porting from VisualWorks®
started in late 2016 by Richard Uttner

- support of an existing application
  - per documaps
  - document management

- target:
  - domain-specific solutions
  - database-centric
  - fat clients with platform-specific GUI
goals

- layers separation
- minimize redundancy, improve re-usability
- minimize work required for the UI and “glue” layers
- improve testability
- support for the common business application patterns
- ...

production quality small business application in few days
participants and roles

- **modelling level:**
  - parts forming the “skeleton” of the application
  - objects exposed by parts (conditions, values, actions)

- **UI level:**
  - generic GUI clients of the framework
  - application specific objects (widget, models,...)

- **“glue” level:**
  - arbitrary objects (startup, environment, use cases,...)
Clients separation
Parts

- data
- aspects (redirections)
- conditions
  - a user can see the reason why something is disabled
    
    \[(\text{APCondition on: } [\text{stringField size} > 0] \text{ ifNot: } \#\text{StringFieldIsEmpty}) & \text{self enablementInputEnabled})\]

- actions, triggers
- subpatrs
aPart

- predefined parts (for lists, trees...)
- enumerations (combo-boxes, menus...)
- prompts, modal windows
- layouts
- Glorp
- Trachel
use cases

- connected via:
  - their creators (typically parts)
  - framework condition objects
  - part actions
  - callbacks to parts maintaining separation from GUI

- do not reference parts, initialized with objects needed for running
- named callback requests (from the creator)

- workflow editor
clients

- typically just GUI clients
- headless clients, forwarding clients...
- particularly useful for tests
- aPart entities with native support of state changes records
- recording client
automatic unit tests code generation

- open recording client on your part with SUTAPIInteractionPrinter, manual interaction
- use the generated SUnit test code
- unit tests repeat the interactions, check states
- prompts/multiple windows support

```smalltalk
self afterDoing: [
    self setAspect: #stringField value: 'foo'. ]

expectStates: [
    APExpectedStates
    expectAllInactive: #( #clearNumber #confirmNumber #saveData)
    expectAllActive: #( #clearString #confirmString
        #disableInput #intField #stringField)
].```
framework usage

- developers should prefer the framework
- not always the most straightforward solution
framework usage

- locate functionality?
- dependencies?
- estimations?
- impacts?

- plan
- tools
- additional effort
from VisualWorks
part of porting of *per documaps* application
close collaboration with Pharo Consortium and INRIA (RMoD)
Challenges

- Language differences
  - namespaces
    
    Store.Model
    UI.Model
  - qualified literals
    
    `{UI.CheckBoxSpec}
  - FFI calls
    
    `<C:typedef int64_t (*callb_after_send_t)(unsigned char* handlerID, int PortServerID, unsigned char* inputBuffer, int cbInput)>"
Challenges

- Semantics differences
  - object initialization (new)
    - inherit from class that behave differently
  - same methods with different behavior (Pragma>>#selector)
  - dependencies mechanism
  - (#Smalltalk = 'Smalltalk') = false
  - 'asdf' readStream upToAll: 'd'; upToEnd
    - 'f' in Pharo, 'df' in VW
  - and many more...
Challenges

- Different code management tools, source formats
  - VW: Store, XML
  - Pharo: Git

- Completely different UI framework
  - UI Painter, different data flow
    - VW: Aspect adaptors
    - Pharo: Value holders
Challenges

- Application strongly Windows oriented
  - first-and-half-class citizen in Pharo

- Application still under active development
  - bi-directional transformation

- Target platform under active development
  - Spec2
Why should you care?

- Most of us will port applications to Pharo

...if you are lucky, from some older Pharo version
Approach

- export code from VW in XML form (*.pst)
  (not stable order, unusable for versioning)
- import into Ring2 model
  – modified scanner & parser
- apply well known code transformations
- store VW metadata (for reverse direction)
- save Ring2 model in Tonel format
- manage with Git
- load into Pharo
Tests!

- with so many small hidden incompatibilities, tests are absolutely necessary
- good code coverage, mutation testing, UI tests
- cheap in long term
UI layers adoption

- **VisualWorks**
  - ApplicationModel
  - UIBuilder
  - InputFieldSpec
  - InputFieldView
  - APValueAdaptor

  
  ![Diagram](Diagram.png)

- **Pharo**
  - “compatible” ApplicationModel
  - UIBuilder replacement

  ![Diagram](Diagram.png)
Future

- aPart Framework will be open-sourced
- native UI with Spec2 (GTK)
- full-featured reference example including database handling with Glorp
- extended documentation
- workflow editor...
the side-effect of SCHMIDT, Pharo Consortium and INRIA collaboration

better Pharo

for you business