specify. simplify. explore.

with ComplexValues

counseling developer Thomas J. Schrader

Smalltalked Visuals GmbH Christian Haider

data workflows fragile
systems complicated
maintenance difficult

?

something missing in
conventional OO

Values - functional style makes a difference

„values | objects“ !

[MacLennan, 1982]

Value is abstract concept

42

abstraction
no lifecycle
stateless
context-free

there are Standard Values

Immediates

SmallInteger 42, Character $a

Literal

Float 13.5, Symbol #none

String 'abc', Array #(1 'xyz' #one)

Value like

Point 1@20, Association #abc -> 42

ColorValue (ColorValue red: 1 green: 0 blue: 0)

a complex Value

```
ChartText
    style: (Textstyle
        color: (CmykColor
            cyan: 1
            magenta: 0.3
            yellow: 0
            black: 0.3)
        font: #{Helvetica}
        size: 12)
    string: 'This is a text'
    position: 5 @ 10
```

a complex Value specified

```
ChartText
    style: (Textstyle ..)
    string: 'This is a text'
    position: 5 @ 10

ChartText class>>localSpecification
        <constant: #style class: #{Textstyle}>
        <constant: #string class: #{String}>
        <optional: #position class: #{Point} default: '0@0'>
        <optional: #kerning class: #{Number} default: '0'>
```

ComplexValue is immutable composite

top down tree

```
Text
  string: 'ComplexValue...'
  style:  (Textstyle
   font: #{Helvetica}
   size:  60)
```

ComplexValues are
real objects
but without identity

Value = behavior + content

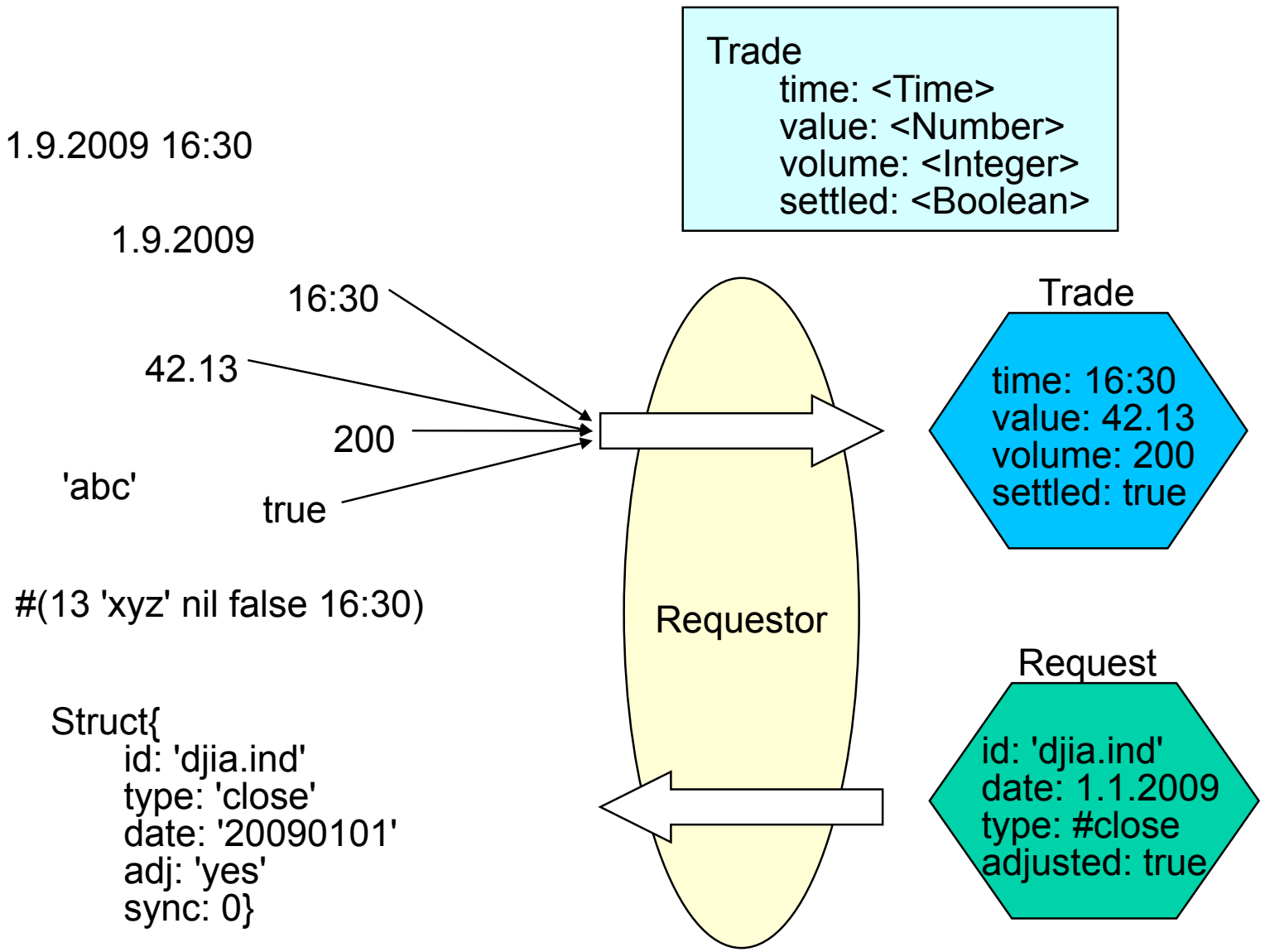
same class & content = same Value

ComplexValue is generated from a specification

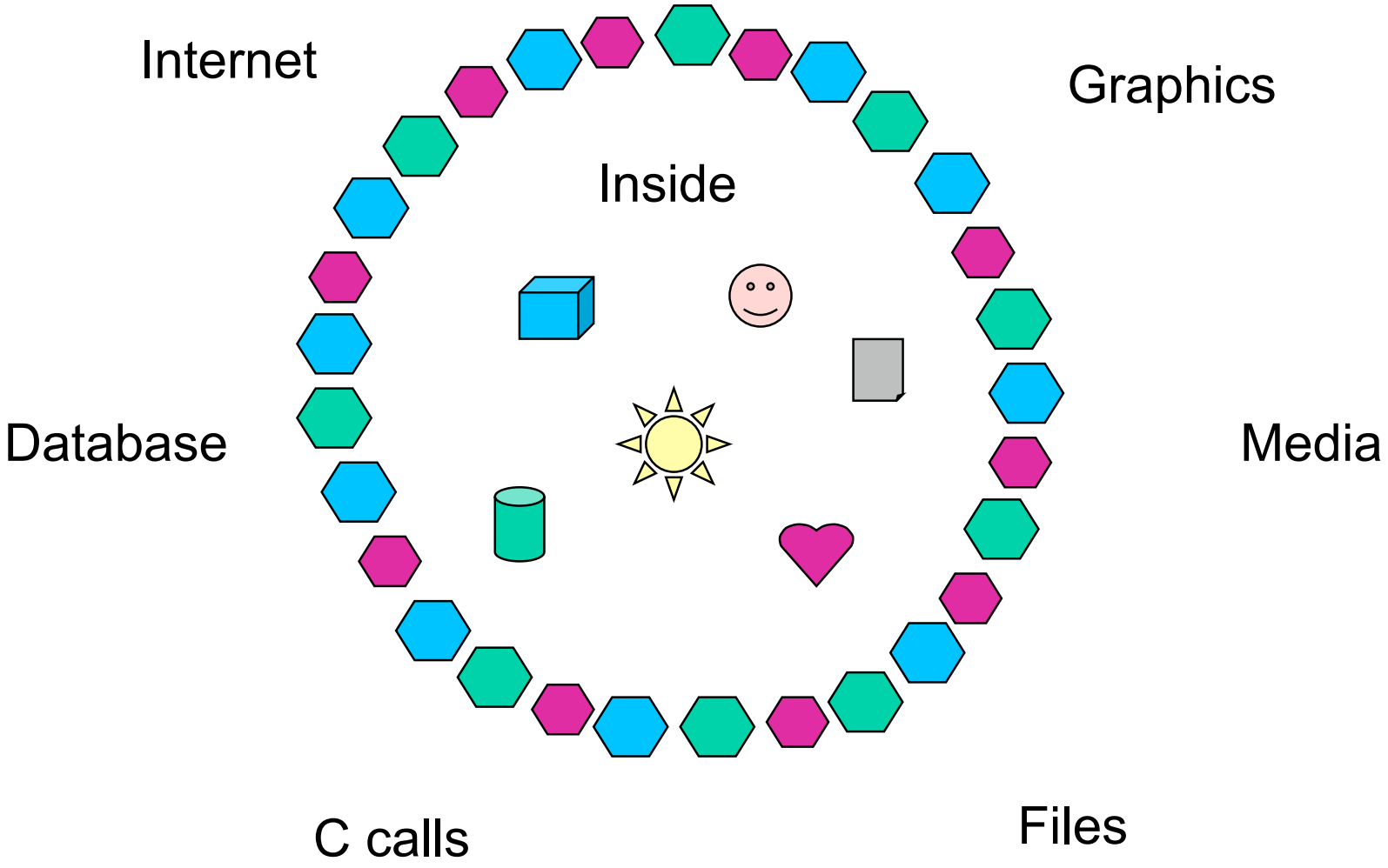
just one specification per Value

AValue class>>localSpecification

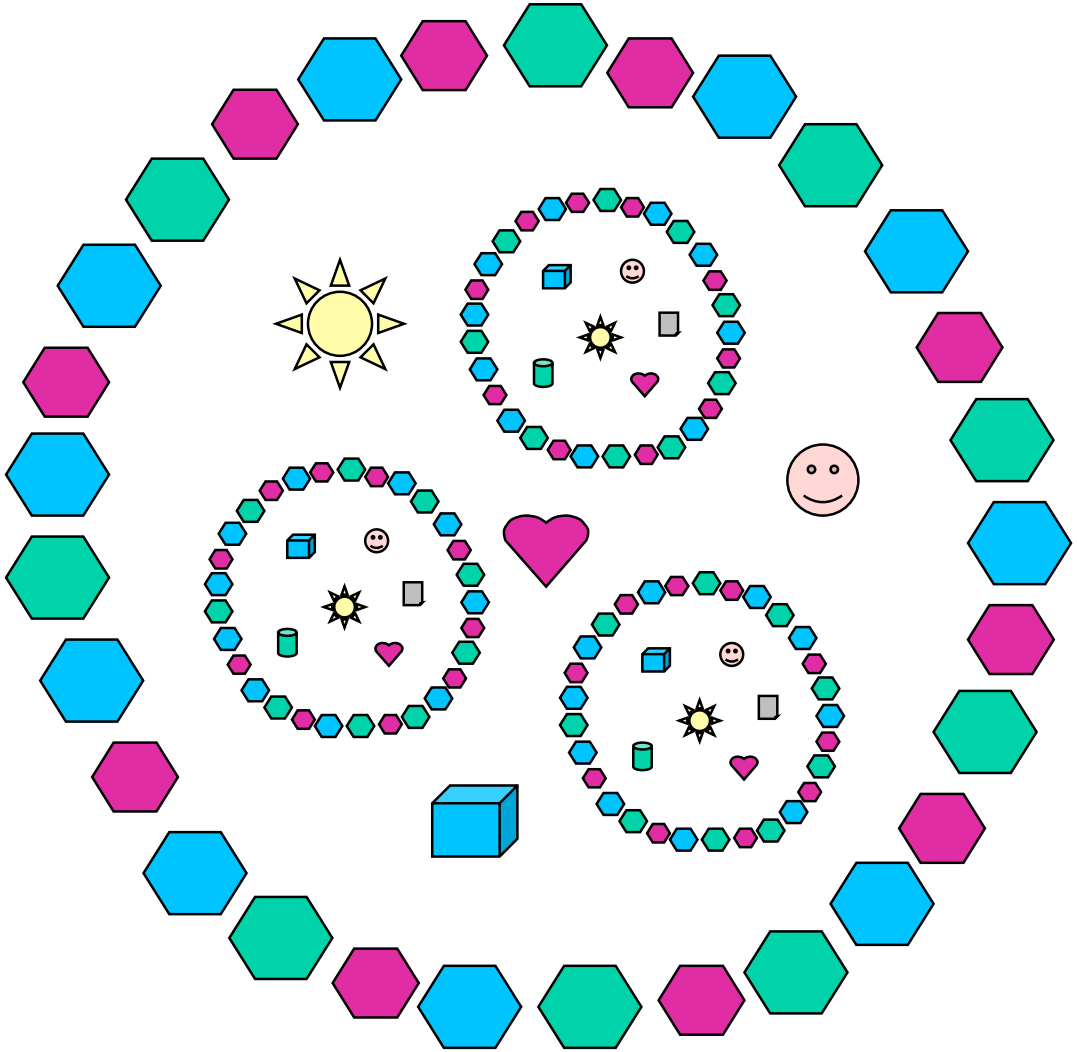
Interface


Trade
time: <Time>
value: <Number>
volume: <Integer>
settled: <Boolean>
1.9.2009 16:30
1.9.2009
16:30
42.13
200
'abc'
true
#(13 'xyz' nil false 16:30)
Struct{
id: 'djia.ind'
type: 'close'
date: '20090101'
adj: 'yes'
sync: 0}
Requestor
Trade
time: 16:30
value: 42.13
volume: 200
settled: true
Request
id: 'djia.ind'
date: 1.1.2009
type: #close
adjusted: true

System Interface



Module Interfaces



Configuration

```
#(#{UI.FullSpec}
    #window:
    #(#{UI.WindowSpec}
     #label: #(#{Kernel.UserMessage}
       #key: #UnlabeledCanvas …)
     #bounds: #(#{Graphics.Rectangle} …) )
    #component:
    #(#{UI.SpecCollection}
     #collection: #(
      #(#{UI.TextEditorSpec}
        #layout: #(#{Graphics.LayoutFrame} … )
        #name: #textEditor
        #model: #textHolder
        #isReadOnly: true
        #tabRequiresControl: true ) ) ) )
```

```
FullSpec
    window: (WindowSpec
        label: (UserMessage
            key: #UnlabeledCanvas …)
        bounds: (Rectangle …))
    component: (SpecCollection
        collection: (Array
            with: (TextEditorSpec
                layout: (LayoutFrame …)
                name: #textEditor
                model: #textHolder
                isReadOnly: true
                tabRequiresControl: true)))
```

VW Setting

```
<?xml version="1.0"?>
<settings domain="VisualWorksSettings">
  <setting>
    <id>
      <key>tools</key>
      <key>browser</key>
      <key>defaultBrowserType</key>
    </id>
    <state>
      <choice-key>Package</choice-key>
    </state>
  </setting>
</settings>
```

```
Settings
        domain: 'VisualWorksSettings'
        setting: (Id
                with: #tools
                with: #browser
                with: #defaultBrowserType)
        state: (ChoiceKey value: 'Package')
```

Why we like Values

adequate

simple and reliable

pretty

practical

Thank you!

Questions?

References

Store

{Values Development}
Cincom Public Store

Technical Article

“Complex Values In Smalltalk”, ESUG 09

Contact

christian.haider@smalltalked-visuals.com
thomas.j.schrader@web.de

Defining a Value

```
localSpecification
    <constant: #constant class: #{Symbol}>
    <optional: #optional class: #{Symbol} default: '#a'>
    <sequence: #array>
    <map: #dictionary>
```

Constructor

```
constant: const optional: opt array: arr dictionary: dict
    | inst |
    inst := self new.
    inst
        initializeConstant: const
        optional: opt
        array: arr
        dictionary: dict.
    ^inst
```

Optional Constructors

```
constant: const
    | inst |
    inst := self new.
    inst initializeConstant: const optional: nil array: nil dictionary: nil.
    ^inst
constant: const optional: opt                          (…)
constant: const optional: opt array: arr               (…)
constant: const optional: opt dictionary: dict         (…)
constant: const array: arr                             (…)
constant: const array: arr dictionary: dict            (…)
constant: const dictionary: dict                       (…)
```

Initializer

```
initializeConstant: const optional: opt array: arr dictionary: dict
    constant := const.

    (opt notNil and: [self optional ~= opt]) ifTrue: [
        optional := opt].

    (arr notNil and: [arr notEmpty]) ifTrue: [
        array := (Array withAll: arr) beImmutable].

    (dict notNil and: [dict notEmpty]) ifTrue: [
        | od |
        od := OrderedDictionary new.
        dict keysAndValuesDo: [:key :value | od at: key put: value].
        dictionary := od beImmutable].

    self beImmutable
```

Accessors

```
constant
"<Symbol>"
    ^constant

optional
    "<Symbol>"
    ^optional ifNil: [#a]
```

```
array
    "<Array>"
    ^array ifNil: [#()]

dictionary
    "<Dictionary>"
    ^dictionary ifNil: [
        Dictionary new
                    beImmutable]
```

Printer

```
printvalueWith: printer
    | args |
    args := OrderedCollection new.
    args add: (printer constant: 'constant' value: self constant).
    args add: (printer optional: 'optional' value: optional).
    args add: (printer array: 'array' value: self array).
    args add: (printer dictionary: 'dictionary' value: self dictionary).
    ^printer printvalue: self arguments: args
```

Opentalk Service

```
passInstVars
   "for OpenTalk StSt"
   ^#(#default #default #default #value)
```