Binding applications with PLASTIC
John Taylor
Institute for Astronomy, Edinburgh / AstroGrid
PLASTIC is...

- A specification for how astronomy tools on a user's desktop can talk to each other
- Simple
- Extensible
- A collaboration between the Aladin, AstroGrid, Topcat, VisIVO teams and others
Contents

- PLASTIC – why and how?
- Demos
- Where are we now?
- What next?
Why?

- We needed a quick and easy way for apps to communicate
  - Don't need guaranteed delivery, encryption, security, other bells and whistles
- Most of the interesting stuff is working out the ways that apps should communicate once they can.
Some history...

- VisIVO & Aladin interoperability (Becciani, Comparato, Gheller...)
- The Aladin plugin interface (Boch, Fernique...)
- The Astro Client Runtime (Winstanley)
- SC4DEVO meetings (Mann)
Architecture: publish-subscribe

"Plastic Hub"
Look Familiar?
Exactly the same options that you get in the AR - if you use Plastic you get all the facilities of the AR “for free” (and vice versa)
What do messages look like?

Why are these IVORMS?

In the future we might want to register them to
• make it easier to search for applications that have certain capabilities
• Guarantee uniqueness
• Facilitate auto-generation of UIs for sending messages?
What do messages look like?

- Messages can carry arguments, and return values
Demos

- Plastic for beginners - JDT
- Integrating Topcat and Aladin – Thomas Boch
- Various other presentations will include asides on PLASTIC
Where are we now?

- An implementation of a Plastic Hub in the AR.
- Plastic-compatible tools:
  - Aladin
  - Topcat
  - VOSpec
  - VisIVO (C++)
  - AstroScope
  - Xmdv-lite
  - TabView
  - VAST
- Coming soon:
  - Anomaly Detector, Eirik?
  - VOEvent (Perl) ?
  - VOQuest
Where are we now II?

- Bindings available for R, Python
- Coming soon: IDL?
- Java libraries for accessing plastic hub
- Website: http://plastic.sourceforge.net
- Mailing lists: http://plastic.sourceforge.net/mail-lists.html
Hidden slides with code examples to follow....
Plasticizing your apps: Java
(non-programmers, please talk amongst yourselves)
Also see:
http://plastic.sourceforge.net/multiproject/plastic-lib

- Implement the PlasticListener interface:
  Object perform(URI sender, URI message, Object[] args)
- Obtain a reference to the PlasticHub
- Register your application
  id = hub.registerRMI("myapp", messages, this)
- Send messages to other apps:
  Map responses = hub.request(id, message, args)
**Plasticizing your apps: xml-rpc (Python)**

*(non-programmers, please talk amongst yourselves)*

- Run an xml-rpc server in your app (with an off-the-shelf-library)
- Obtain the URL of the hub's xml-rpc server
- Register your application
  ```python
  id = 
  s.plastic.hub.registerXMLRPC("myapp", [], myURL)
  ```
- Send messages to other apps:
  ```python
  responses = 
  s.plastic.hub.request(id, message, args)
  ```
Plasticizing your apps: scripting

You can also use Plastic to visualise data directly from your favourite scripting environment (Python in this case)

- Obtain the URL of the hub's xml-rpc server
- Register your application
  ```python
id =
  s.plastic.hub.registerNoCallback("myapp")
```
- Send messages to other apps:
  ```python
responses =
  s.plastic.hub.request(id, message, args)
```
PLASTIC
Platform for Astronomy Tool InterCommunication

- Moulded by
  Thomas Boch
  Marco Comparato
  John Taylor
  Mark Taylor
  Noel Winstanley

- Other contributors:
  Alasdair Allan
  Sebastien Derriere
  Pierre Fernique
  Doug Tody
Anomaly Detection

- Using:
  - CEA
  - ACR
  - Plastic
  - Aladin/Topcat/....everything, really

- Credits:
  - Andy Connolly (Pitt)
  - Bob Mann (Edi)
  - John Taylor (Edi)
  - The AstroGrid & Plastic teams
Anomaly Detection – what & why

- Two prototype applications showcasing what you can do with the ACR and Plastic.
- Quick to write – all the real work is done by ACR, CEA and any Plastic applications.
- Prototype/demo software, driving development of infrastructure but still doing something “useful”.
- Datamining half of DS6
- Collaboration outside VOTech
Anomaly Detection Application

Integrate into the AstroGrid infrastructure in 3 ways:
- Server application (CEA)
- Desktop application (ACR)
- Hybrid application:
  - All the grunt is done on the server
  - Presents the user with a more friendly, powerful UI than is available through the vanilla ACR task launcher

The same AnomalyDetection algorithm is available as:
- A CEA app
- A parameterised workflow
- A hybrid ACR-CEA app

Ease of use
Anomaly Detection Application

- Take a source VOTable (e.g. From MySpace)
- Run a remote task that extracts the column headings
- Get the user to select the column headings of interest, and reduce the table to these columns
- Create a workflow document that sends this table to the Anomaly Detector CEA application (plus does some table format stuff)
- Submit the workflow to JES
- Visualise the results via Plastic