Cross desktop Application Scripting
Ideas for an implementation

Desktop Developer Conference 2006,
Ottawa, Canada

Tuesday July 18th 2006

Hubert Figuière <hub@figuiere.net>

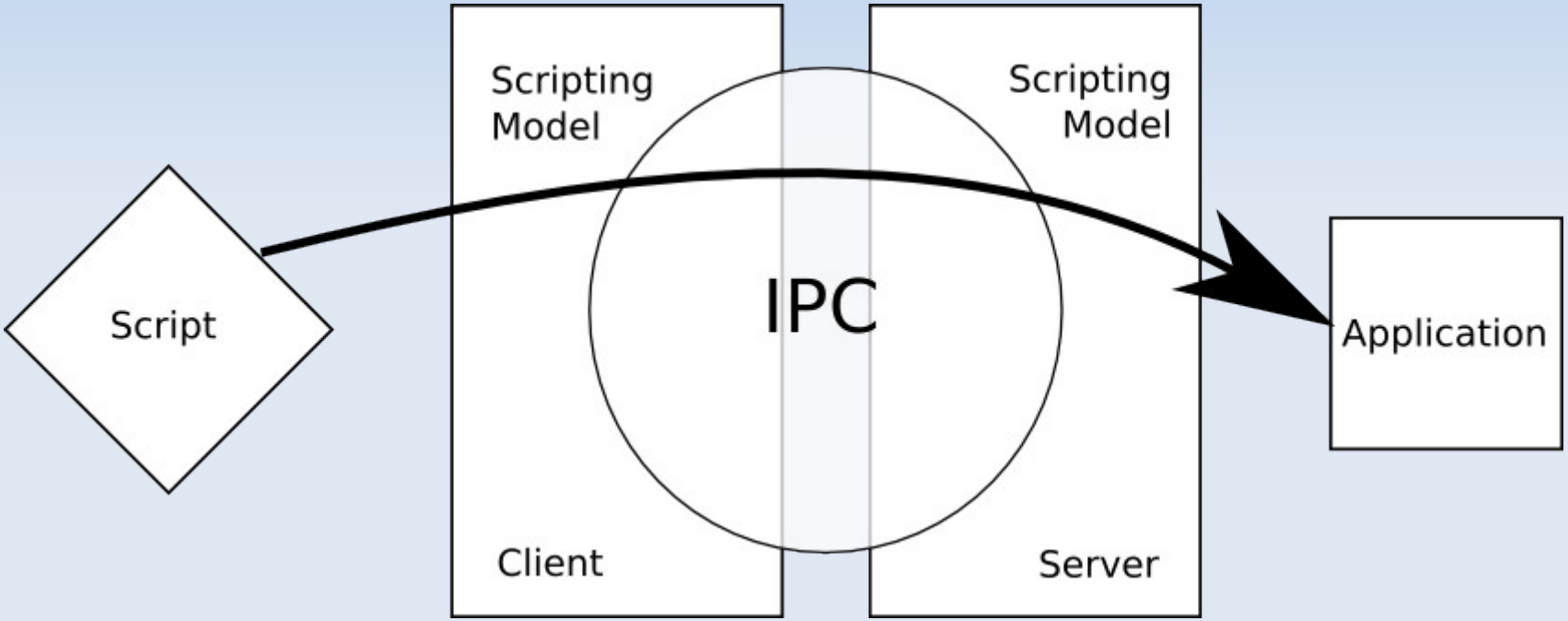
Application Scripting

- High level application control.
  - It is telling an application to do some task the application is designed for.
  - Fetch data from the runing application
- It is not meant for writing complete apps like Python or Perl are used.
- Think of a high-level IPC

Scripting System

- A language (any)
- A high-level “protocol”: *Scripting Model*
  - Preferably with an Object Oriented paradigm
- An IPC mechanism
- Applications

Scripting System (2)



What already exists?

- Kross http://kross.dipe.org/readme.html
  - KOffice scripting in Python and Ruby
- Kommander http://kommander.kdewebdev.org/
  - Use UI build with QtDesigner and DCOP
- Gnome + ATK + ORBit + Python http://www.jamesh.id.au/talks/lca2004/
  - control UI through the ATK

What already exists (2)

- Dogtail
  http://people.redhat.com/zcerza/dogtail/
  - UI testing tool. Use ATK and other a11y technologies
- StepTalk
  http://www.gnustep.org/experience/StepTa
  - Scripting for GNUStep
  - Use Objective-C introspection
  - Language independente

What already exists (3)

- All these applications with scripting capabilities
  - Gimp
  - Mozilla
  - etc.

What can use it?

- Atomato
  - Automation tool for GNOME
  - Typically benefiting from a universal way to remote control applications
  - Inspired from Apple Automator: http://www.apple.com/macosx/features/autom
- All these applications with scripting capabilities

What for?

- Automating tasks
- Fetching data from an application
  - Current song being played in Music Player
  - Exporting data
- Integrating applications together

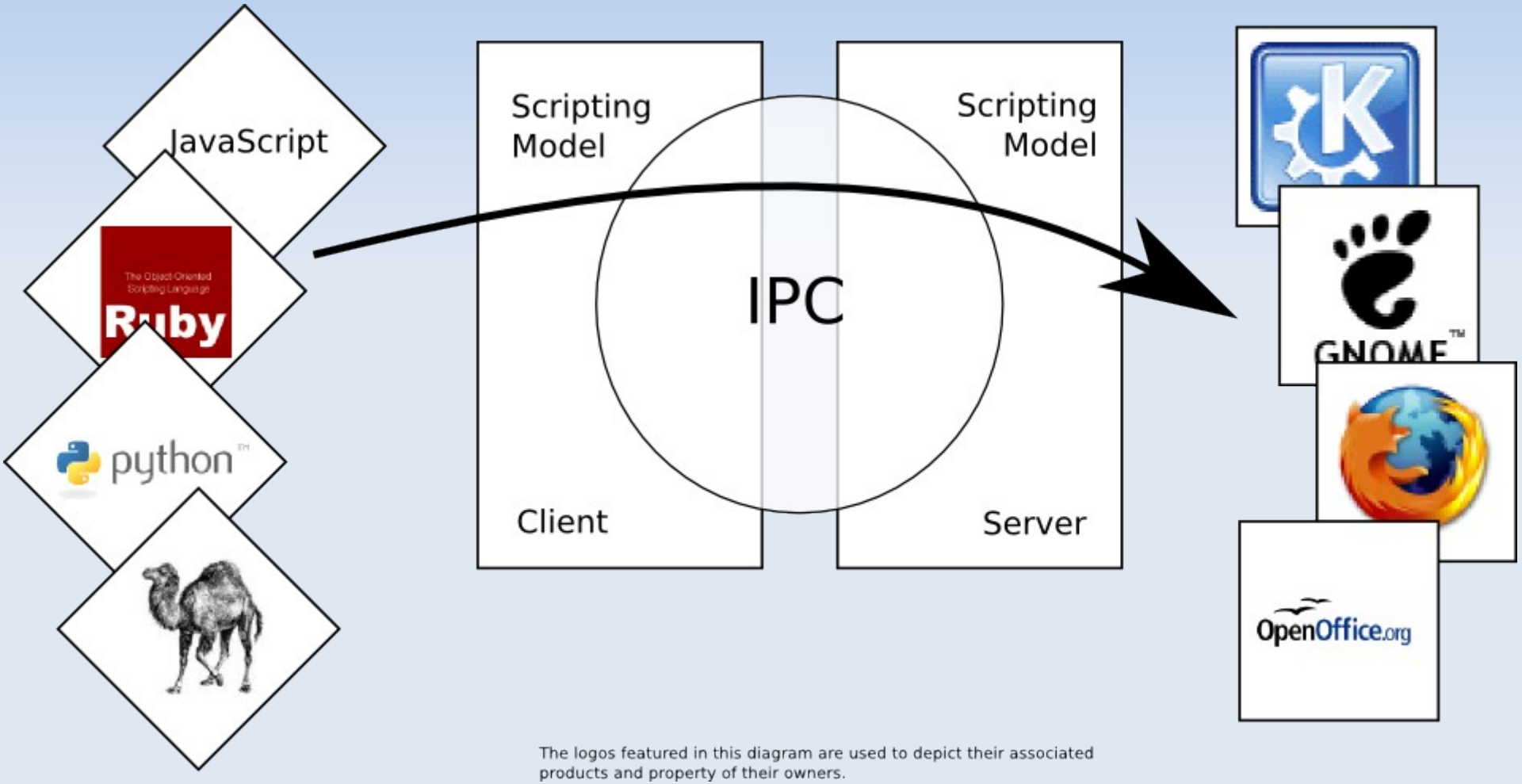
Inspiration

- AppleScript
  - Provides all the feature wished
  - But specific to MacOS X

Goals

- Maximum interoperability
  - KDE
  - Gnome
  - Mozilla
  - OOo
- Standardization of commands
- Language independence
- Security

Goals (2)



The logos featured in this diagram are used to depict their associated products and property of their owners.

IPC

- D-Bus
  - Adopted by Gnome
  - Adopted by KDE for KDE4
  - Structured IPC with access control
- DCOP
  - In KDE3. Will be replaced by D-Bus in KDE4
- Other?

Language

- Pick the language of your choice
  - Write an adapter to the S*cripting Model*
- Language by default (suggestion)
  - JavaScript
    - Because lot of end use know it (web designer, etc)
    - Easy to learn
    - Light implementation
  - Nobody want to script in Scheme or Lisp (or at least nobody sane)

Applications

- Server: the application controled
  - Glue to the scripting model
- Client: controling another application
  - Call the scripting model APIs

Scripting Model

- An object oriented model
  - You fetch objects
    - word
    - canvas
    - email
  - You manipulates objects
    - delete word
    - create email
  - etc.
  - More natural

Scripting Model (2)

- Translate the applications data model (functionnality)
  - `delete word 1 of paragraph 3 of document of front window`
  - `draw rectangle with (5, 10, 100, 100) in canvas of front window`
- Does not translate the user interface
  - at least not as a first choice

Scripting Model (3)

- Standardized
  - Having to rewrite the script because we changed application would suck
  - Manipulating data should be done the same way in all “similar” applications
    - Word processors
    - Drawing programs
    - Web browsers
    - e-mail programs
    - etc.

Standardization

- Task done the same way whatever the application is
  - Writing a text is writing a text be it in KWord or AbiWord
  - Creating a table in a spreadsheet is the same be it Gnumeric or KSpread
  - etc.
- Described in a “registry”

“Registry”

- Contain and complete description of the model
- Maintained as an open specification
- Updated periodically for additions
- The goal: interoperability
- The scripting “bible”

Knowing what an app. supports?

- How do we know what an application support
  - Querying its interface
- What does the application provide?
  - A list of object and commands it understands

Challenges

- In order to be able to have the scripting model adopted:
  - Write it without Qt (licensing incompatible with Gnome policy)
  - Provide plain C API (Gnome and its gazillion of language bindings) and C++ (KDE)
- Make it easy to use on the server side
  - providing scripting should be easy or nobody will do it
- Secure

Security

- Very hard to achieve
  - Things shouldn't happen without user approval
    - Sending mail
      - The “I Love You” fiasco in 2000 and later on Windows
    - Worm and virus propagation
      - Why antivirus and anti-spyware editors do love Microsoft
  - But things should work
    - I have no answer right now

Questions