ÉTOILÉ STATUS UPDATE

FOSDEM 2011

http://www.etoileos.com

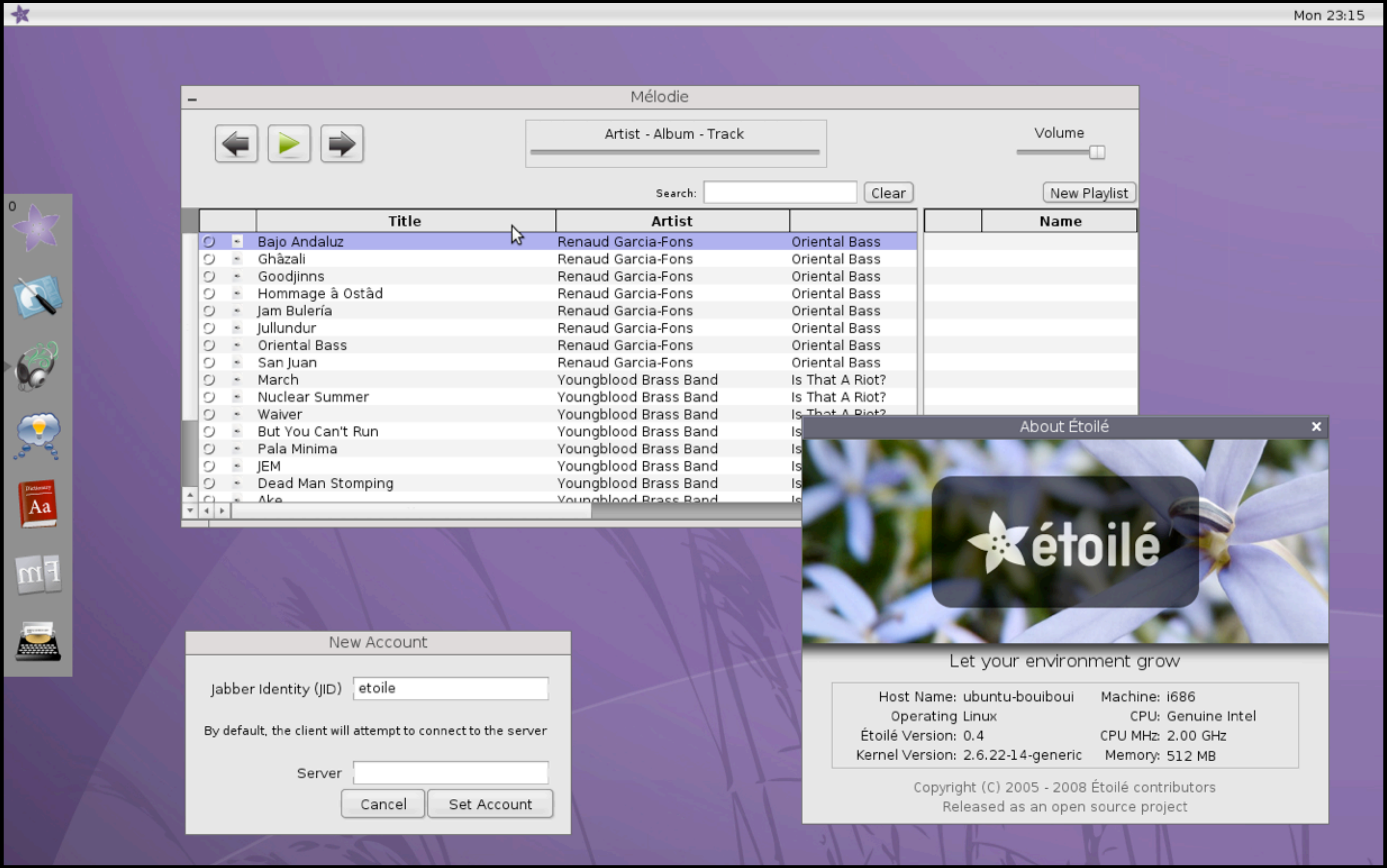
What is it?

- Étoilé is a user environment designed from the ground up around the things people do with computers: create, collaborate, and learn.

Goals

- Composite Document
- Collaboration
- Persistence & Versioning
- Clean, consistent and plastic UI
- Fast and Easy Development

Étoilé 0.4



		Title	Artist	
○	▪	Bajo Andaluz	Renaud Garcia-Fons	Oriental Bass
○	▪	Ghâzali	Renaud Garcia-Fons	Oriental Bass
○	▪	Goodjinns	Renaud Garcia-Fons	Oriental Bass
○	▪	Hommage à Ostâd	Renaud Garcia-Fons	Oriental Bass
○	▪	Jam Bulería	Renaud Garcia-Fons	Oriental Bass
○	▪	Jullundur	Renaud Garcia-Fons	Oriental Bass
○	▪	Oriental Bass	Renaud Garcia-Fons	Oriental Bass
○	▪	San Juan	Renaud Garcia-Fons	Oriental Bass
○	▪	March	Youngblood Brass Band	Is That A Riot?
○	▪	Nuclear Summer	Youngblood Brass Band	Is That A Riot?
○	▪	Waiver	Youngblood Brass Band	Is That A Riot?
○	▪	But You Can't Run	Youngblood Brass Band	Is
○	▪	Pala Minima	Youngblood Brass Band	Is
○	▪	JEM	Youngblood Brass Band	Is
○	▪	Dead Man Stomping	Youngblood Brass Band	Is
○	▪	Ake	Youngblood Brass Band	Is

Jabber Identity (JID) etoile

By default, the client will attempt to connect to the server

Server

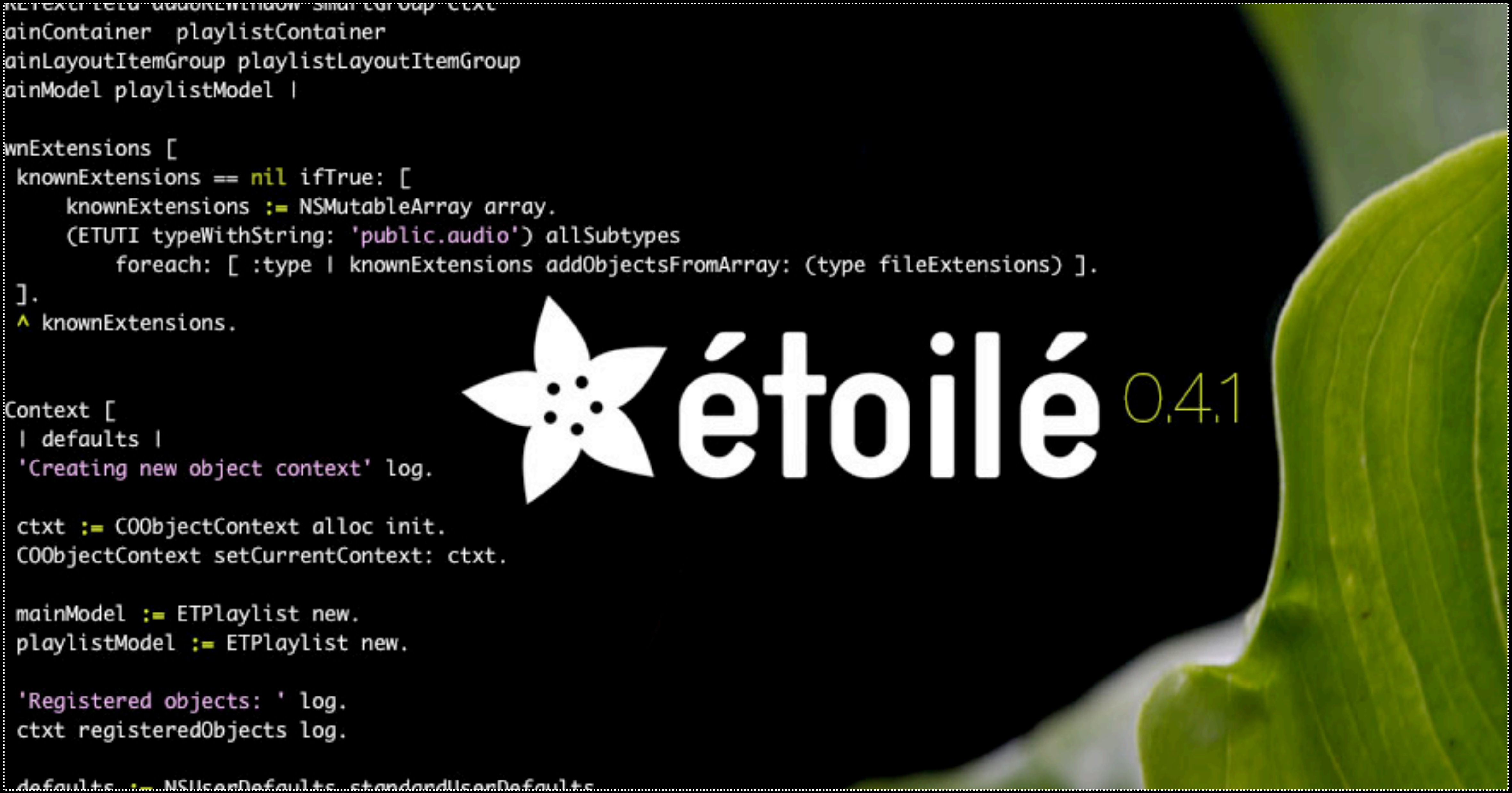
Let your environment grow

Host Name: ubuntu-bouiboui
Operating Linux
Étoilé Version: 0.4
Kernel Version: 2.6.22-14-generic

Machine: i686
CPU: Genuine Intel
CPU MHz: 2.00 GHz
Memory: 512 MB

Copyright (C) 2005 - 2008 Étoilé contributors
Released as an open source project

A while ago (or almost)



```
KETextField addOKEWindow smartGroup ctxt
ainContainer  playlistContainer
ainLayoutItemGroup playlistLayoutItemGroup
ainModel playlistModel |

wnExtensions [
 knownExtensions == nil ifTrue: [
     knownExtensions := NSMutableArray array.
     (ETUTI typeWithString: 'public.audio') allSubtypes
         foreach: [ :type | knownExtensions addObjectsFromArray: (type fileExtensions) ].
 ].
 ^ knownExtensions.


Context [
 | defaults |
 'Creating new object context' log.

 ctxt := COObjectContext alloc init.
 COObjectContext setCurrentContext: ctxt.

 mainModel := ETPlaylist new.
 playlistModel := ETPlaylist new.

 'Registered objects: ' log.
 ctxt registeredObjects log.

 defaults := NSUserDefaults standardUserDefaults.
```

Release Status

- 0.4.1 released in March 2009
  - not supported anymore
- 0.4.2 was expected in early 2010
- Was pushed back several times due to EtoileUI, libobjc2 and theming

Next Releases

- Switching to modular releases
- Roadmap remains identical
  - http://etoileos.com/dev/roadmap/
- Just releasing modules as soon as they are ready

0.4.2

- Previous Fall
  - libobjc2
- February/March
  - Foundation, DocGenerator, LanguageKit
- Spring
  - EtoileUI, Updated Theming

What's new since last year?

GNUstep Progresses

- libobjc2
- DBusKit
- Theming
- Opal (aka CoreGraphics)
- Image Drawing Improvement

Apple vs GNUstep

Apple Names	*GNUstep Names*
AppKit	Gui
CoreGraphics	Opal
Foundation	Base
CoreFoundation	CoreBase

libobjc2

- A new ObjC runtime
- Inspired by the Étoilé ObjC runtime
- Based on the new Apple runtime API
- Compatible with the old GNU runtime API

ObjC 2 Features

- Full ObjC 2 Support with Clang (or GCC 4.6?)
  - Non-fragile Instance Variables
  - Properties
  - Fast Enumeration and Proxy
  - Blocks
  - etc.

Additional Features

- Type-Dependent Dispatch
- Low-Memory Profile
- Object Planes
  - Message interception between group of objects
- Experimental LLVM Optimizations

libobjc2

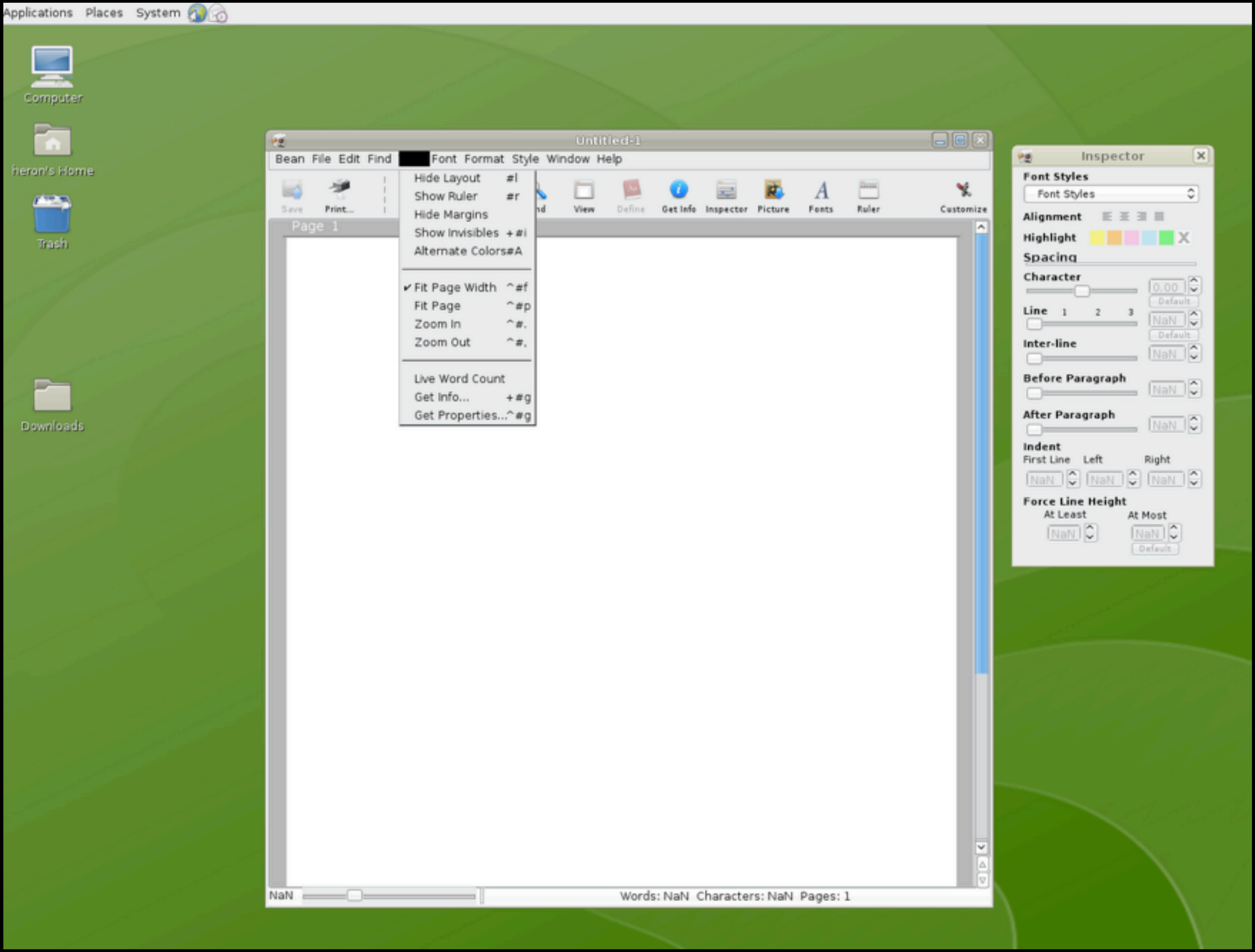
- Version 1 released in October
- 1.1 in January
- Now Stable and Documented

DBusKit

- ObjC API for Freedesktop DBus
- GNUstep Google SoC project
- More in the DBusKit talk later

Theming

- Most controls can be themed now
- Pixmap themes a la Camaleon
- … Camaelon is now dead and buried
- More in the Gorm & Theming talk later



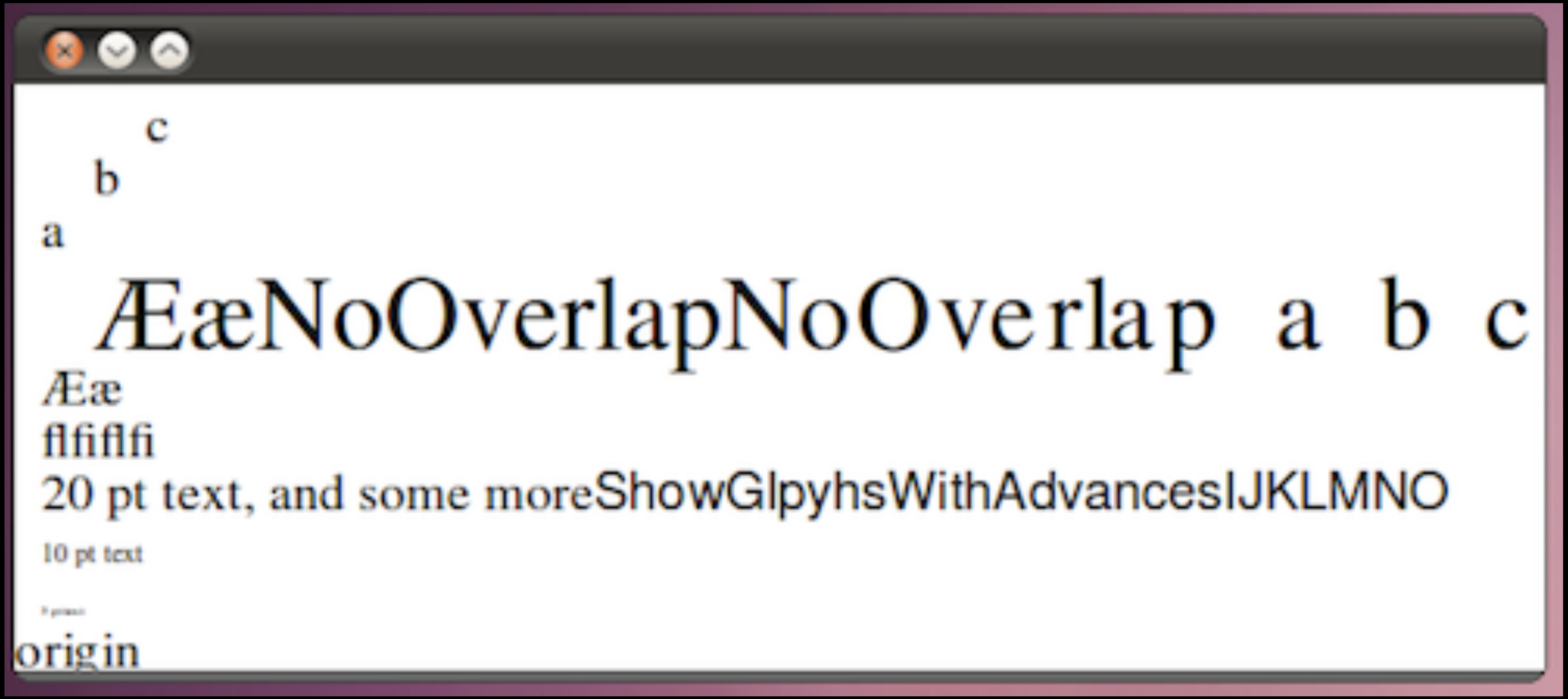
Bean Text Editor & GNOME theme

Opal

- CoreGraphics C API Implementation
- On top of Cairo
- Written in ObjC with Foundation
- GNUstep Google SoC project

More about Opal

- Image support
    - tiff, jpeg, and png
- Color Space & Transform support
    - based on lcms
- CoreText implementation underway



c

b

a

ÆæNoOverlapNoOverlap a b c

Ææ

ﬂﬁﬂﬁ

20 pt text, and some moreShowGlpyhsWithAdvancesIJKLMNO

10 pt text

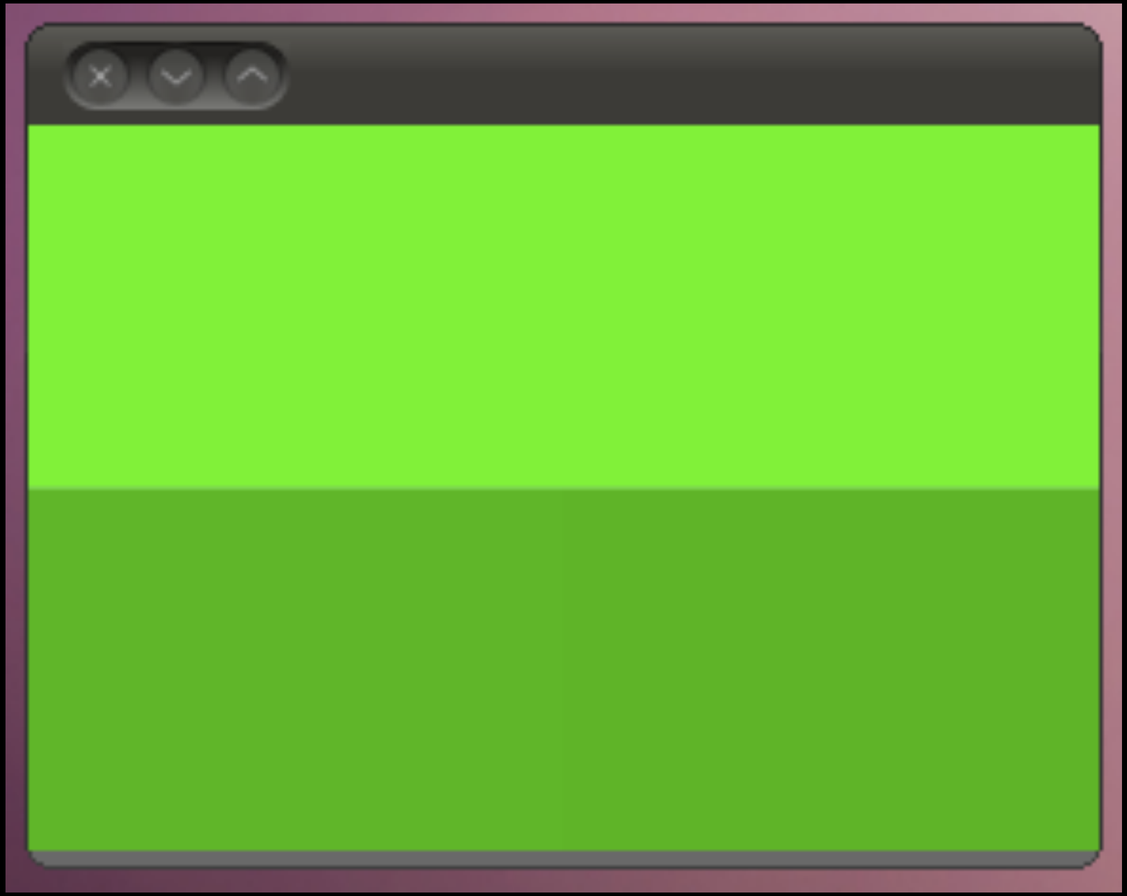
[illegible]

origin

Glyphs

&

Color Matching



Étoilé Progresses

- ProjectManager
- EtoileUI
- EtoileFoundation Metamodel
- ObjectMerging
- EtoileText
- SourceCodeKit

Project Manager

Project Manager

- For now a Compositing Window Manager
- Divided in two modules
  - XCBKit
  - Project Manager Framework
- Will coordinate Étoilé project model
  - project switching, persistency etc.

XCBKit

- ObjC API wrapping the X11 XCB library
- Built with Foundation
  - XCB protocol handled with a run loop
- Can be used to write Window Manager

EtoileText

What is EtoileText?

- Structured Text Editing Model
- Stores the whole text into a text tree that represents chapters, sections, paragraphs etc.
- When editing text, the text changes are applied at the tree level

Styling and Export

- Text tree nodes can be styled
- Presentational and semantic structure are keep apart
- For example
    - Can import TeX and output HTML

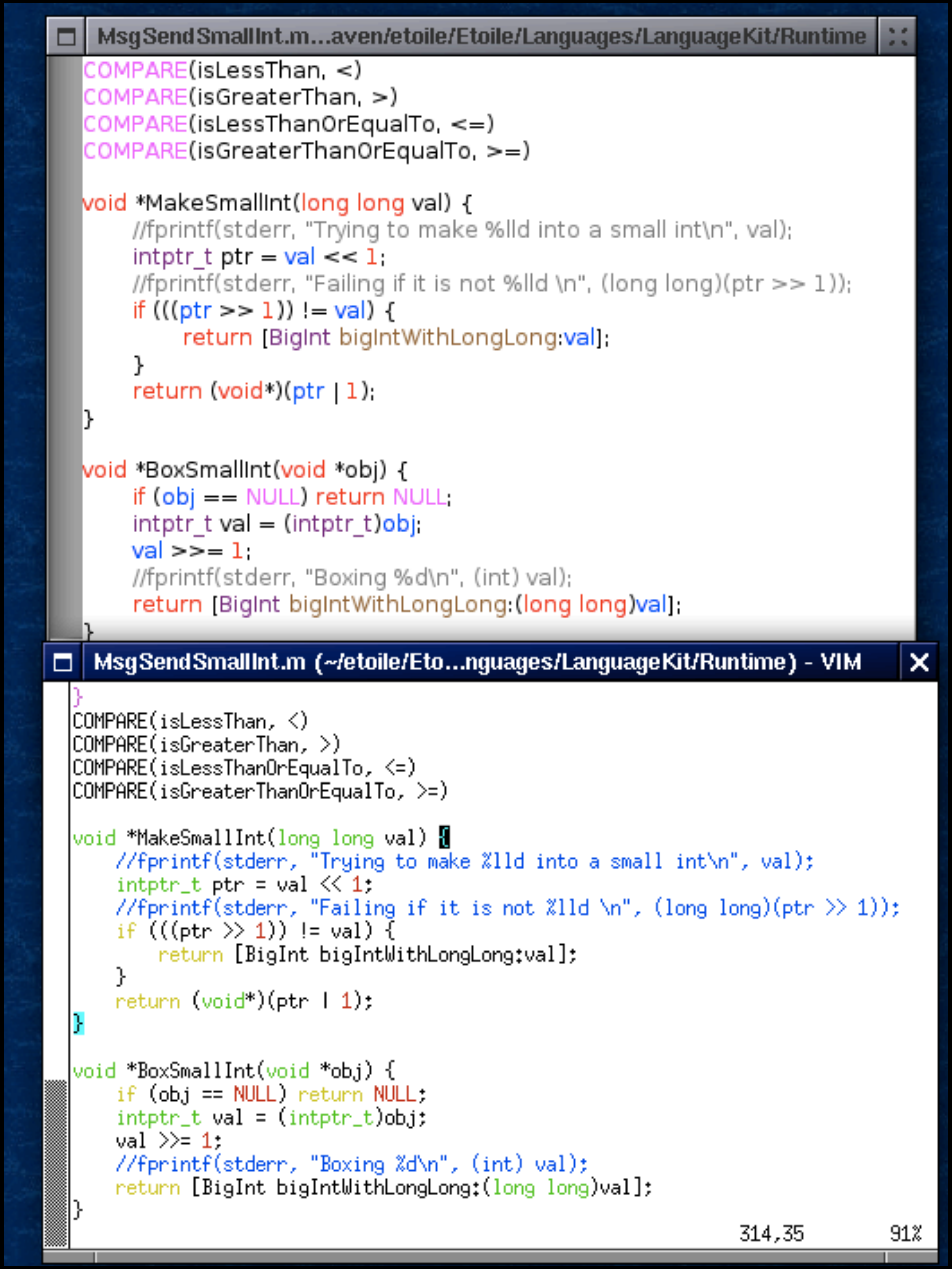
SourceCodeKit

SourceCodeKit

- Clang-driven
  - Code Indexing
  - Syntax Highlighting
- CodeMonkey will probably use it

Syntax Highlighting

SourceCodeKit
vs
VIM


MsgSendSmallInt.m...aven/etoile/Etoile/Languages/LanguageKit/Runtime
MsgSendSmallInt.m (~/etoile/Eto...nguages/LanguageKit/Runtime) - VIM

```
COMPARE(isLessThan, <)
COMPARE(isGreaterThan, >)
COMPARE(isLessThanOrEqualTo, <=)
COMPARE(isGreaterThanOrEqualTo, >=)

void *MakeSmallInt(long long val) {
	//fprintf(stderr, "Trying to make %lld into a small int\n", val);
	intptr_t ptr = val << 1;
	//fprintf(stderr, "Failing if it is not %lld \n", (long long)(ptr >> 1));
	if (((ptr >> 1)) != val) {
		return [BigInt bigIntWithLongLong:val];
	}
	return (void*)(ptr | 1);
}

void *BoxSmallInt(void *obj) {
	if (obj == NULL) return NULL;
	intptr_t val = (intptr_t)obj;
	val >>= 1;
	//fprintf(stderr, "Boxing %d\n", (int) val);
	return [BigInt bigIntWithLongLong:(long long)val];
}
```

```
}
COMPARE(isLessThan, <)
COMPARE(isGreaterThan, >)
COMPARE(isLessThanOrEqualTo, <=)
COMPARE(isGreaterThanOrEqualTo, >=)

void *MakeSmallInt(long long val) {
    //fprintf(stderr, "Trying to make %lld into a small int\n", val);
    intptr_t ptr = val << 1;
    //fprintf(stderr, "Failing if it is not %lld \n", (long long)(ptr >> 1));
    if (((ptr >> 1)) != val) {
        return [BigInt bigIntWithLongLong:val];
    }
    return (void*)(ptr | 1);
}

void *BoxSmallInt(void *obj) {
    if (obj == NULL) return NULL;
    intptr_t val = (intptr_t)obj;
    val >>= 1;
    //fprintf(stderr, "Boxing %d\n", (int) val);
    return [BigInt bigIntWithLongLong:(long long)val];
}
                                                              314,35        91%
```

DocGenerator

DocGenerator Goals

- Presentation
  - uncluttered, compact, yet easy read
  - reducing navigation and scrolling
- Smart link insertion
- Can extract doc with autogsdoc, Clang or others

DocGenerator Demo

Basic Markup

- @group to organize classes, categories etc.
- @task and @taskunit to regroup methods, functions, constants etc.
- @abstract, @section, @param, etc.

Additional Markup

- Subset of GSDoc markup
  - Code examples
  - Various kind of lists
  - etc.

Smart Link Insertion

- (-[MyClass bip] is dumb, but less than MyProtocol.)
- Will correctly detect both in the middle of punctuation
  - -[MyClass bip]
  - MyProtocol

Page Generation

- One class or protocol per page
- Categories consolidated on a common page per class
- Can be extended to support other strategies
  - e.g. API Overview presents all classes, protocols and categories on a single page

Page Model

- Basic Template support
  - Overview
  - Menu
- Page subclassing possible to customize
  - doc element arrangement
  - HTML output

Future Plans

- Property and ivar support
- Collapsable/expandable source code per method or function
- Public vs internal doc
- Smalltalk doc generation
- SourceCodeKit/Clang and EtoileText integration

EtoileFoundation
&
Metamodel

EtoileFoundation

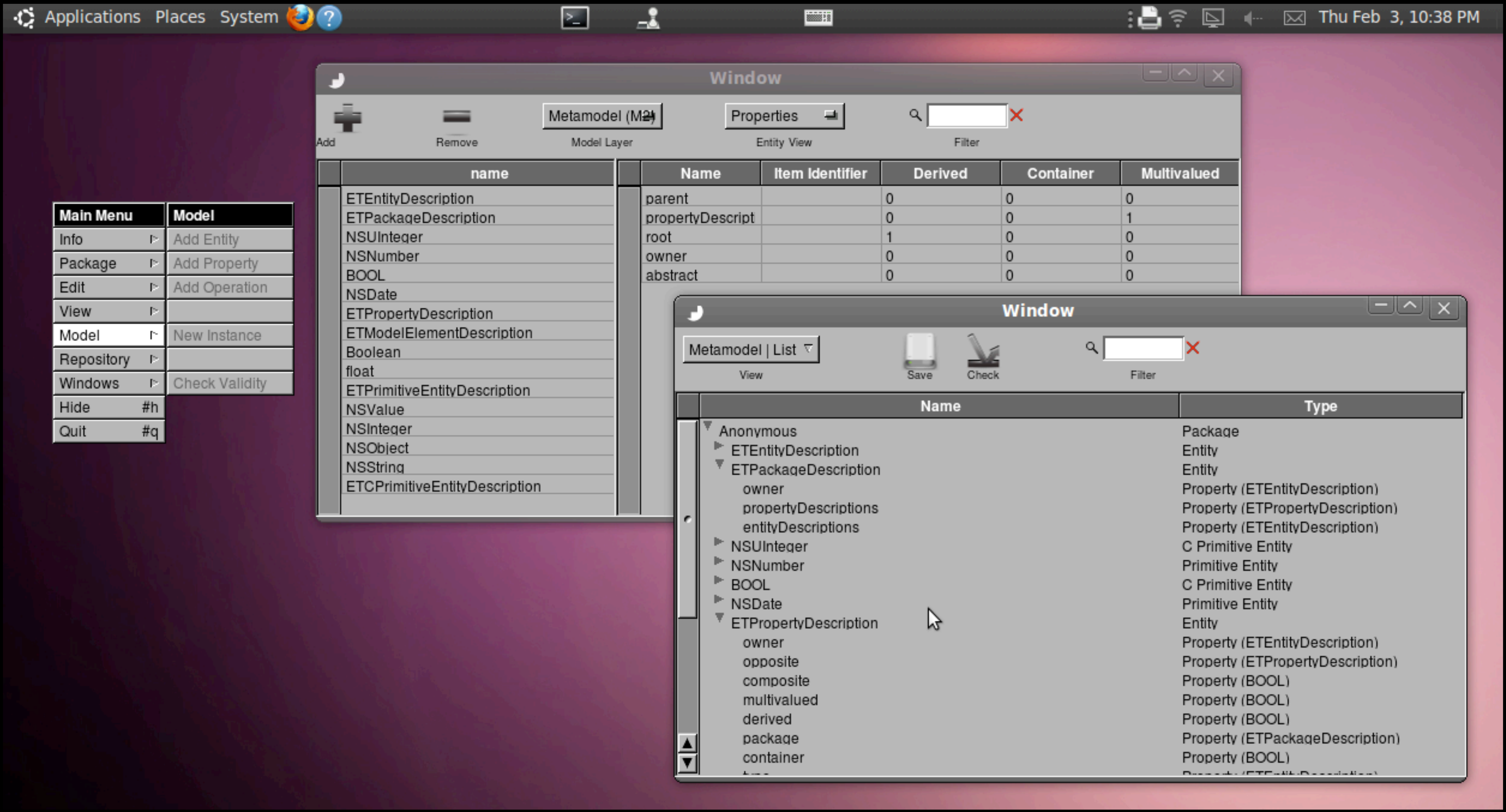
- Prototypes, mixins
- Collection Protocols
- High-order Messaging
- FAME-inspired Metamodel
- Mirror-based Reflection
- UTI, UUID, History, Socket etc. classes

Metamodel

- Inspired by
  - FAME Smalltalk and Java
  - EMOF
  - CoreData and EOF
- UI Generation with EtoileUI underway
- Will be used in CoreObject

Minimalistic

- Property, Entity and Package descriptions
- Property role stereotypes
  - number, relationship, option list etc.
- Model and Metamodel Repository
- Self-described
  - Metamodel can evolve at runtime



name
ETEntityDescription
ETPackageDescription
NSUInteger
NSNumber
BOOL
NSDate
ETPropertyDescription
ETModelElementDescription
Boolean
float
ETPrimitiveEntityDescription
NSValue
NSInteger
NSObject
NSString
ETCPrimitiveEntityDescription

Name	Item Identifier	Derived	Container	Multivalued
parent		0	0	0
propertyDescript		0	0	1
root		1	0	0
owner		0	0	0
abstract		0	0	0

Name	Type
▼ Anonymous	Package
▶ ETEntityDescription	Entity
▼ ETPackageDescription	Entity
owner	Property (ETEntityDescription)
propertyDescriptions	Property (ETPropertyDescription)
entityDescriptions	Property (ETEntityDescription)
▶ NSUInteger	C Primitive Entity
▶ NSNumber	Primitive Entity
▶ BOOL	C Primitive Entity
▶ NSDate	Primitive Entity
▼ ETPropertyDescription	Entity
owner	Property (ETEntityDescription)
opposite	Property (ETPropertyDescription)
composite	Property (BOOL)
multivalued	Property (BOOL)
derived	Property (BOOL)
package	Property (ETPackageDescription)
container	Property (BOOL)

Model Builder

Editing a package & browsing a repository

ObjectMerging

ObjectMerging

- Family of CoreObject prototypes
- Same core ideas and API than CoreObject
- Will probably replace current CoreObject

Goals

- Selective Undo Support
- Simpler Implementation
- More robust
  - Catch more serialization logic issues
  - Object graph integrity checks
  - Prevent deterministic replay mistakes

Persistency

- New persistency approach based on
  - metamodel-driven serialization
  - object graph diffing
  - rather than message recording

Persistency

- At commit time
  - Object graph diff computed by reducing changes to primitive operations
- History is a commit sequence

Primitive Operations

- Property update
- Set add and remove
- Sequence insert and remove

New Merging Model

- Object Graph Diff and Patch
- We don’t use
  - Operational Transformations (OT)
  - Address Space Transformation (AST)
  - or some hybrid models (such as OT with tombstones)

Existing Merging Models

- OT are a proved model, but
    - slow (commutation)
    - complex (transpose)
- AST is simpler, but
    - requires the entire history in memory :-/

Undo in less than minute

- Nobody has ever built a revision control system based on OT or AST
- Merging 10000 operations can take minutes to hours with OT

Where is the challenge?

- Core Object is not just a collaboration system like Gobi, Google Wave etc.
- It’s a revision control system
- A core object history could be as huge as
  - 500 000 commits
- Yet selective undo must be immediate

EtoileUI

EtoileUI

- Very close to be released :-)
- Since last February
  - Bug fixes, code cleaning and API tweaks
  - … and some new features
- More in the EtoileUI talk later

Étoilé Developers

Eric Wasylishen ⁂ Opal, Theming, ObjectMerging

Christopher Armstrong ⁂ ProjectManager

Niels Grewe ⁂ DBusKit, EtoileSerialize

David Chisnall ⁂ libobjc2, LanguageKit, EtoileText

Quentin Mathé ⁂ EtoileUI, DocGenerator, Metamodel

http://www.etoileos.com