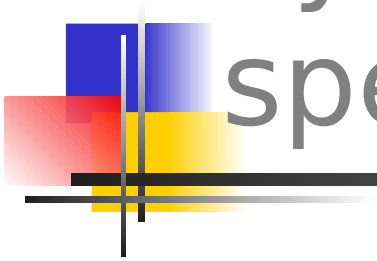
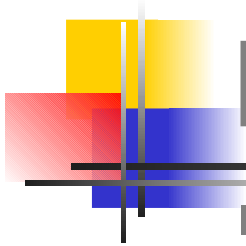


Python as a domain-specific language

Anders Hammarquist*,
Ronny Wikh, Jacob Oscarsson
*Contact: iko@strakt.com

STRAKT

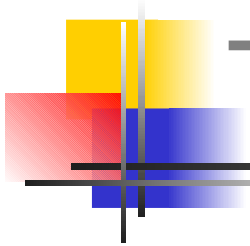
© 2006 AB Strakt



Background

- New BLM language for CAPS
  - previous too limited
- BLM language is used to declare business objects and program the business logic in CAPS
  - expected to be used by external people

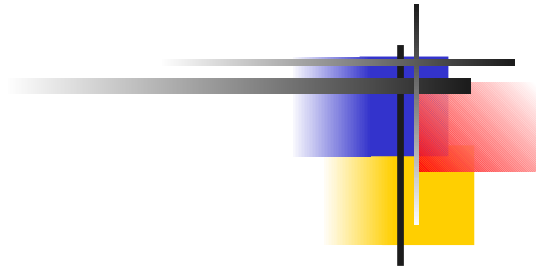
© 2006 AB Strakt



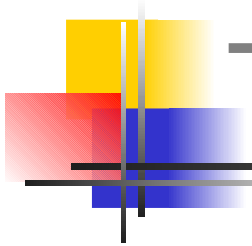
The old

- A custom language for declaring business objects
- only simple inheritance
- no multiple inheritance
- no polymorphism
- limited modularity
- heavy interdependence
- large source files!

© 2006 AB Strakt



```
toc CaseCategory(Category)
  "Category of Cases"

  attribute linkedCheck is Bool
    "Linked object check"
    restriction quantity(1)

    computation code
      return [ len( Case.query(
          category=self).run()) != 0 ]
    %%
```



The new

- Better inheritance model
- Multiple inheritance (mixins)
- Polymorphism
- Less interdependence
- Smaller source files
- pick and mix for building applications
- Introspection
- Easier testing

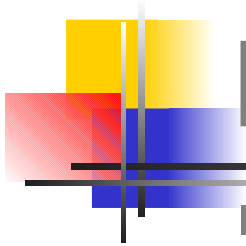
© 2006 AB Strakt

Why not Python?

Why not just use Python?

?

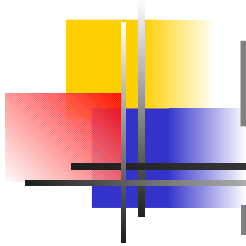
© 2006 AB Strakt



Problems

- No typing constructs
  - needed for database model
- Can't invent keywords
  - can we fit our model in the current language?
- You're supposed to use domain-specific languages!

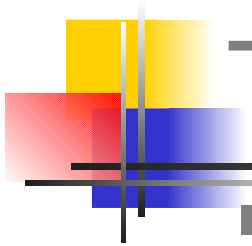
© 2006 AB Strakt



Benefits

- Python has the features we were lacking
  - Multiple inheritance
  - Polymorphism
  - Introspection
  - Testing frameworks
- We'll benefit from future development
- Only one language to learn

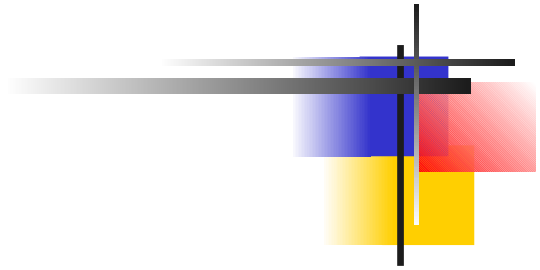
© 2006 AB Strakt



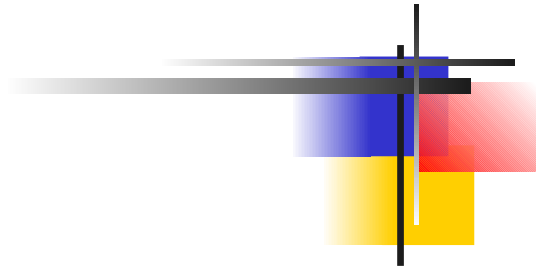
Tools of the trade

- Metaclasses
  - __extend__ (PyPy) to patch a class
  - Typing of object attributes
  - Syntax checking
- “Metamodules”
- General syntax abuse
  - unexpected instantiations
  - default args = type specifications
- Decorators

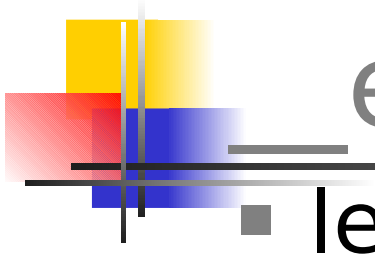
© 2006 AB Strakt



```
class CaseCategory(category.Category):
  "Category of Cases"

  class linkedCheck(Bool(Quantity(1))):
    "Linked object check"

    def on_computation(attr, self):
      return [ len(Case._query(
        category=self).run()) != 0 ]
```



```
toc CaseCategory(Category)
  "Category of Cases"

  attribute linkedCheck is Bool
    "Linked object check"
    restriction quantity(1)

    computation code
      return [ len( Case.query(
          category=self).run()) != 0 ]
    %%
```



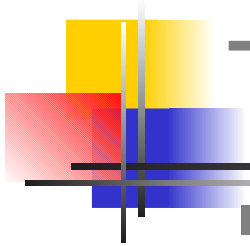
__extend__

- lets us extend (patch) a class from a different module
- gives more mix & match flexibility
- Metaclass detects __extend__ and patches existing class

comment.py defines toc Comment

```
class __extend__(people.Person):
  class comments(Relation(Weak())):
    "Comments"
    related = Comment.parent
```


© 2006 AB Strakt

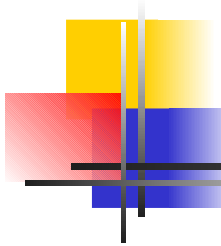


Typing attributes

- Declared using class in the TO-class
  - TO metaclass instantiates
  - descriptor, verifies data in __set__
- Subclass a data type instance(!)
  - Lets us add limitations

```
class linkedCheck(Bool(Quantity(1))):
```


© 2006 AB Strakt



Syntax checking

- Metaclass __new__ verifies sanity of the class
  - Loading before committing to DB gives compile-time checking
  - Things checked include
    - Only one data type in attributes
    - Base classes not used

© 2006 AB Strakt

“Metamodules”

- Our purpose similar to metaclasses
- Implemented by postprocessing of loaded modules
- Needed to deal with module-level attributes and methods

© 2006 AB Strakt

External (remote) methods

- Needs type information
- decorator for return type
- default arguments for call types

```
@method(Int())
fooMethod(self, fooArg=String()):
   return [ 42 ]
```

- Attribute type instances reused
- Introspection to get argument type
  - types checked on call & return

© 2006 AB Strakt

What we did

- Python
  - decorators
  - metaclasses
  - introspection
- BLM language
  - typed data + checking
  - syntax checking
  - domain-specific Python

STRAKT

© 2006 AB Strakt

What did we get?

- Python with some strange conventions
- Lost some detail (everything is just “class”)
- Gained all Python features
  - no need to emulate
- Cleaner code
- Modular code

© 2006 AB Strakt

Questions?

- Details of the how?
  - Complete would need a tutorial
- How long did it take?
  - about 6 months, but that includes other changes. Initial “language” took about 2 weeks.
- ???

© 2006 AB Strakt

The End

Anders Hammarquist
iko@strakt.com

STRAKT

© 2006 AB Strakt