CoreObject & EtoileUI

Swansea 2009

Étoilé

A desktop environment built around:

- Pervasive Data Sharing & Versioning
- Composite Document
- Collaboration
- Document-oriented

Raskin's First Law

A computer shall not harm your work or, through inaction, allow your work to come to harm.

Versioning

Makes the user more at ease with:

- No save
- Document History
- Undo/Redo on all persistent data
- Versioning that scales to video, image, etc.

Raskin's Second Law

- A computer shall not waste your time or require you to do more work than is strictly necessary

~~Import/Export/Convert~~

- No document or content export/import necessary within Étoilé
- Import/export for communicating with the outside world is built in

Data Sharing

Eliminates name service mulplication.

Shared content access is about NewtonSoup-like properties or attaching metadatas.

- We need something like a filesystem but with:
  - real semantic
  - fine-grained structure access
  - multiple views or organization levels

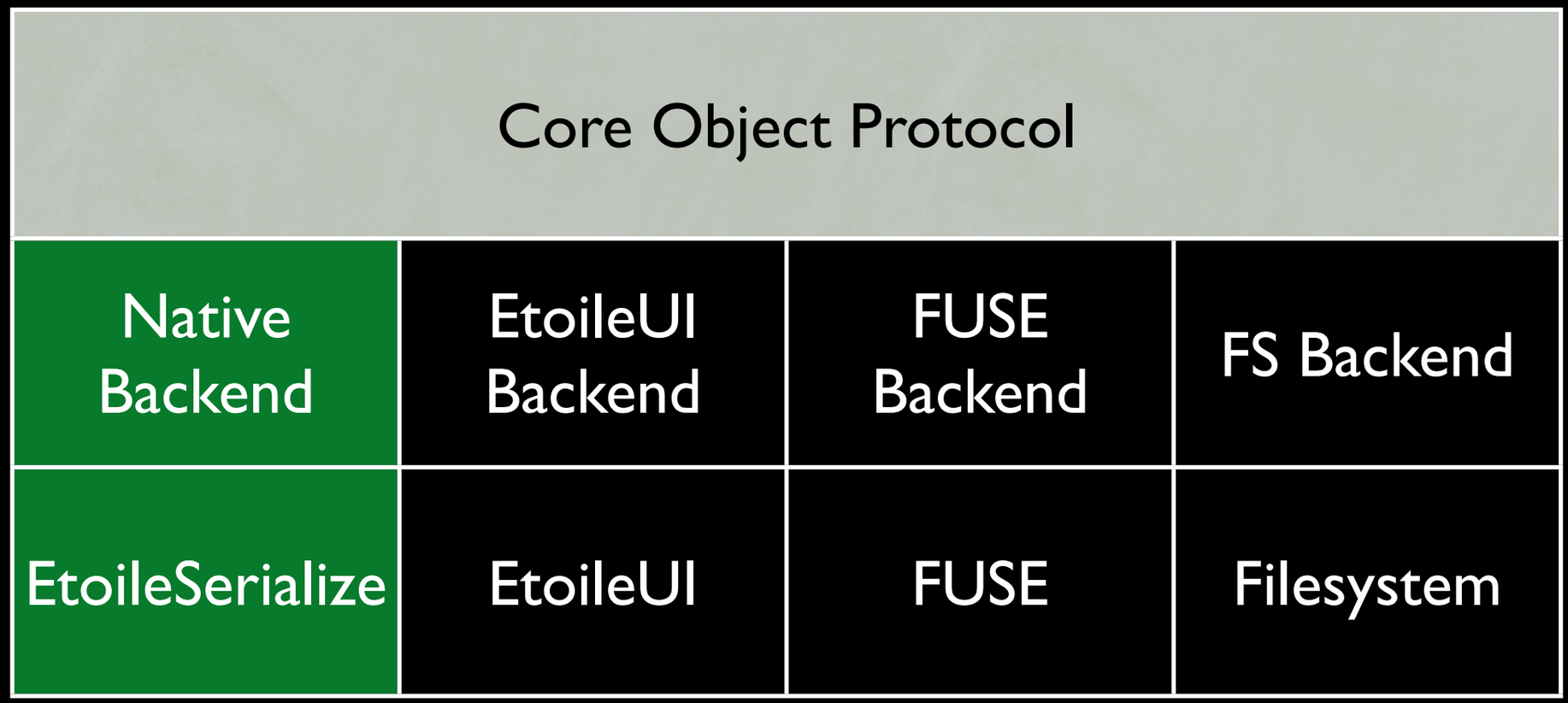
CoreObject Protocol

The protocol role is twofold:

- organize objects and documents
- expose internal document structure or object content

CoreObject

EtoileUI backend exposes composite document structure in term of CO interfaces.



Object Store

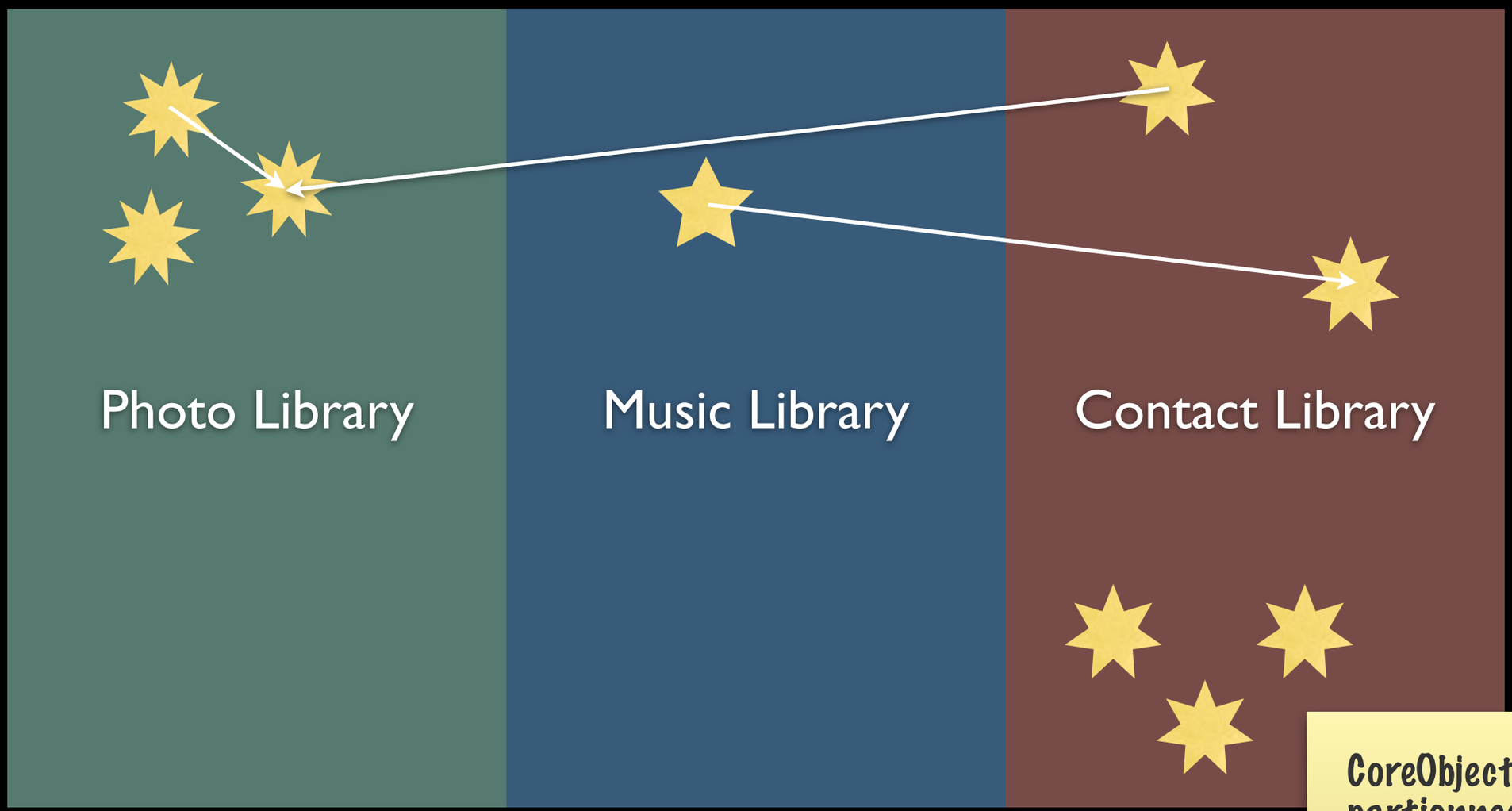
Follows prevalence model.

- No Object-Relational-Mapping
- Stores changes as logical operations with:
  - serialized messages
  - snapshots
- Inspired by NewtonSoup
- Uses a SQL database as metadata server

Multi-level Versioning

- Fine-grained versionning with various levels:
  - Global (private)
  - Context
  - Persistent Root

Object Contexts



⟶ Relationships between persistent roots

CoreObjectGraph partionned into object contexts

Example

```
COGroup *library = [[COGroup alloc] init];

ETMusicTrack *track = [[ETMusicTrack alloc] init];

[track setValue: @”More Flowers” forProperty:
kETAlbumName];

[group addMember: track];


COGroup *playlist = [[COGroup alloc] init];

[library addMember: playlist];
```

http://etoile-project.org/

Jesse Ross

jesse@je...

All

Inbox

Events

People

Trash

Shelf

How it began

- Why UI in Photoshop and IDE are so rigid?
- Unifying Pane-based UI with PaneKit
- AppKit is great but NSView and NSCell hierarchy doesn't scale
- AppKit API is too big for my mind :-)

What does it solve?

- Generic protocol for Structured Document
- Building blocks for Graphics Editor
- Complex widget development
- Zero UI glue code
- UI Plasticity

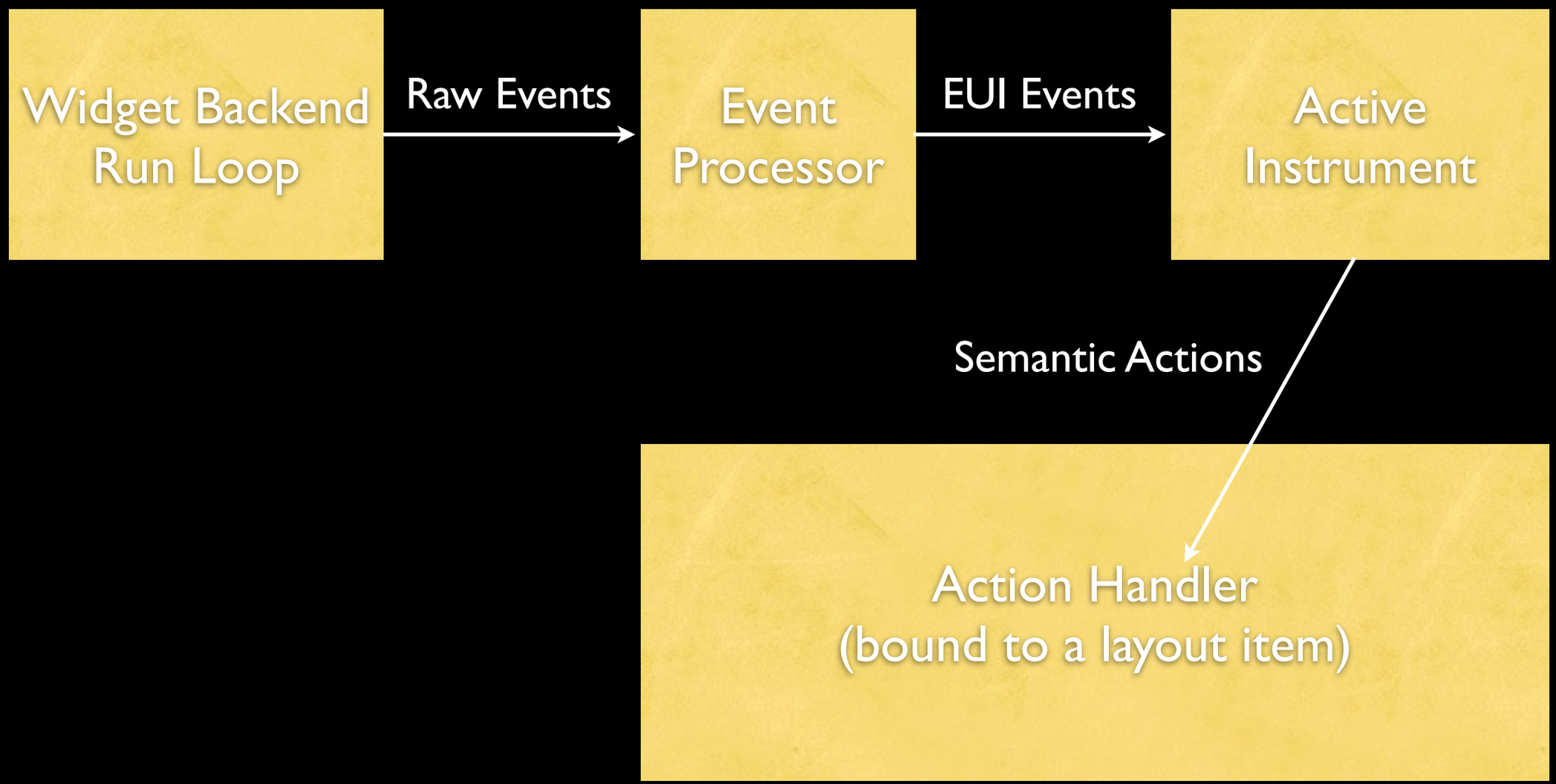
Instrument Examaple

- ETSelectTool *tool = [[mainViewItem layout] attachedInstrument];
- [[tool selectionAreaItem] setShape: [ETShape circle];

Turtles all the way down

- Everything is a layout item
  - selection rectangle
  - handles
  - shapes
  - windows
  - layers

From Events to Actions



What do we gain?

- Input Device Indepent
- Multi-instruments Interaction
  - one per input device (e.g bimanual interaction)
  - one per user (e.g. collaboration)
- Ability to operate over process boundaries

Separation of Concerns

- No monolithic view/wigdet, but rather…
- UI aspects stored in a repository
  - Styles
  - Layouts
  - Action Handlers
  - Views