }
```

Data Model Modularity

```
module usermanagement

  entity User {
    username :: String
    password :: Secret
  }
```

```
module paper

  entity Paper {
    title :: String
    authors -> Set<User>
    abstract :: Text
  }
```

Data Model Modularity

```
module usermanagement

  entity User {
    username :: String
    password :: Secret
    authoredPapers -> Set<Paper>
  }
```

```
module paper

  entity Paper {
    title :: String
    authors -> Set<User>
    abstract :: Text
  }
```

Data Model Modularity

```
module usermanagement

  entity User {
    username :: String
    password :: Secret
    authoredPapers -> Set<Paper>
  }
```

```
module paper

  entity Paper {
    title :: String
    authors -> Set<User>
    abstract :: Text
  }
```

```
module access control

  rule page viewPaper(p : Paper) {
    principal in p.viewAccess
  }
```

Data Model Modularity

```
module usermanagement

  entity User {
    username :: String
    password :: Secret
    authoredPapers -> Set<Paper>
  }
```

```
module paper

  entity Paper {
    title :: String
    authors -> Set<User>
    abstract :: Text
    viewAccess -> Set<User>
  }
```

```
module access control

  rule page viewPaper(p : Paper) {
    principal in p.viewAccess
  }
```

Data Model Modularity

```
module usermanagement

  entity User {
    username :: String
    password :: Secret
  }
```

```
module paper

  entity Paper {
    title :: String
    authors -> Set<User>
    abstract :: Text
  }
```

```
  extend entity User {
     authoredPapers -> Set<Paper>
  }
```

```
module access control

  rule page viewPaper(p : Paper) {
    principal in p.viewAccess
  }
```

```
  extend entity Paper {
    viewAccess -> Set<User>
  }
```

Data Model Modularity

Mismatch with Hibernate and Java

Entity → Java Hibernate class

Java does not support partial classes

Scenarios

Just don't do it

Model Transformation

Adapt the Platform

Replace the Platform

Just don't do it

Just don't do it

access control

templates

data modularity

limit policy support

template inheritance

don't do it

Just don't do it

✔ inexpensive DSL development

✔ simple DSL development

✖ limit DSL potential

✖ possibly expensive application development

Model Transformation

Model Transformation

- ✔ independent of framework used
- ✔ uses same techniques as DSL generator
- ✖ significantly increases generated code size
- ✖ can lead to abstraction inversion

Adapt the Platform

Adapt the Platform

access control

templates

data modularity

extend existing library

adapt Facelets templates

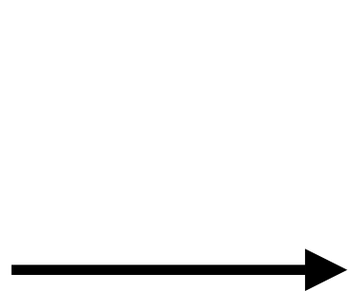
add partial classes to Java

Adapt the Platform

- ✔ DSL synchronized with platform
- ✔ DSL generation is simple
- ✖ keep update in sync with main framework developments
- ✖ convince framework developers to include changes

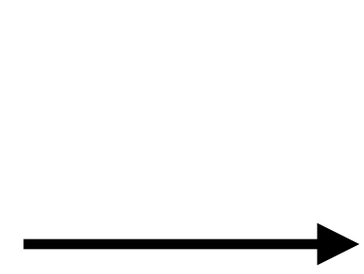
Replace the Platform

Replace the Platform



Plain Java servlets

Replace the Platform



Plain Java servlets

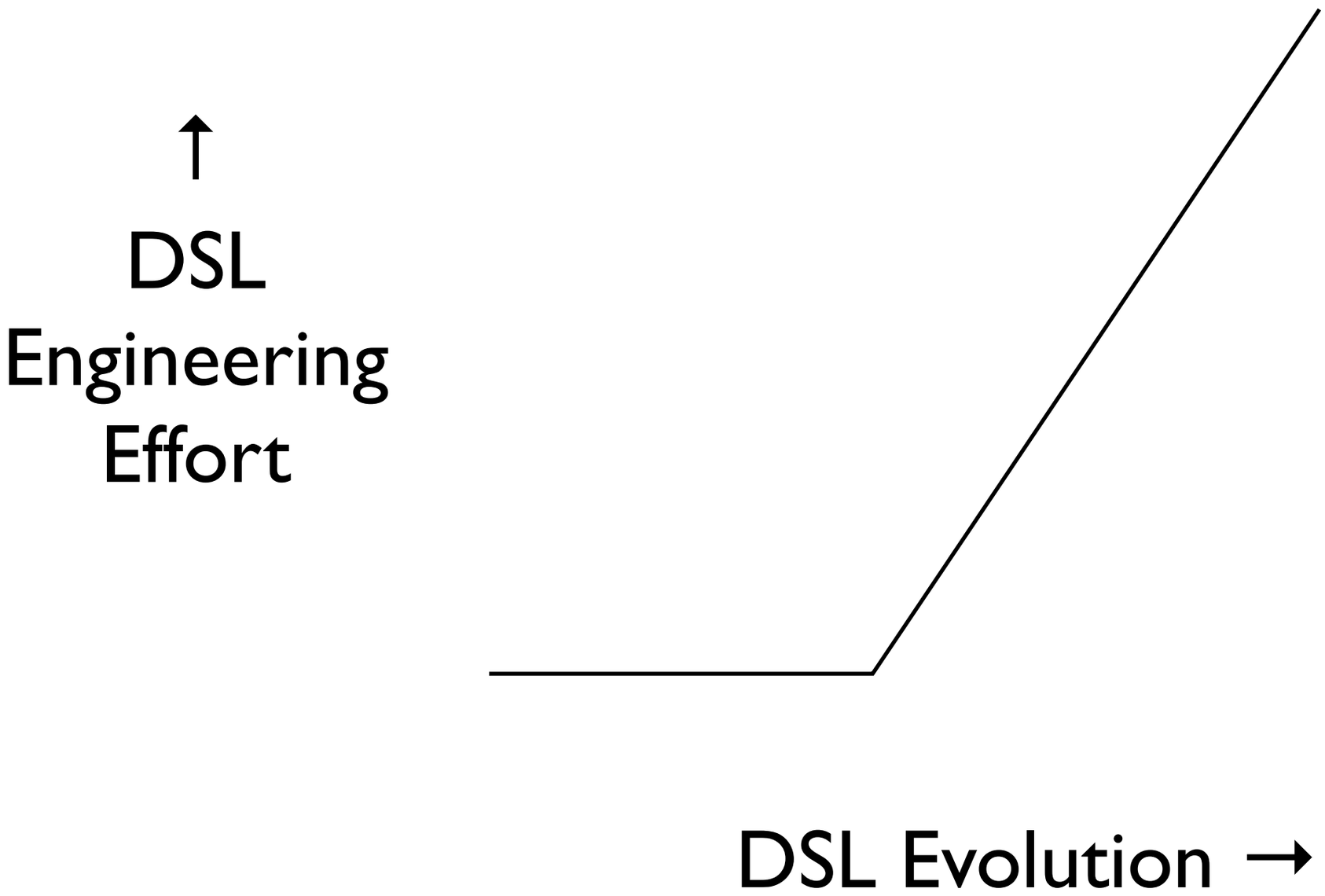
✔ one-to-one mapping

✔ better control of semantics

✖ lost effort

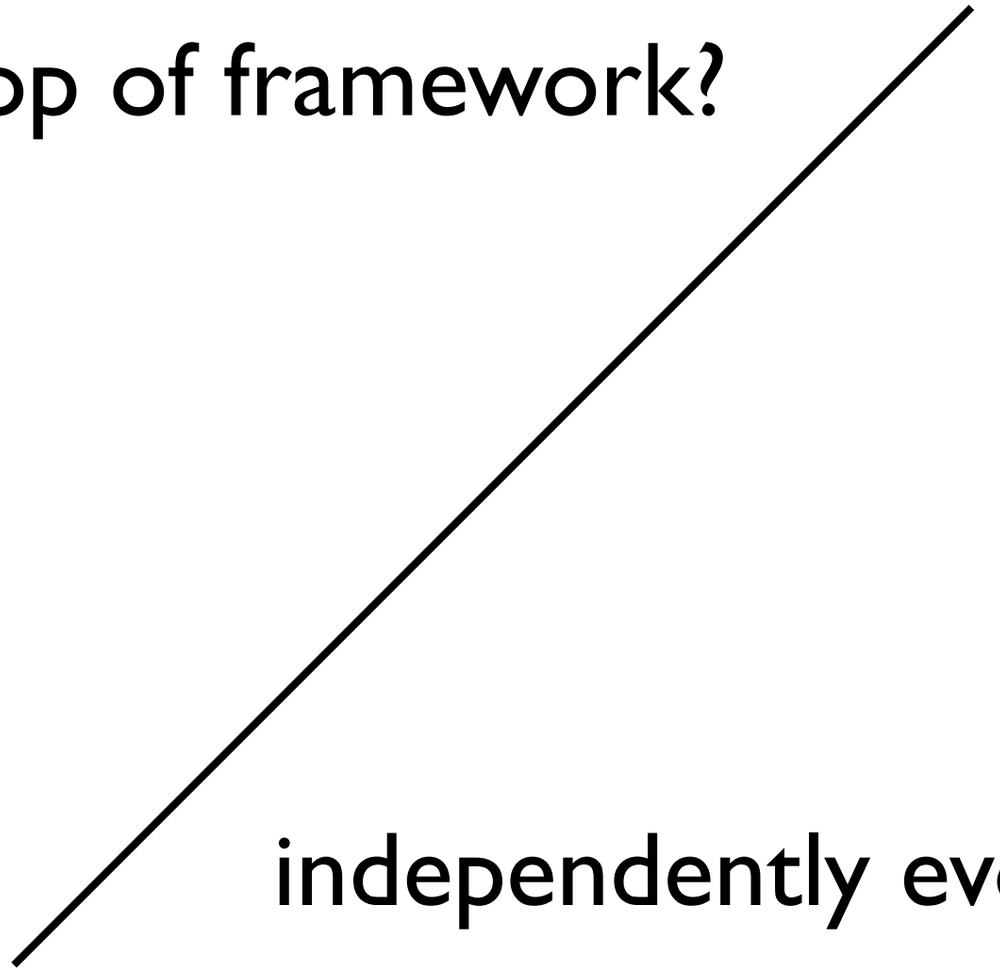
✖ reinvent the wheel

Conclusion



DSL

thin layer on top of framework?



independently evolving language?

webdsl.org

WebDSL Evolution

Initial Solutions

- Template Mechanism
- Access Control
- Data Model Modularity

} Model Transformation

Current Solutions

- Template Mechanism — Platform Replaced
- Access Control
- Data Model Modularity

} Model Transformation