Towards an Astro Apps Store

- The App Stores and App Economy Era
- Advantages of a common application framework
- The basic ingredients
- Feasibility: the FASE prototype as working example
- In practice
- The paradigm
- Issues

The Apps Stores and Apps Economy era

amazon

android

Samsung Apps

NOKIA ovi

LG world

Package Managers

FINK

MacPorts

rpm

DEB

synaptic

aptitude

apt-get

Ingredients

- A well defined packaging system
  - Describe the software and how to manage it
  - Place the code in a specific structure
  - Implement specific interfaces for the plug-in facility
- A distributed software repository (registry)
  - Software easily retrievable
  - Search engine: look for the software you need
  - Easy install
- An abstract API to access and execute the software
  - Several profiles
- Libraries

Advantages of an apps framework for the astronomy

- Simplify the astronomer life
  - Astronomical applications discovery
  - Applications installation and upgrade
- Easy development and distribution
  - Libraries
  - Packaging
- Sharing of interoperable code
- Enhanced capabilities (VO, GRID, HPC)

FASE prototype

- Is it all really feasible? YES IT IS!
- Future Astronomical Software Environment
  - OPTICON FP6/7 + USVAO + INAF + OAMP LAM + ESO + ESA + …
    - https://www.eso.org/wiki/bin/view/Opticon/WebHome
  - Documents:
    - High level requirements doc (P. Grosbøl et al.)
    - Architectural document draft (D. Tody et al.)
    - White paper on the architecture (D. Tody et al.)
  - Python/ANSI-C prototype (L. Paioro et al.)
    - Non public software using FASE
      - LBT LUCIFER pipeline
      - EUCLID NISP simulation software
      - VIPERS survey VLT VIMOS data reduction system (old prototype)
    - Public demos
      - TOPCAT, DS9 (SAMP)
      - SExtractor (process spawning)
      - ESO-CPL example recipes (direct C bindings)

The Astro Apps Store

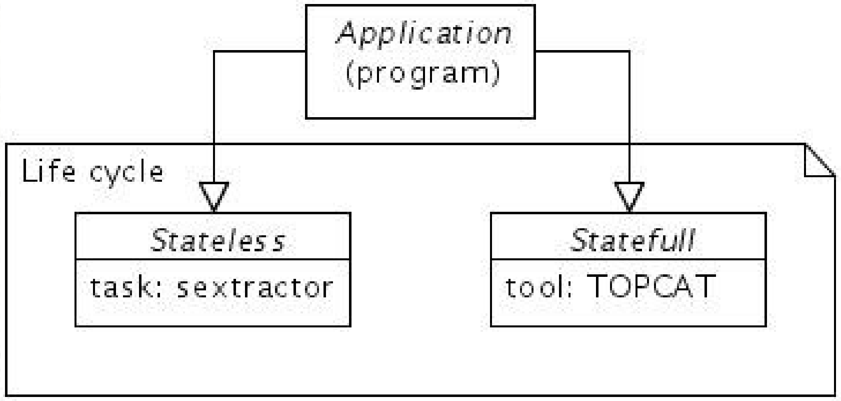
- Astro Apps Store:
  - http://antigone.lambrate.inaf.it
  - Based on Python Package Index technology
  - The core system:
    - http://antigone.lambrate.inaf.it/fase

In practice

- Packages/Apps description
  - XML description file (AppRegExt?)
  - http://antigone.lambrate.inaf.it/media/xml/package.xsd
- Package Management
  - Register/Download/Upload protocol
  - Packaging
    - zip, tar.gz, etc.
    - including XML description file (AppRegExt?)
  - Install mechanism
- API coupled with execution engine
  - SAMP + Discovery and activation service

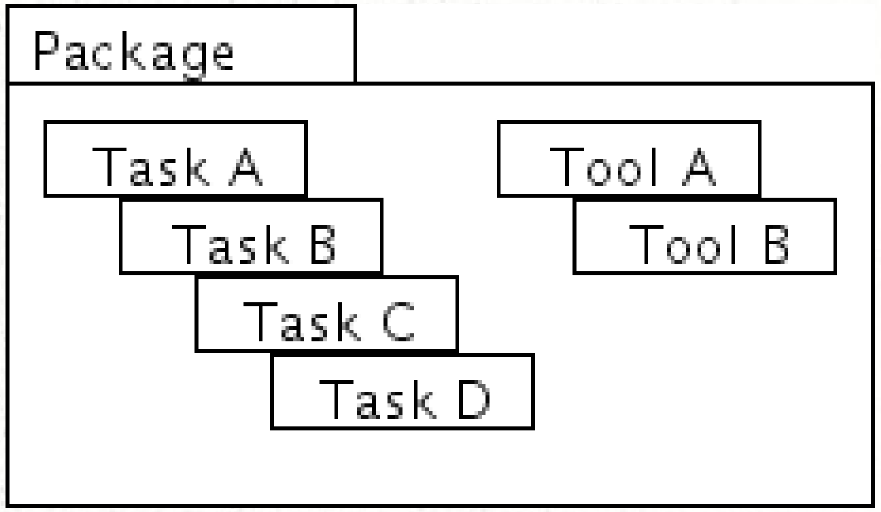
The paradigm

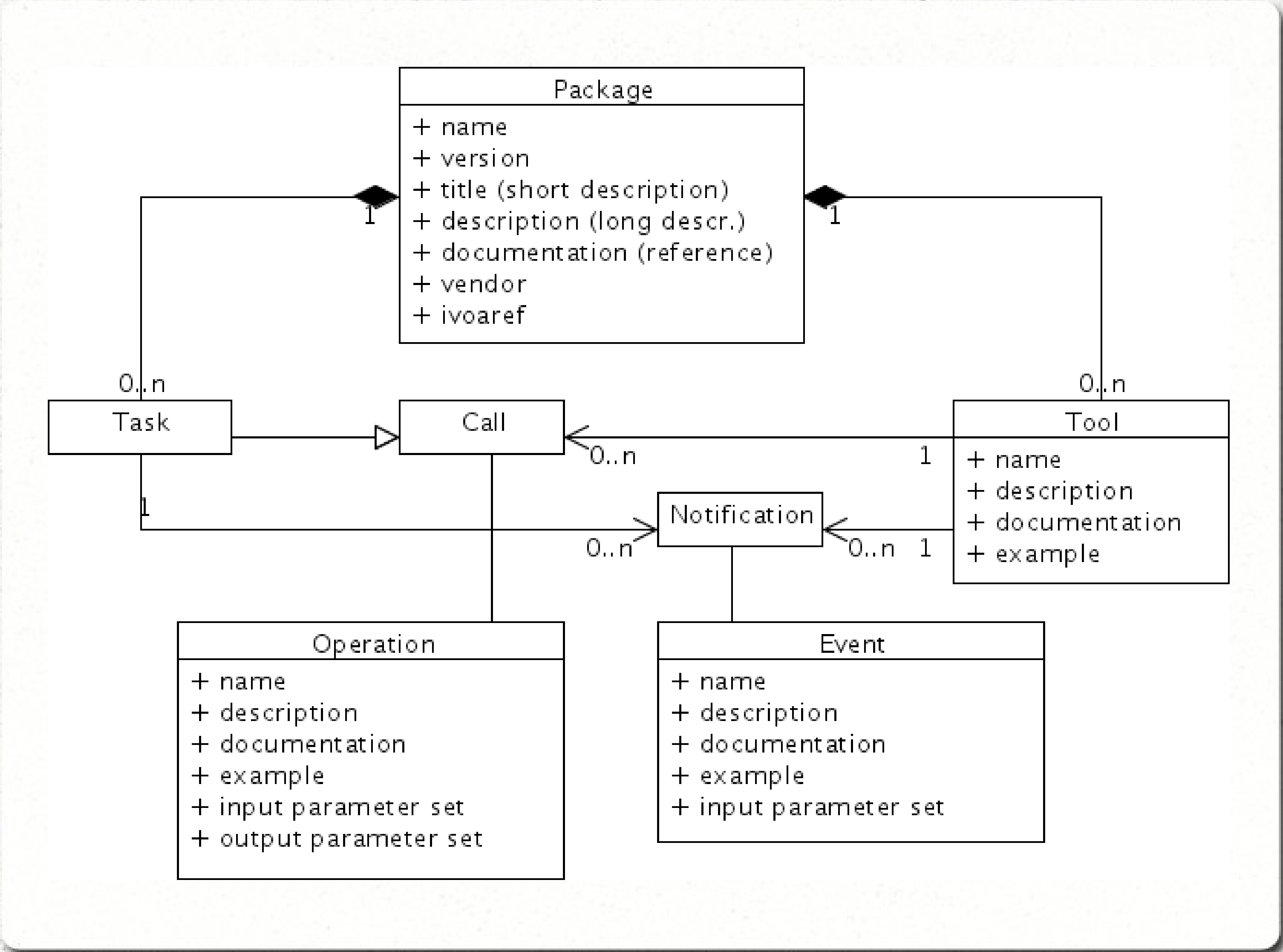
- Task-like:
  - Stateless (starts and ends)
  - Performs 1 operation
  - Gets an input parameter set and returns an output parameter set
- Tool-like
  - Statefull (persistent)
  - Can perform several operations
  - Each operation has an input parameter set and returns an output parameter set



The paradigm

- Programs collected into packages
  - One or more tasks
  - One or more tools




Package
+ name
+ version
+ title (short description)
+ description (long descr.)
+ documentation (reference)
+ vendor
+ ivoaref
1
1
0..n
0..n
Task
Call
Tool
+ name
+ description
+ documentation
+ example
0..n
1
1
Notification
0..n
0..n
1
Operation
+ name
+ description
+ documentation
+ example
+ input parameter set
+ output parameter set
Event
+ name
+ description
+ documentation
+ example
+ input parameter set

Issues for the AppRegExt

- Description of packages and applications
  - Name, version, license, etc.
  - Classifiers, keywords
  - Documentation: do we need to specify a standard mark-up language?
- Distinction between stateful and stateless apps
- MTypes
  - Can a task be mapped to an MType?
  - Can a tool be mapped to a SAMP client?
  - What about the events notified? And the calls performed?
  - MTypes description in XML format (machine readable)
- Dependencies