Run Through The Jungle

Mathias Bauer
StarOffice/OpenOffice.org
Sun Microsystems Inc.

Agenda

- Some words about the speaker
- Introduction
- Rumours and facts about modularity
- OOo code architecture
- Framework Environment
- Applications in the framework
- Integration of features into the UI
- OOo Writer architecture
- Q & A

Notes about the speaker

- At Sun Microsystems:
  - > Working on StarOffice/OpenOffice.org since 1995
  - > Application Framework, 3rd party integration
  - > Manager Software Engineering
    - – Application Framework Team
    - – Writer, Math, Text Engine
- For OpenOffice.org:
  - > Working on the project since its foundation
  - > Application Framework Project Lead
  - > In the past also worked on “Programmability”

Introduction

- Interesting places (work in progress)
  - http://wiki.services.openoffice.org/wiki/Architecture
  - http://wiki.services.openoffice.org/wiki/Framework
  - http://framework.openoffice.org/servlets/ProjectDocumentList
- What you should know already
  - Some basic ideas what OOo can do
  - Some basic knowledge about which parts OOo has
  - Basic UNO concepts (interfaces, services)

Possible meanings of “modularity”

- Split up code into libraries, perhaps load on demand
  - > Loading on demand can hurt code quality
  - > Too many libraries can have negative impact on disk space, memory footprint and performance
- Make many parts of the program removable
  - > Questionable if applicable to parts of applications
  - > Removable parts must be loadable on demand
- Reduce build dependencies and narrow interfaces
  - > Can be quite some work if done after the fact
  - > Improves code maintainability and stability
  - > Bad or unclear modularity makes code look like a jungle

Concrete example: OOo on Windows

- http://wiki.services.openoffice.org/wiki/Architecture/Libraries
- Full installation contains 309 libs, 100 MB
- Startup without application uses 68 libs, 28 MB
- Adding writer loads 21 libs more, 17 MB
- Many libraries are loaded on demand (most of them being UNO services)
- Some big libraries are loaded though the loading code uses only a few symbols
- Biggest problem is huge svx library

First Impression of an OOo developer noob

“Better run through the jungle, Woa, dont look back to see.”

John Fogerty
Creedence Clearwater Revival
“Run Through The Jungle”, 1970

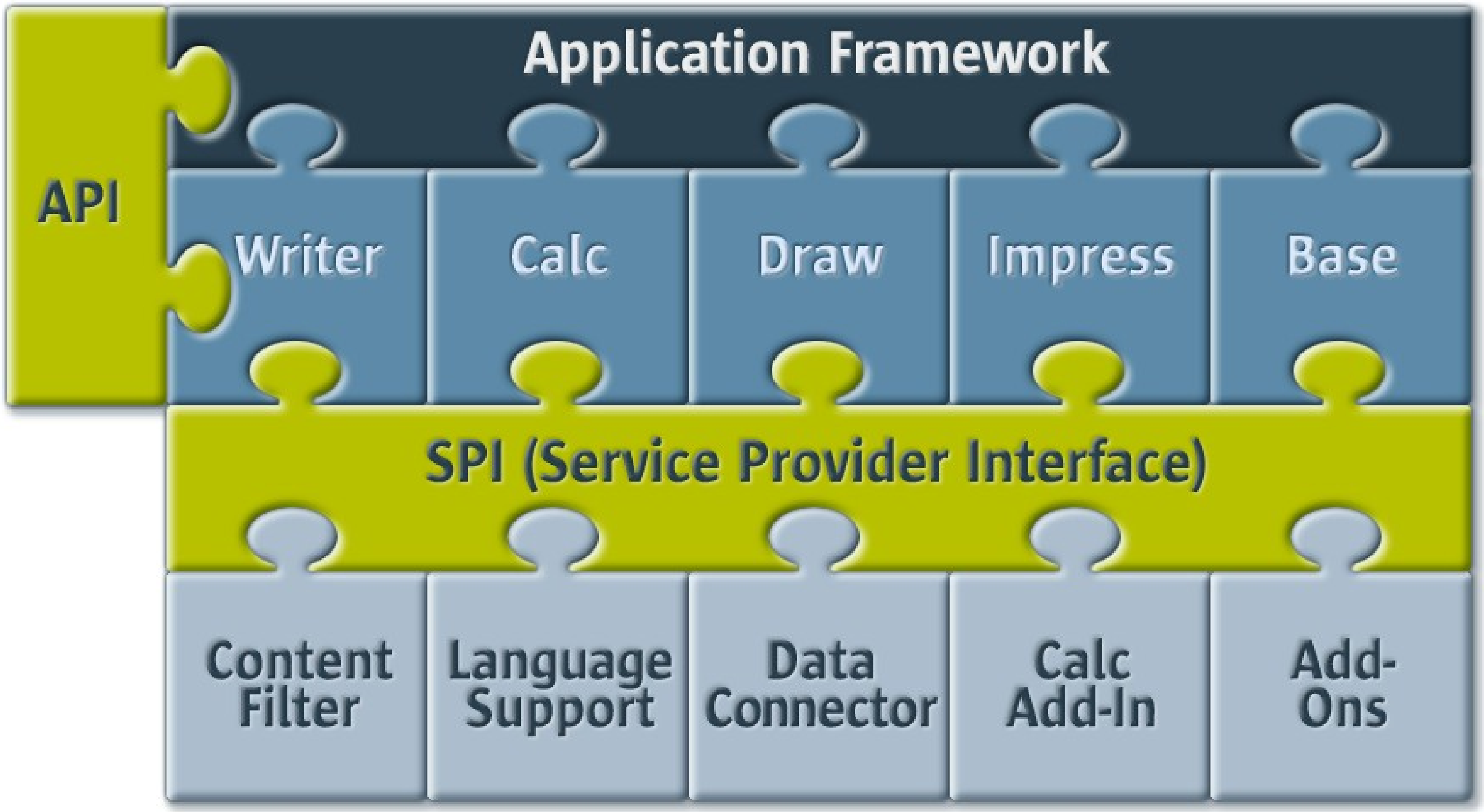
Elements of the OOo architecture

- GUI class library (StarView, VCL)
- Application Framework
- Support libraries
  - Adaptors, helpers
  - C++ service class libraries
  - GUI classes (dialogs etc.)
- Libraries containing UNO services
- UNO Language binding and runtime
- System integration
- Application libraries

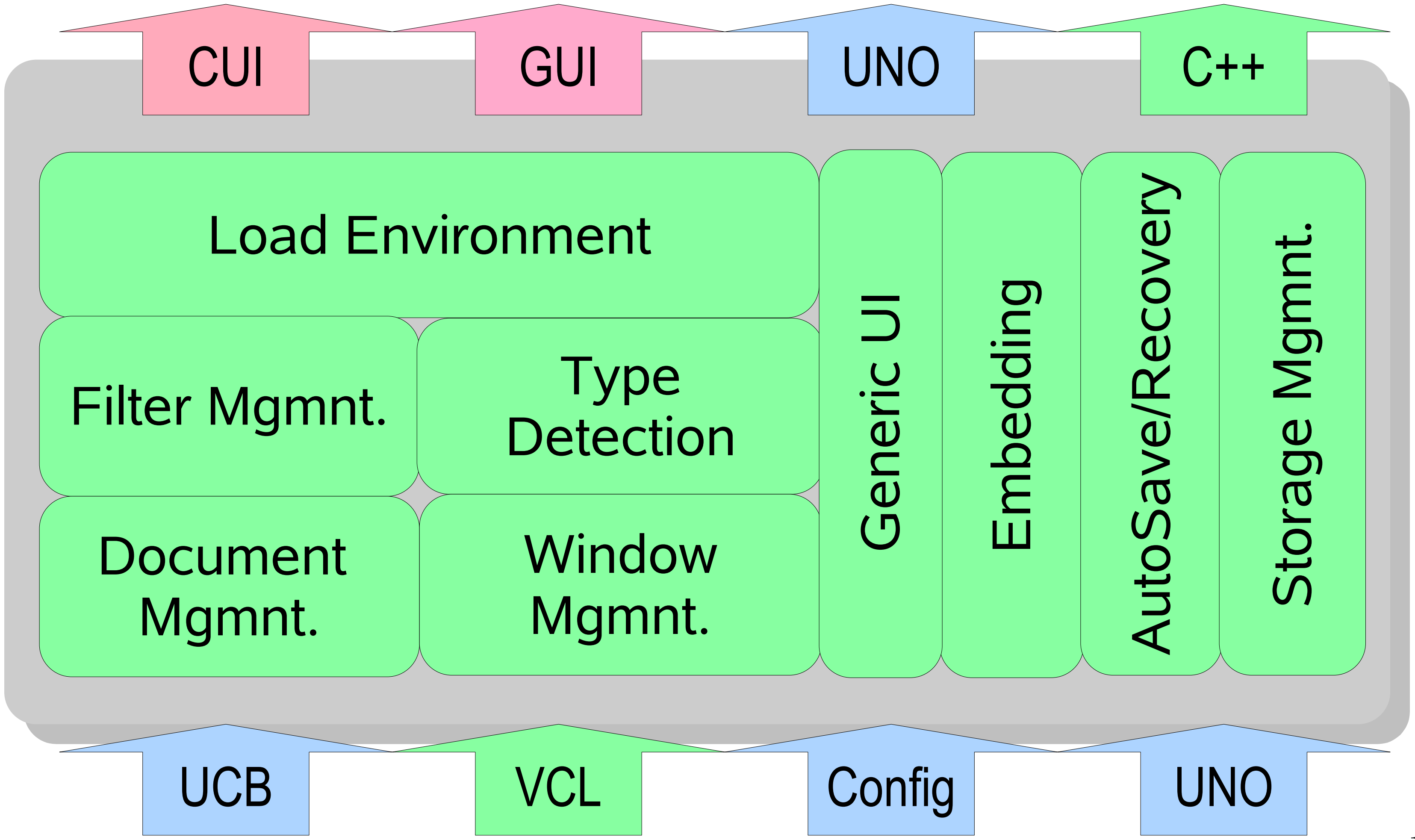
“Template” based framework

- Characteristics
  - Control over all aspects of the program
  - Base classes for Application, Document, View etc.
  - Derived classes only add specific aspects
- Advantages
  - High code reuse
  - Fast addition of new features if they fit into the picture
- Possible drawbacks
  - Tight coupling of modules and classes
  - Large build dependencies
  - “High level hacking”

Coarse-grained architecture of OOo



Framework Architecture

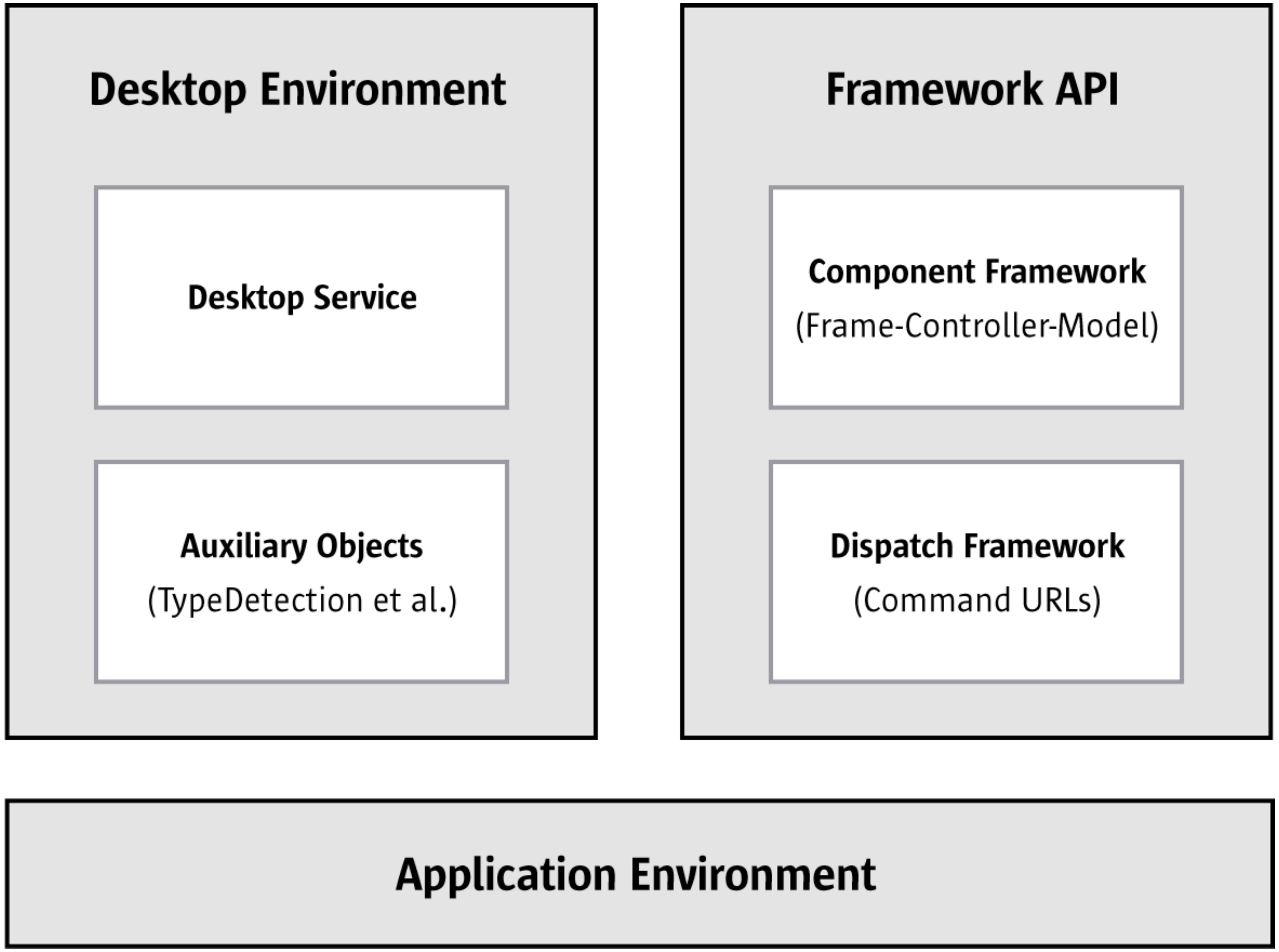


General purpose services

(Module names in paranthesis)

- Embedding objects (embeddedobj)
- Embed OOo through OLE2 (embedserv)
- Storage and package management (package, sot)
- Configuration (configmgr)
- Document filter management (filter)
- Templates management (sfx2)
- Dispatch Provider for global functionality (sfx2)

Application Environment

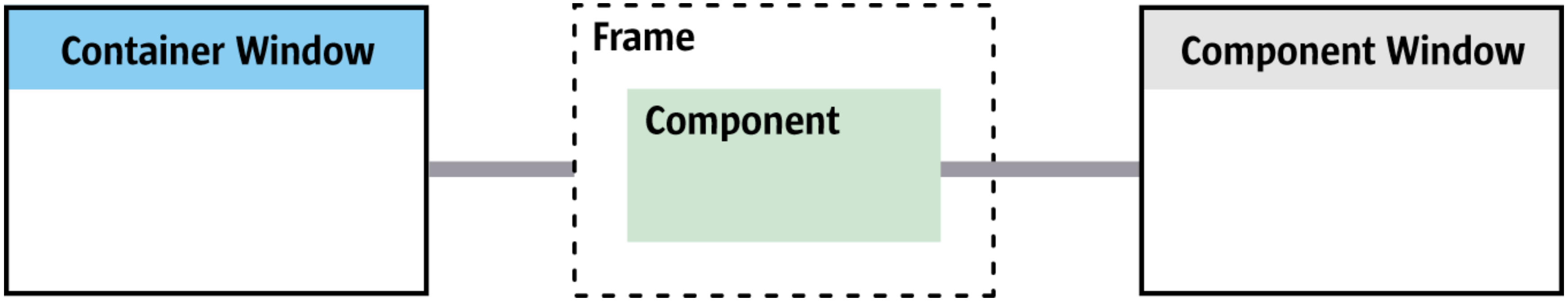


Starting OOo: the desktop module

- UNO bootstrapping (creation of service manager)
- Creation and initialisation of some general services
  - Configuration
  - UCB
  - Desktop
  - GlobalEventBroadcaster
- Instantiate handler for command line / pipe
- React according to command line arguments
- Exact description at http://wiki.services.openoffice.org/wiki/Architecture/Process_Flow

Framework API

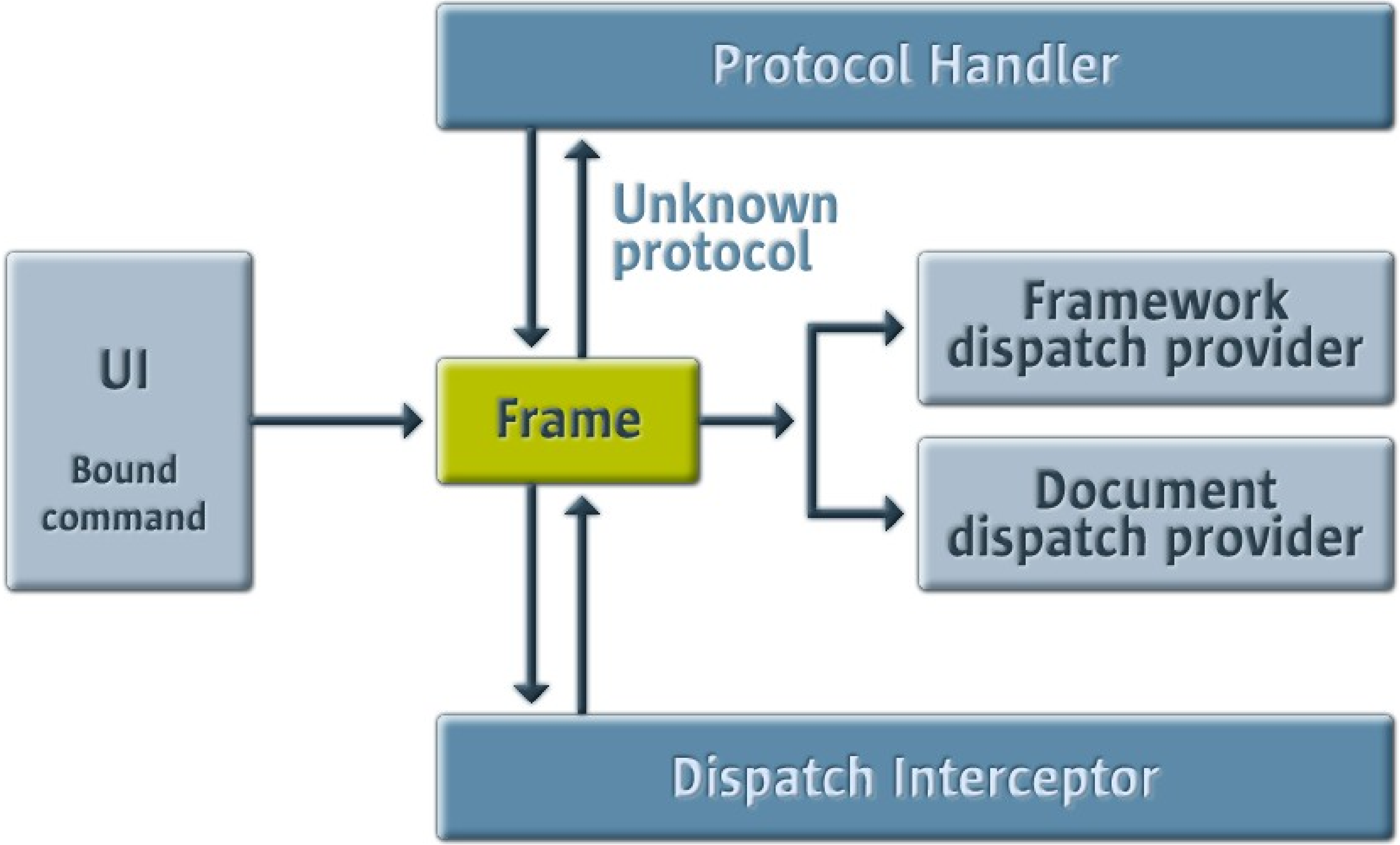
- Frame
  - > Owns component
  - > Controls task window
  - > Anchor for dispatching
- Component
  - > Controller, model is optional (MVC paradigm)
  - > Has component window
  - > Participates in dispatching



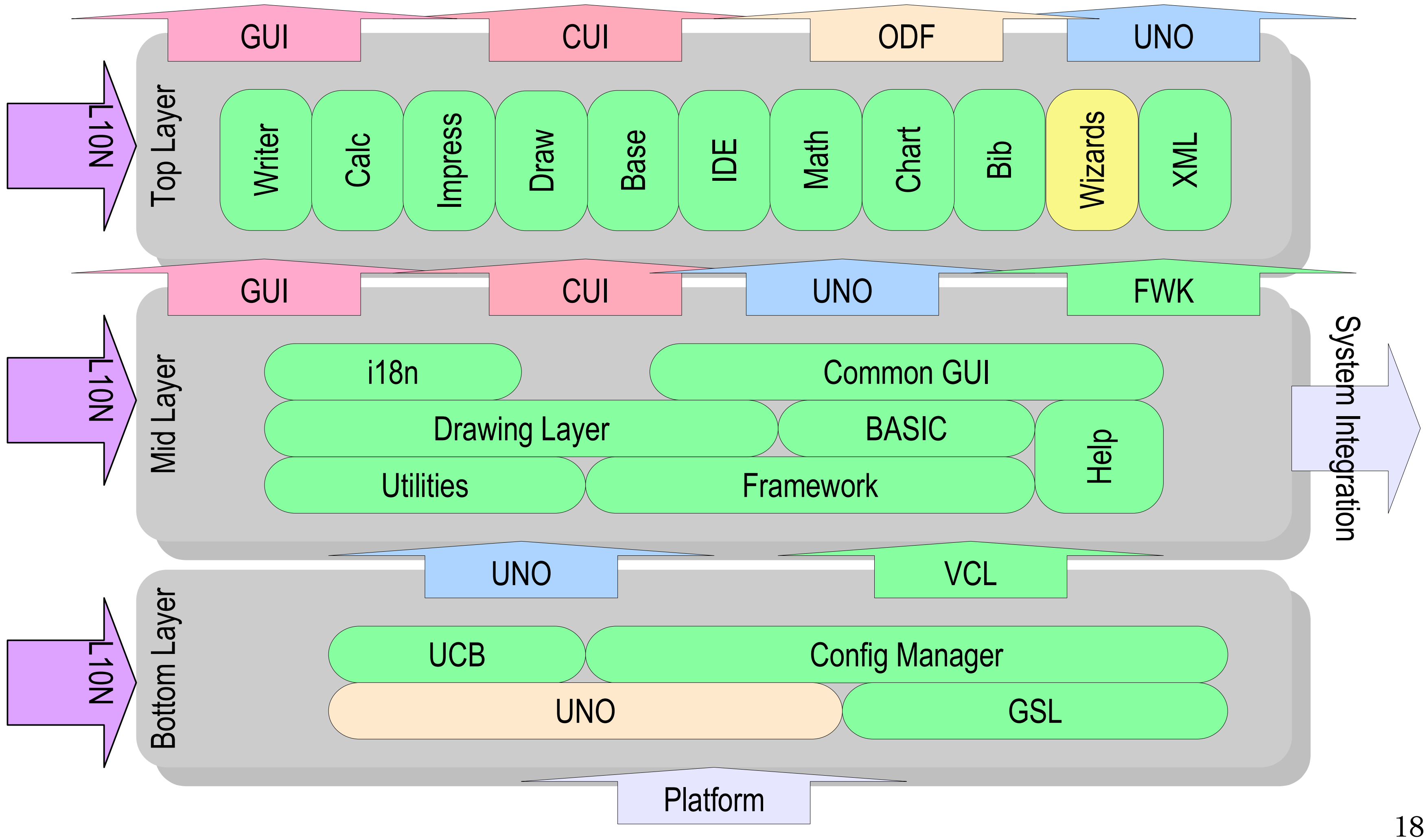
Creating/loading a document

- Framework creates Frame and its window
- Framework loads document into Frame
  - > Does type and filter detection
  - > Hands over Frame to FrameLoader
  - > FrameLoader creates document service (model) according to filter
  - > Hands over medium to document for loading
  - > Creates a view/controller pair for the model
  - > Plugs Controller/Model/Frame together
  - > Controller requests UI elements, provides configuration
  - > Frame creates UI elements according to configuration

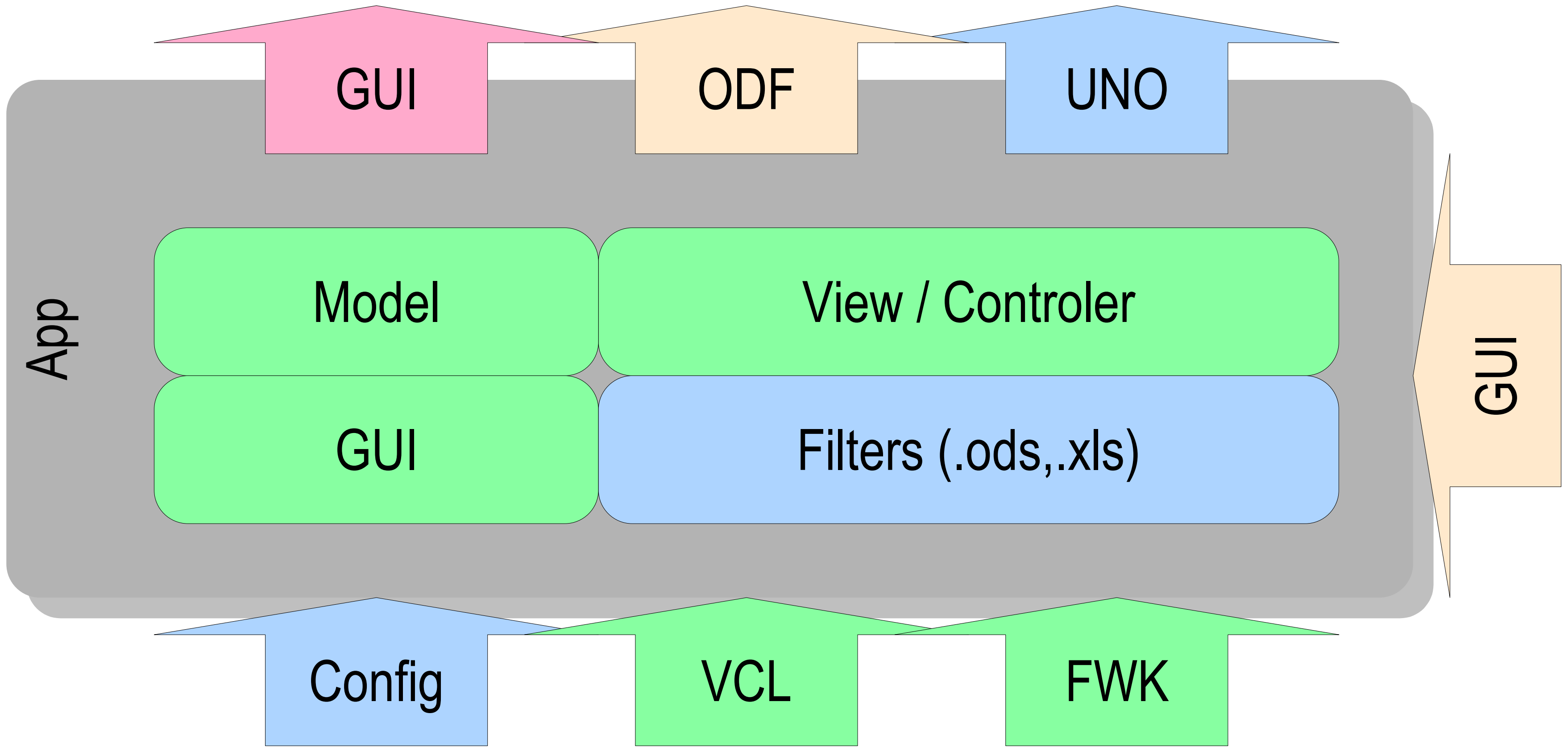
Command Dispatching



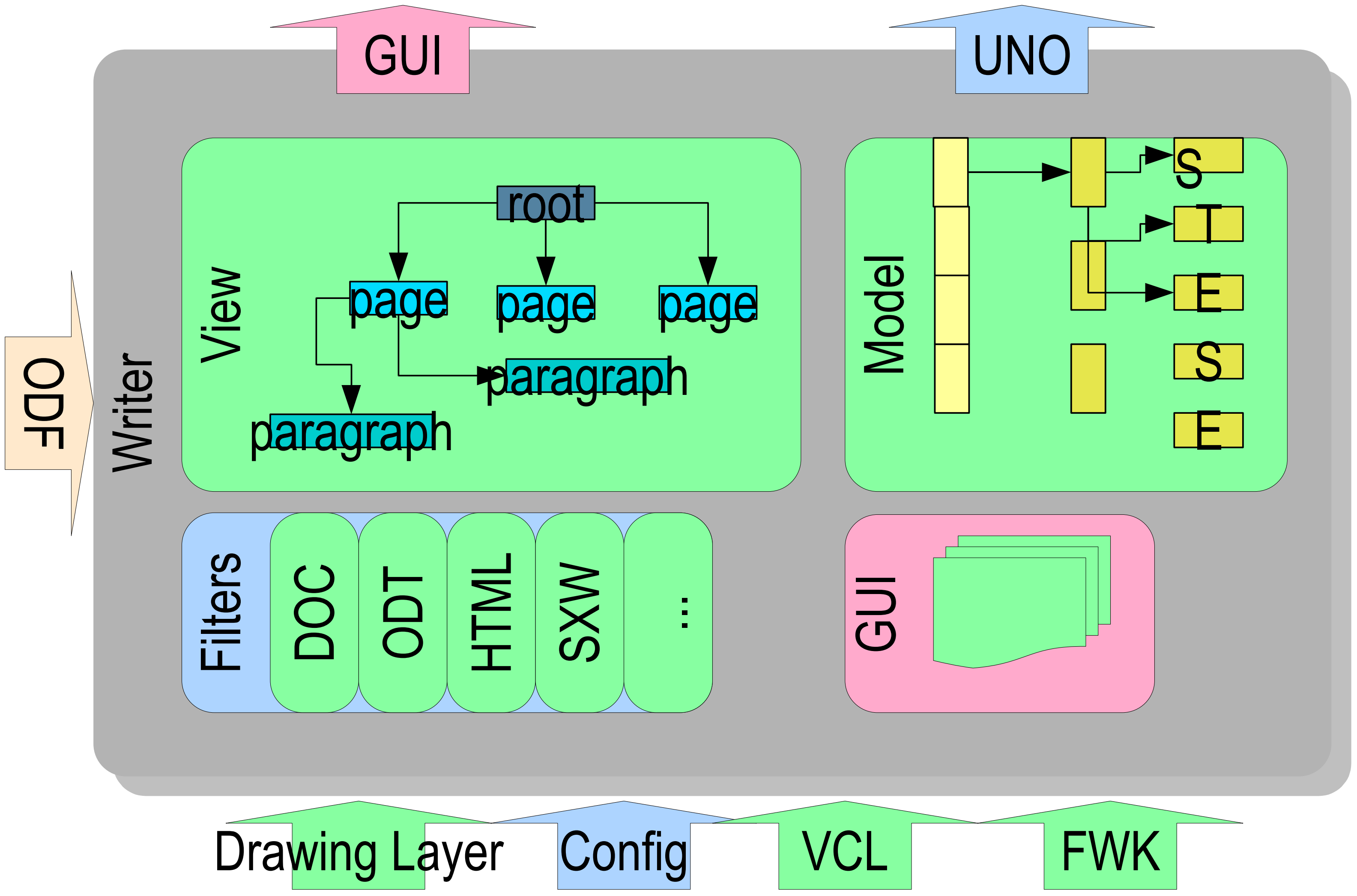
Overall architecture



Basic Application Architecture



Writer Architecture



Run Through The Jungle - finished

Mathias Bauer
Mathias.Bauer@sun.com