# LibreOffice: Code Structure

By Miklos Vajna

**Senior Software Engineer at Collabora Productivity**

2017-10-11

## About Miklos

- From Hungary
  - More blurb: http://vmiklos.hu/
- Google Summer of Code 2010/2011
  - Rewrite of the Writer RTF import/export
- Writer developer since 2012
- Contractor at Collabora since 2013

## Thanks

- This is an updated version of Michael Meeks’ talk from last year

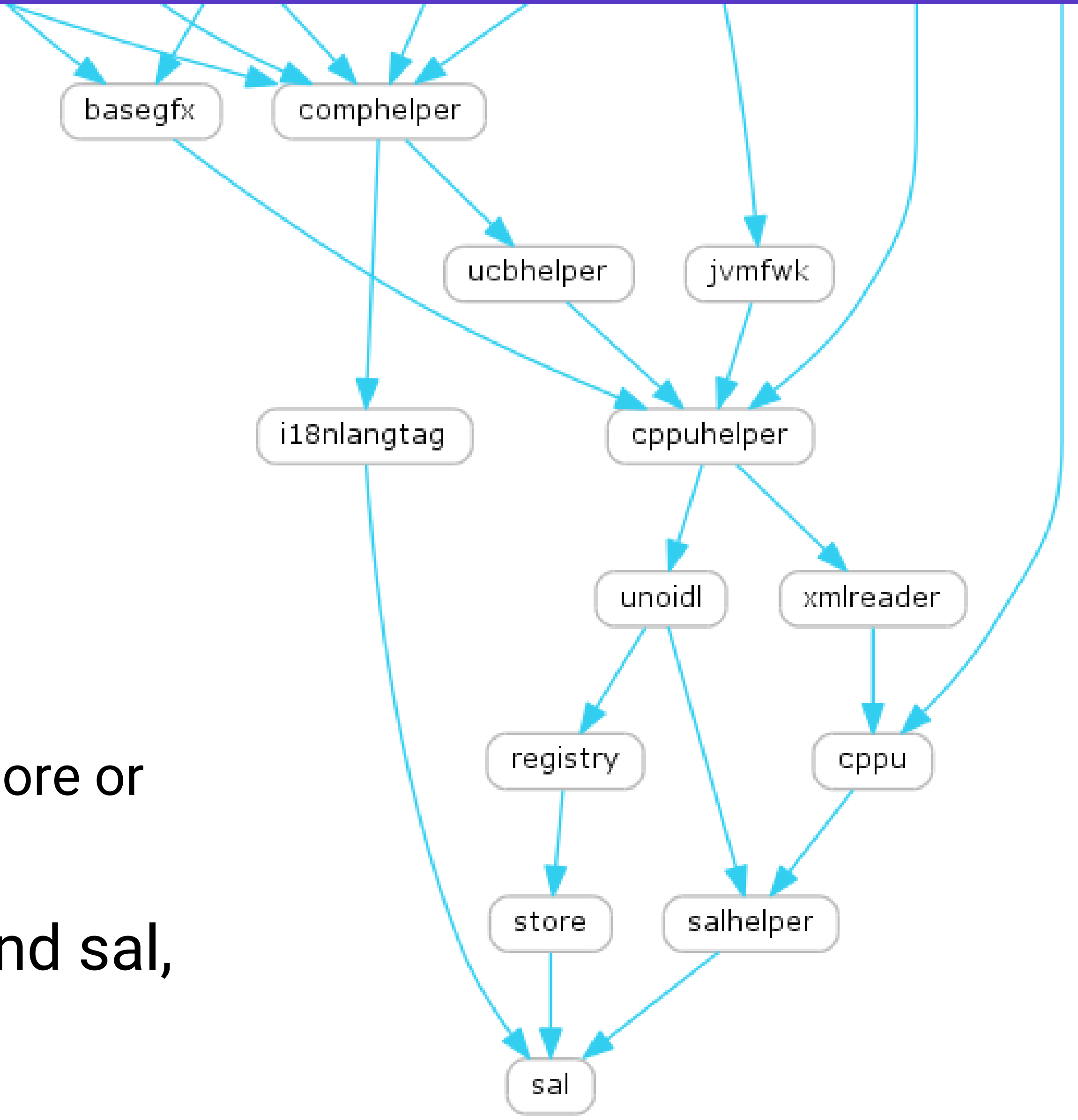
## Overview

- Code-base overview
  - Internal core modules, internal leaf
  - Ignoring externals
- Building / packaging: gnumake, scp2
- Code organisation, git bits
- Keep in mind: this is a 20 years old code-base
  - The quality is much better than you would expect after knowing its age
  - Things continue to improve over time

## Module overview lowest level

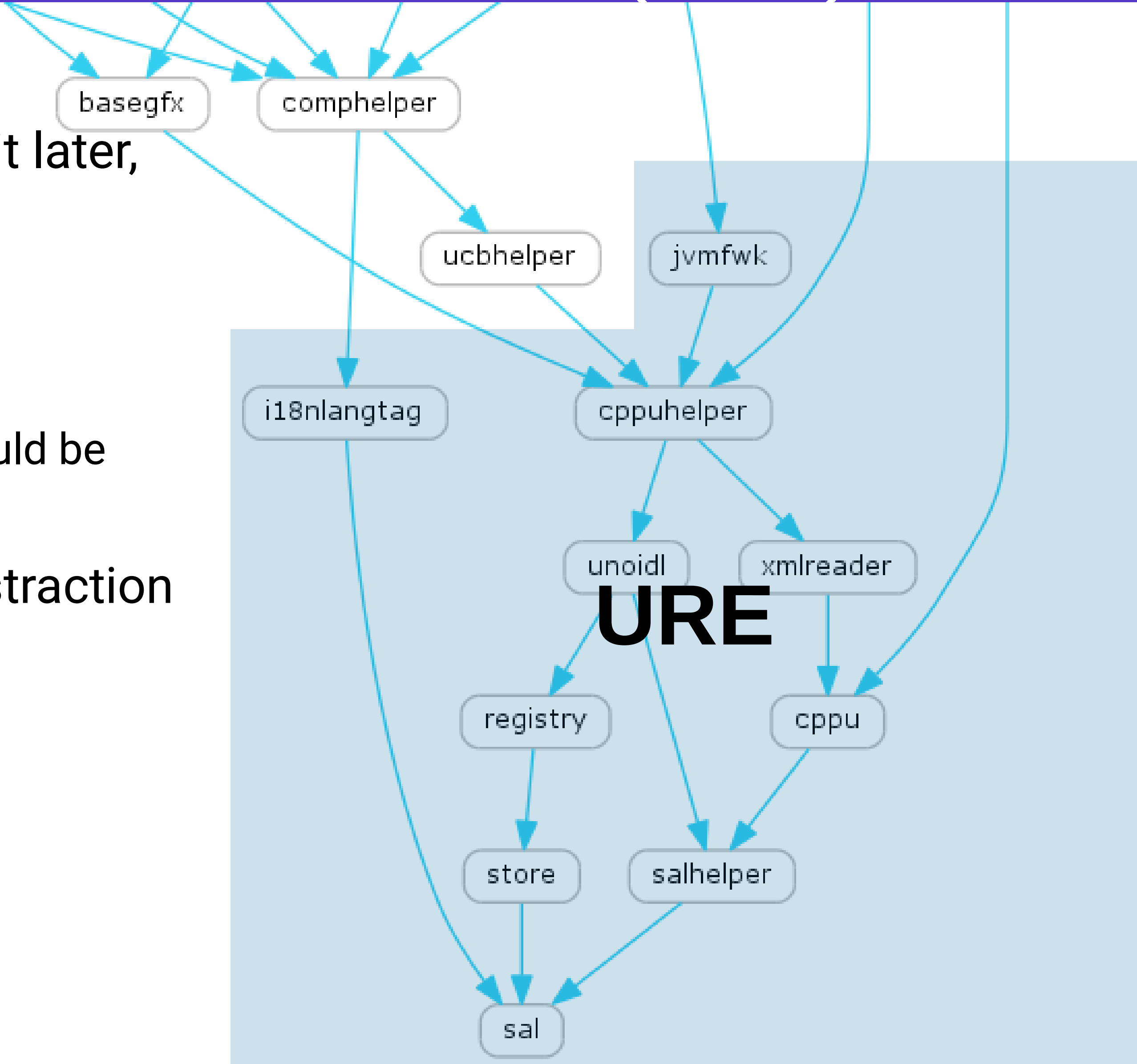
# Internal non-leaf modules: UNO modules

- Module = toplevel dir
  - `make dumps-deps-png`
- Each module has a README
  - e.g. sal/README
- *sal*: at the bottom
  - The system abstraction layer
  - *tools* is an obsolete internal (more or less) duplication of this
- *salhelper*: wrapper code around sal, also part of the URE



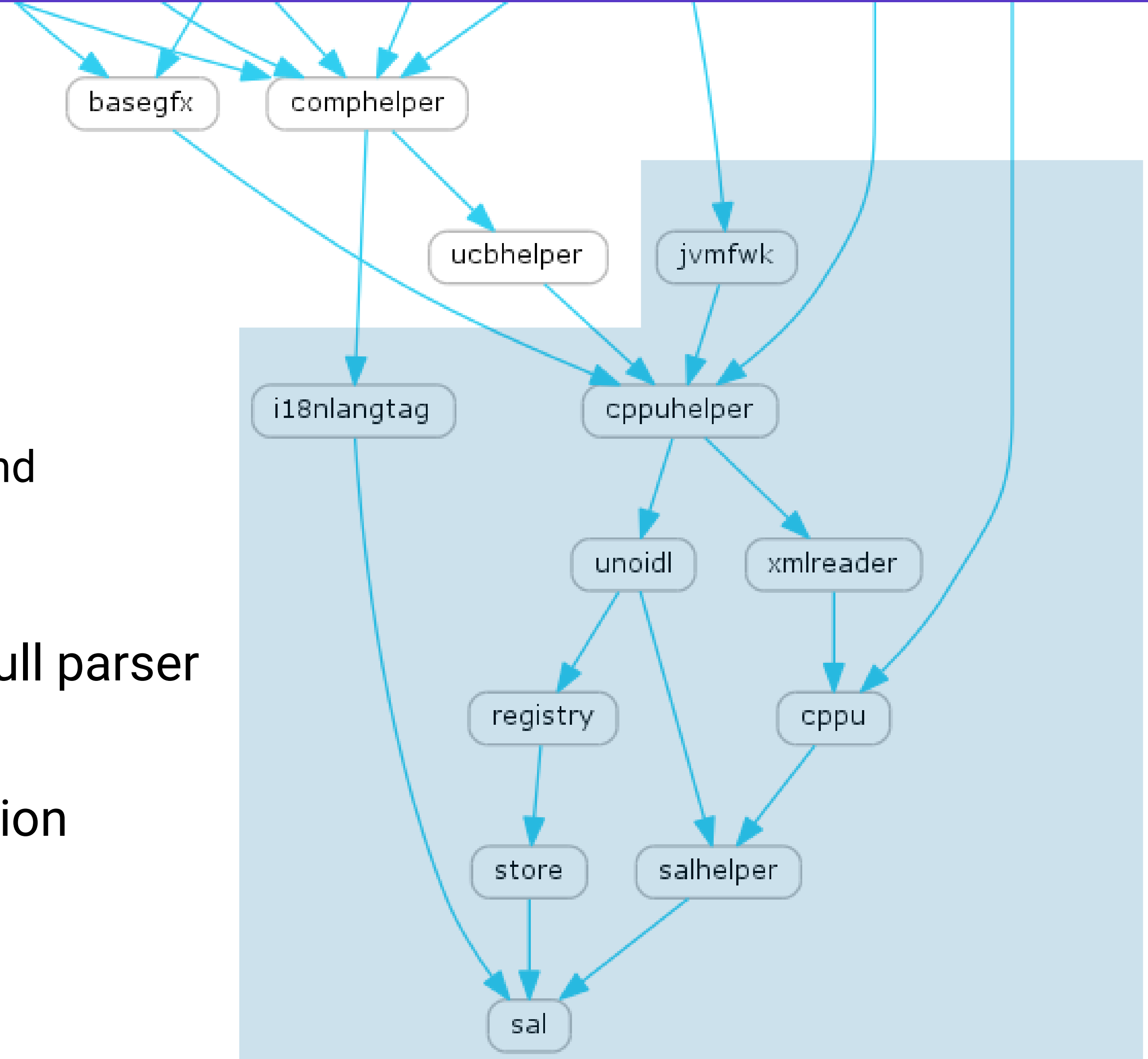
# What is the Uno Runtime Environment (URE)?

- We'll come to UNO in detail a bit later, but for now:
  - Uno Runtime Environment
  - See also JRE, Java Runtime Env.
  - Belongs to the idea that UNO would be reused somewhere else
- Provides an API/ABI-stable abstraction layer for the suite
  - Allows writing C++ extensions
- Modify carefully:
  - Should not change the ABI
  - ABI control via C .map files



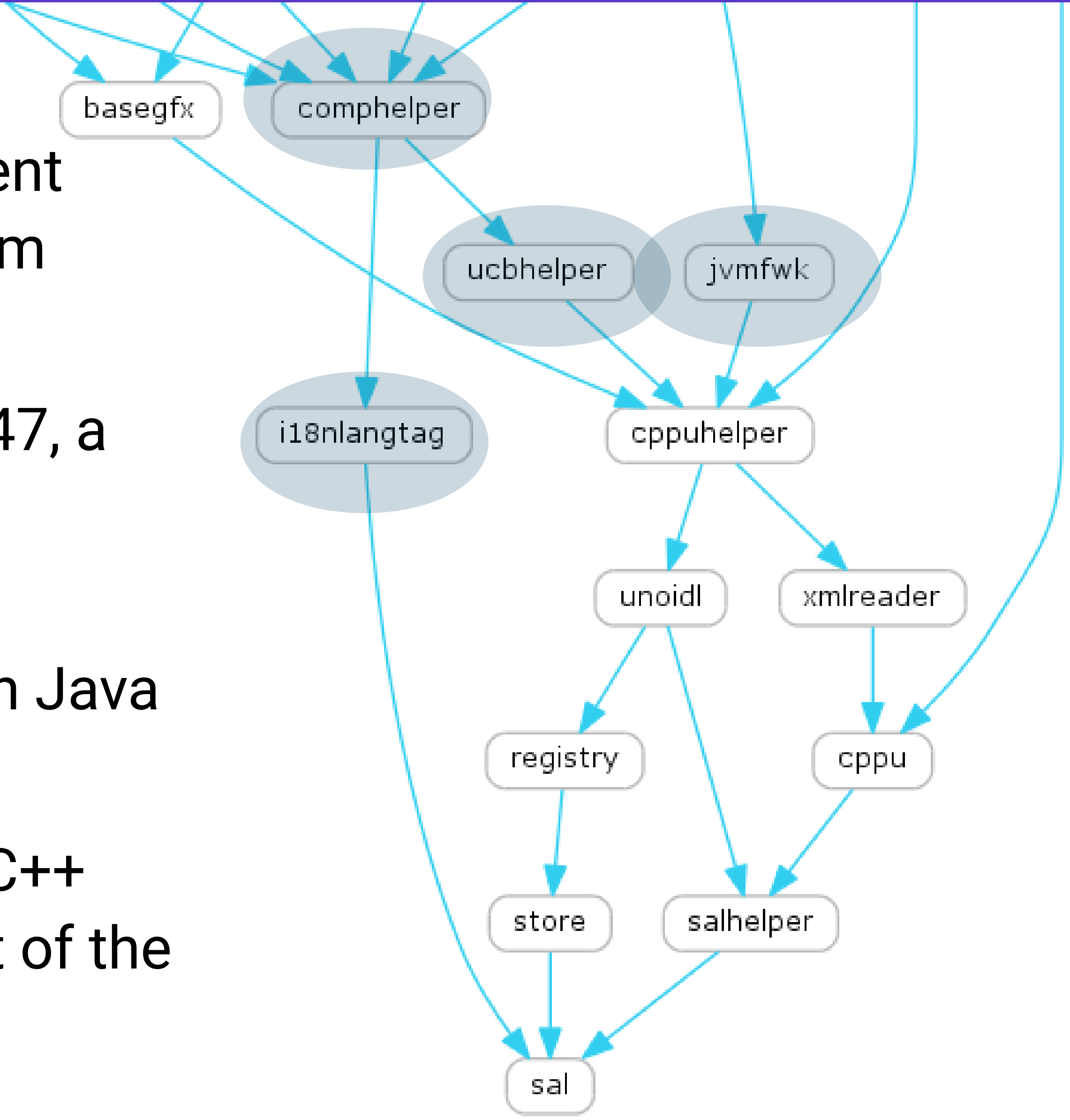
# UNO modules

- *store*: legacy .rdb format
- *registry*: UNO type regisistry
- *unoidl*: a .idl file compiler
- *cppu*: C++ UNO
  - Implements basic UNO types and infrastructure for C++, e.g. WeakImplHelper
- *xmlreader*: very simple XML pull parser
- *cppuhelper*: boostraps UNO, createInstance() implementation leaves here



# More related modules

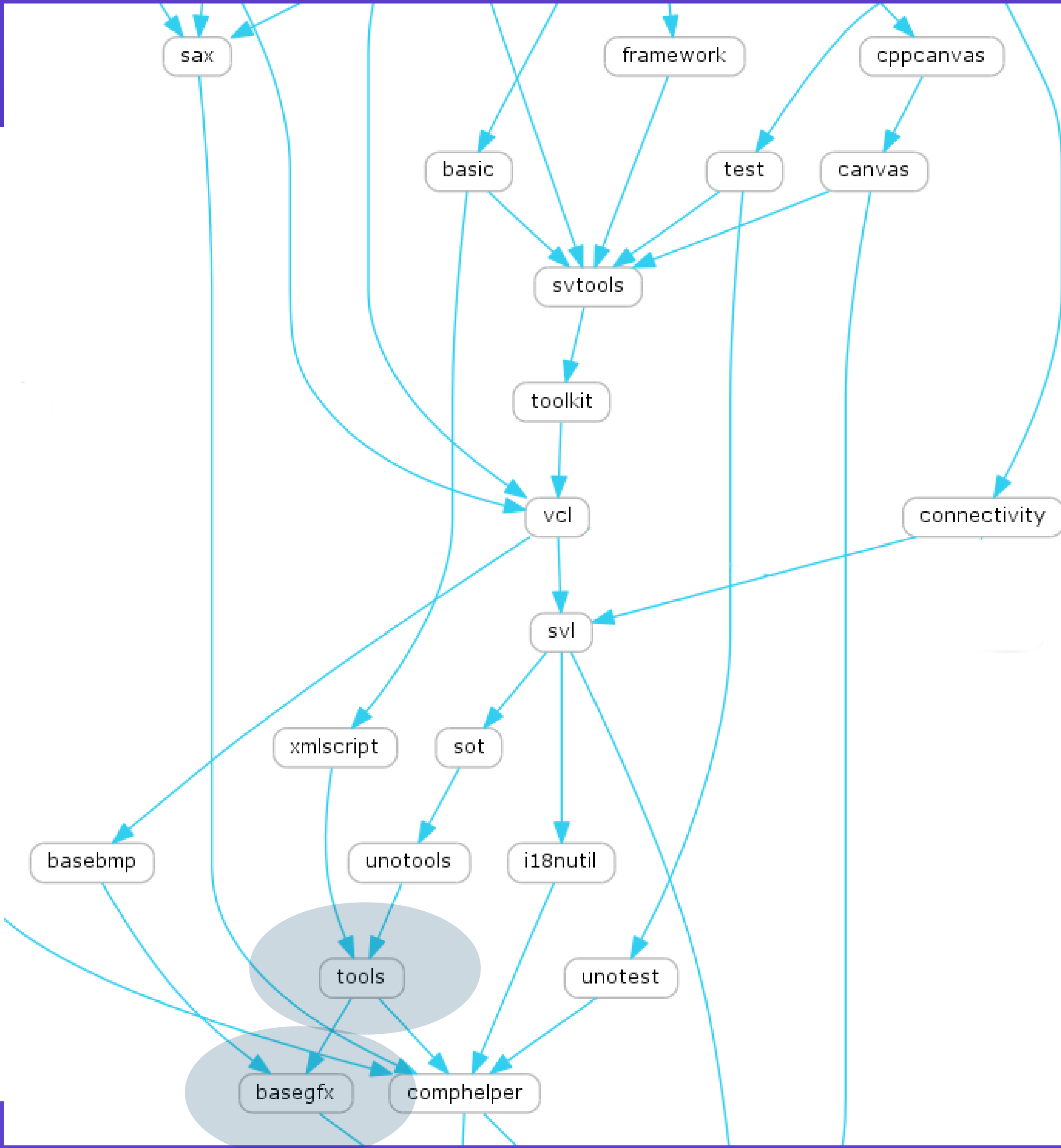
- *ucbhelper*: Universal Content Broker, a Virtual File System abstraction
- *i18nlangtag*: handles BCP47, a powerful way to represent languages/locales
- *jvmfwk*: glue layer between Java and UNO
- *comphelper*: lots of good C++ stuff, intentionally not part of the URE



## Module overview middle level

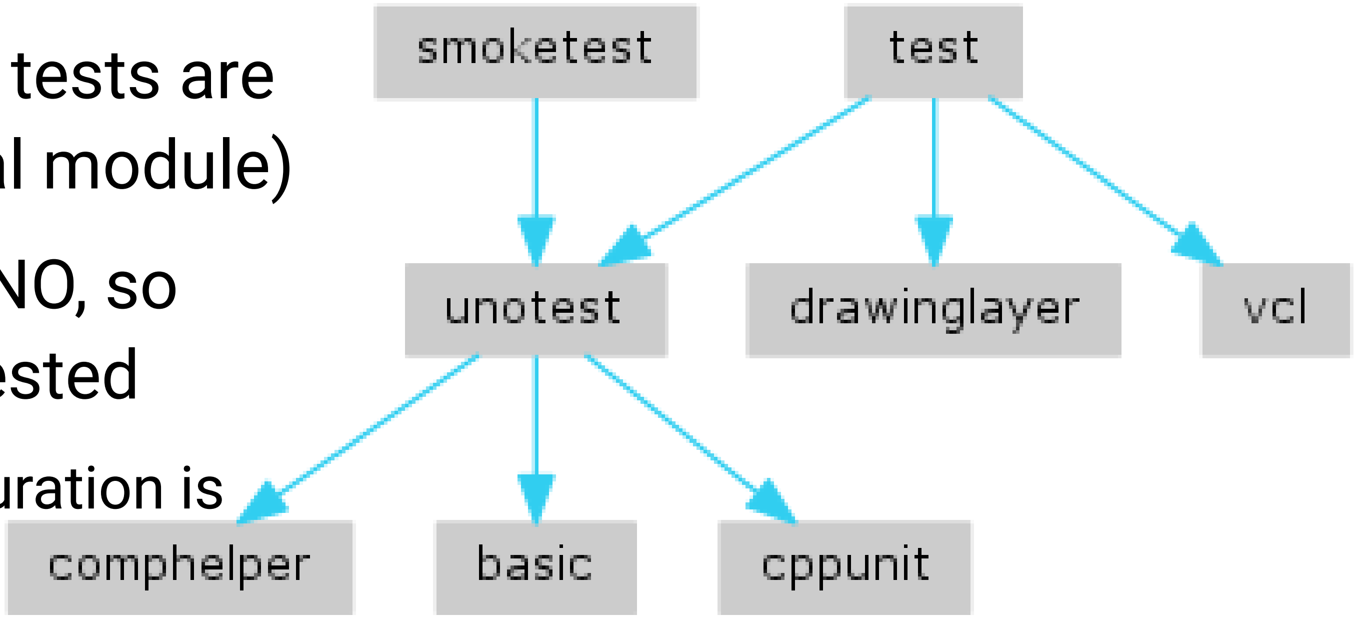
# Internal related modules

- *basegfx*: algorithms and graphic types for basic graphics
- *tools*: more basic types
  - *SvStream*: internal stream type
    - Equivalent of UCB / *sal* file pieces
  - *Color*: e.g. COL_RED
  - *INetURLObject*: URL handling
  - SolarMutex (the big LO lock)
  - Polygon / Polypolygon
  - Date / time classes



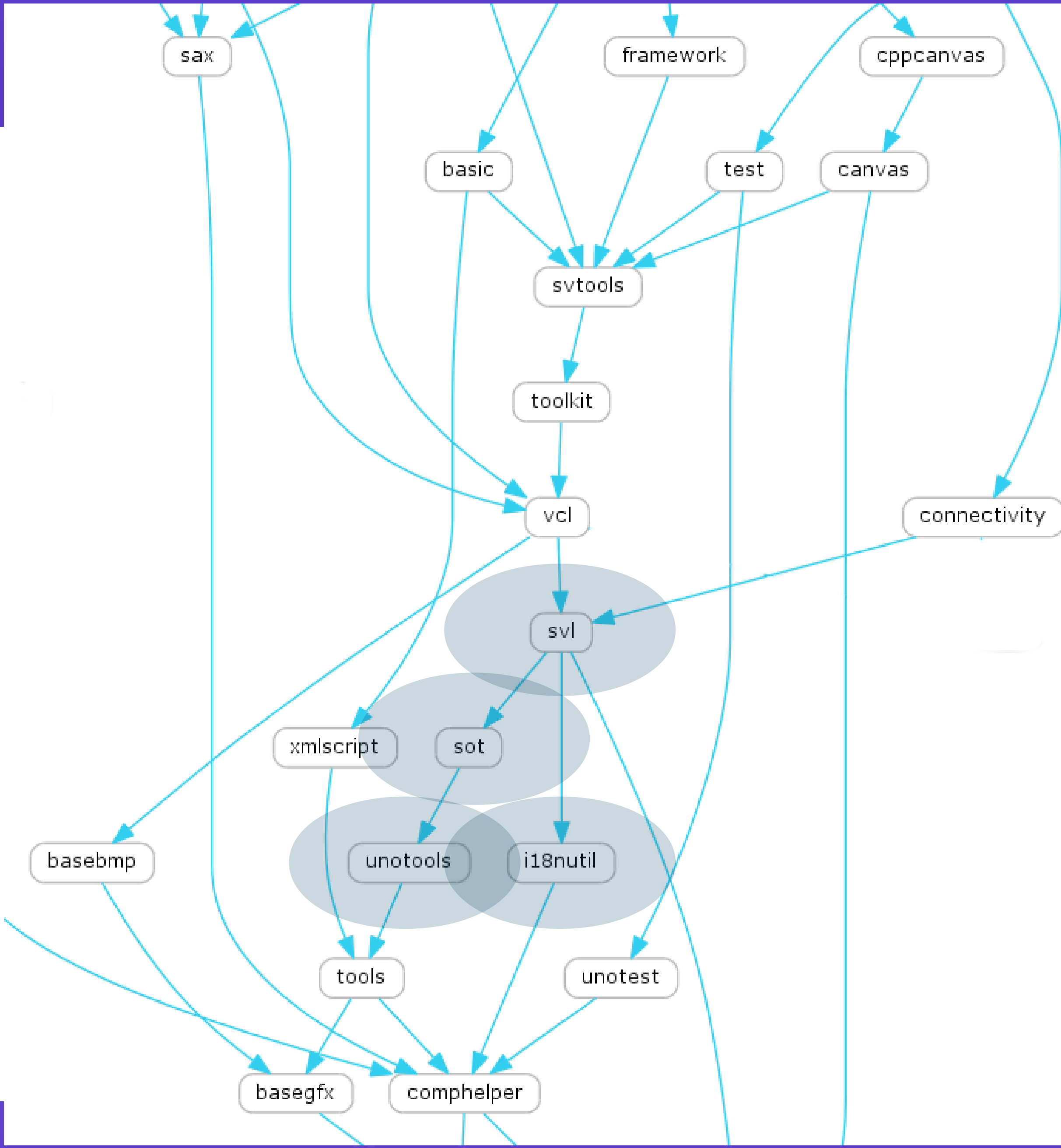
# Unit testing modules

- *cppunit*: all of our C++ tests are CppUnit tests (external module)
- *unotest*: bootstraps UNO, so components can be tested
  - types, services, configuration is available
- *test*: non-UNO part of test setup: VCL, UCB, etc.
- CppUnit_*.mk files in the modules



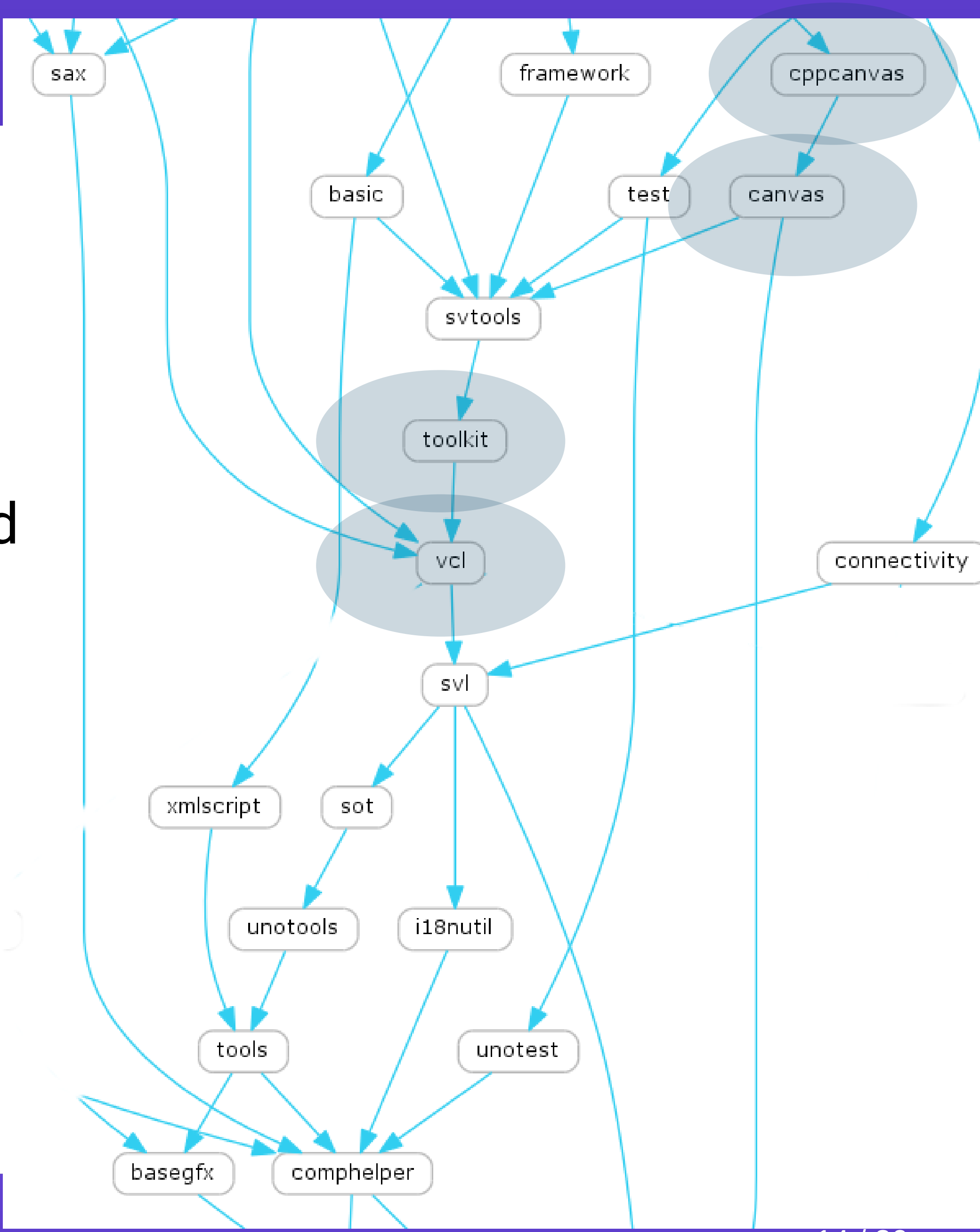
# Other non-graphical modules

- *i18nutil*: C++ wrapper around low-level UNO interfaces
- *unotools*:
  - XStream ↔ SvStream conversion
  - boost::gettext wrapper
- *sot*: OLE2 binary storage implementation
- *svl*: non-graphical parts, which were in svx/sfx2 earlier
  - *SfxItemSet*: an id-any map
  - undo/redo
  - crypto pieces



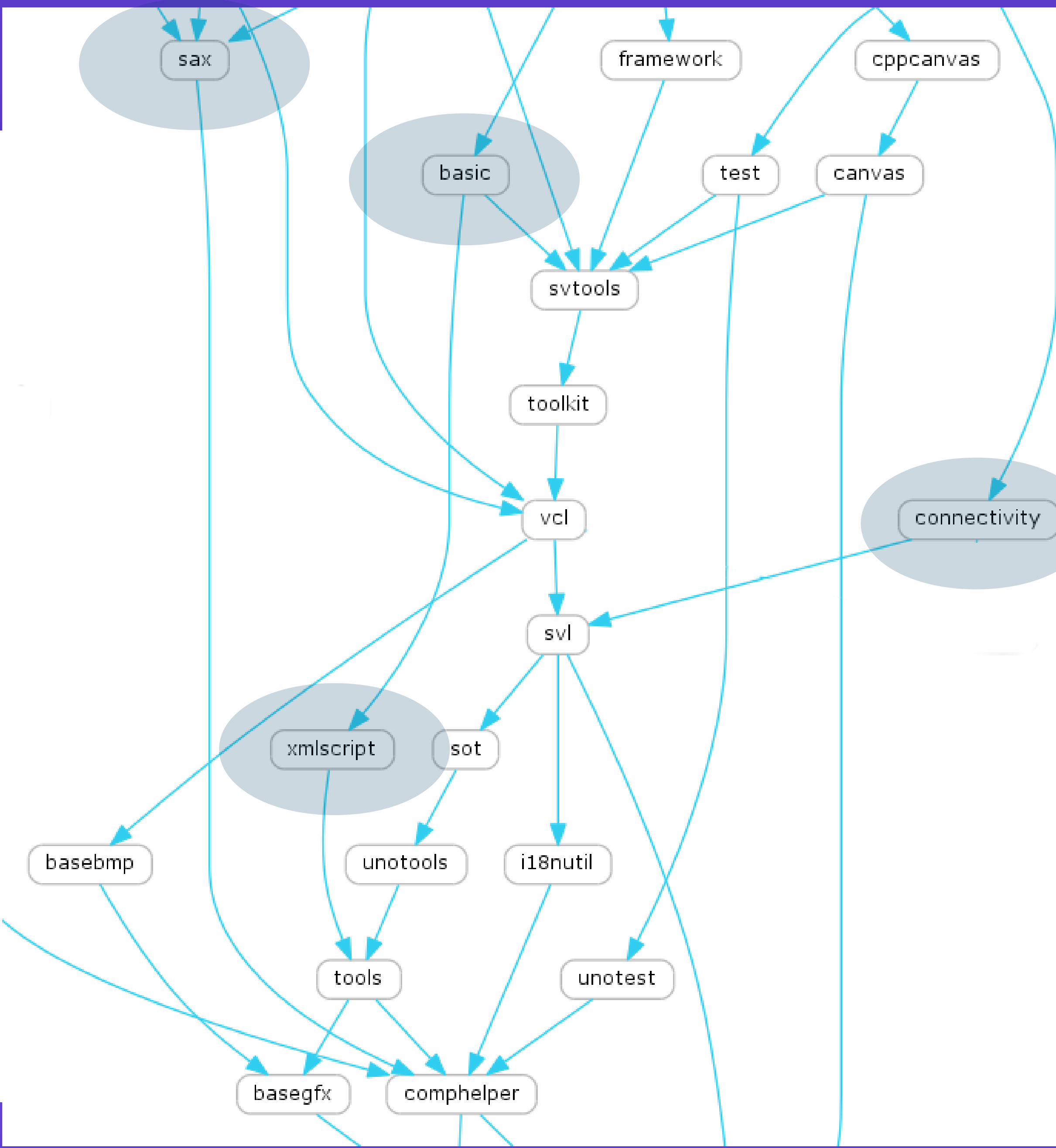
# Graphical / toolkit modules

- *vcl*: Visual Class Libraries, the LibreOffice graphical toolkit
- *toolkit*: UNO API wrapper around vcl
- *canvas*: rendering UNO API that supports alpha and anti-aliasing, used by *slideshow*
  - DirectX, Cairo and VCL backends
- cppcanvas: wrapper around the UNO API
- emfio, svgio: drawinglayer-based EMF/SVG import



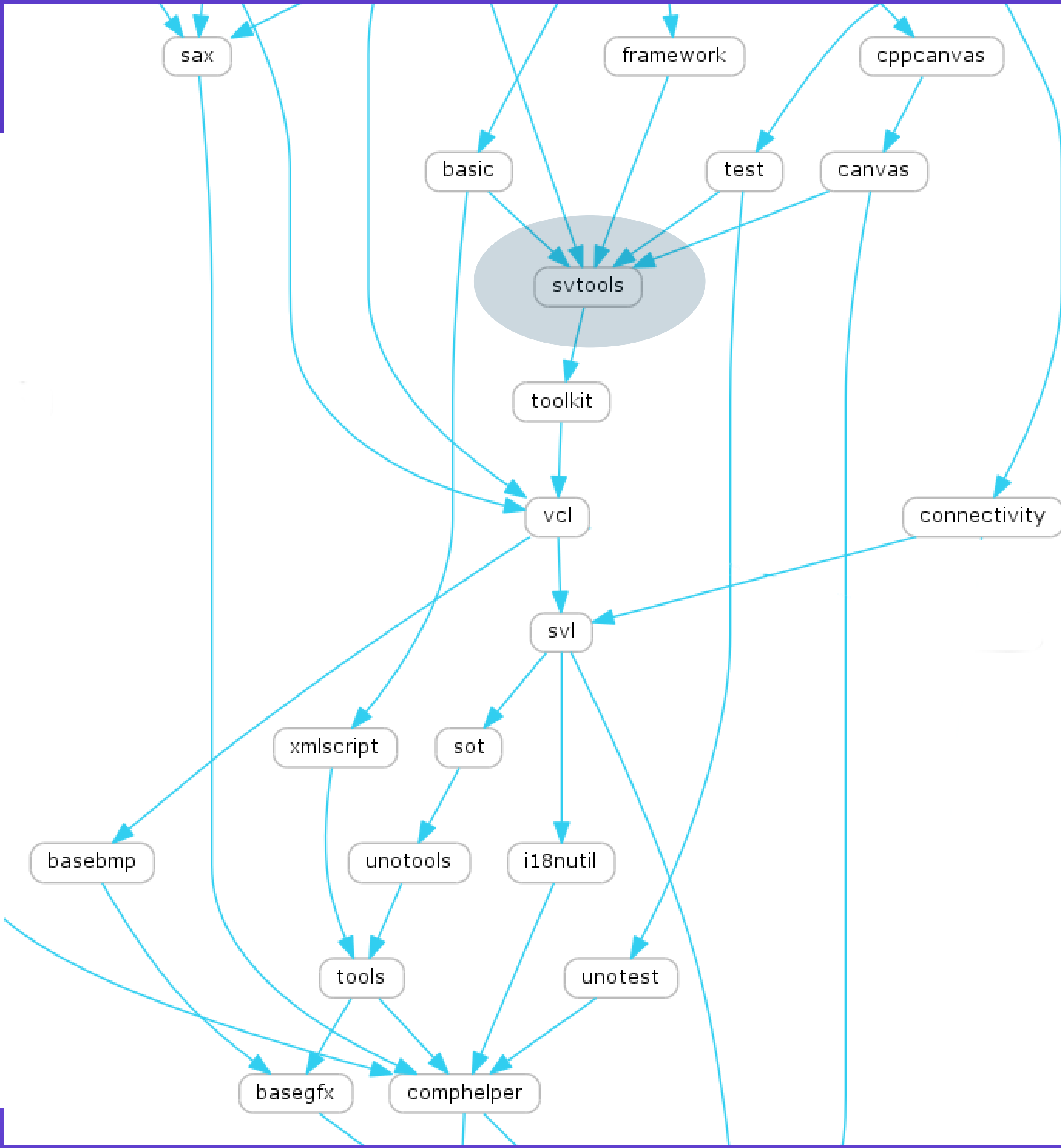
# Non-graphical modules

- *basic*: StarBasic interpreter
- *xmlscript*: Basic dialog loader/serializer
- connectvity: database drivers
  - pgsql, mysql, address books, jdbc, odbc, Calc/Writer
- *sax*: libxml2 wrapper, provides the fast parser (a SAX API)



# Graphical modules

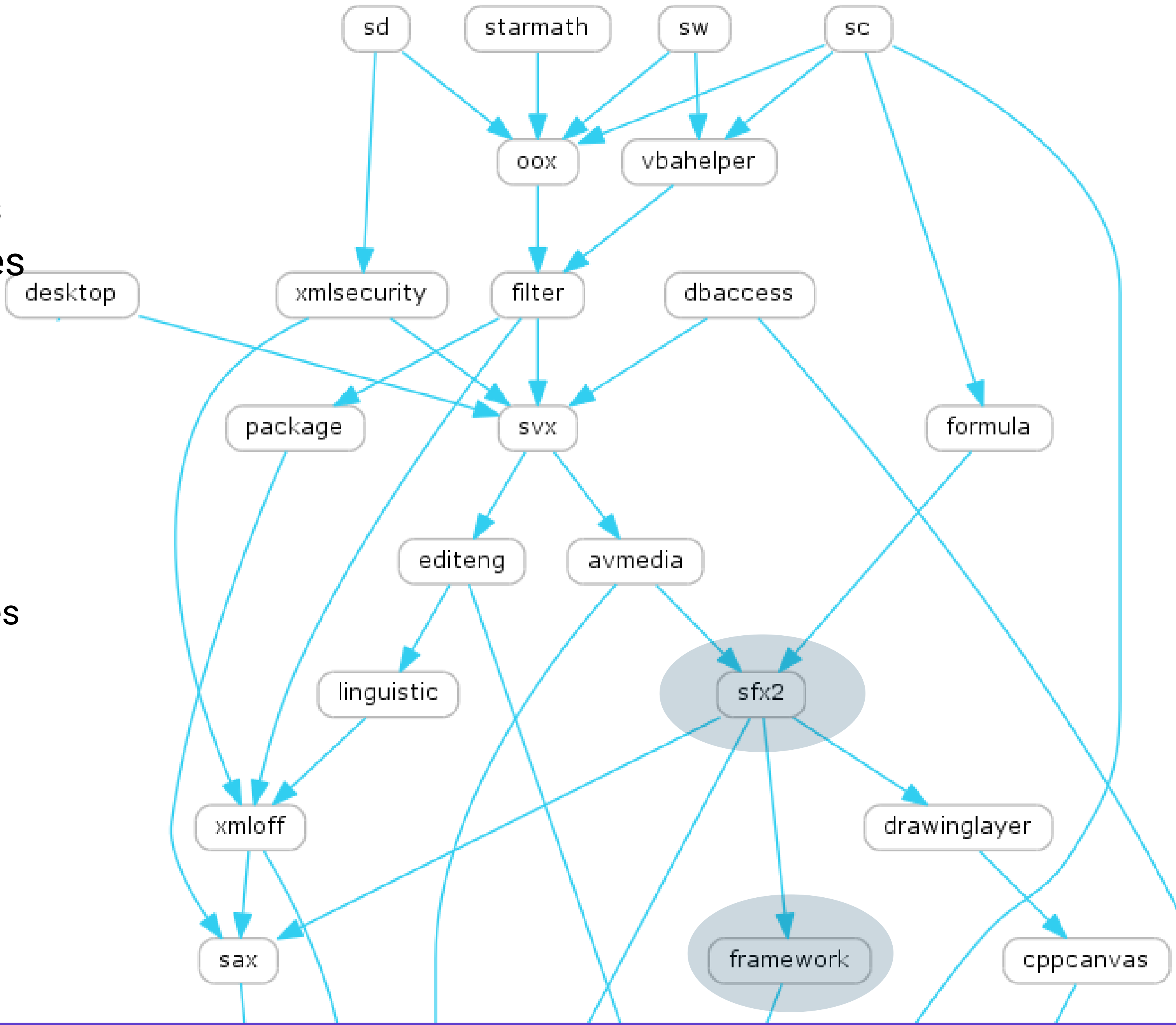
- *svtools*:
  - Tree / list VCL widgets
  - Table widget
  - Dialog helpers (e.g. closing listener)
  - Accessibility helpers (e.g. accessible ruler)
  - *configmgr* wrappers
  - Printing options
  - Image map handling
  - Wizard framework



## Module overview
## Upper level

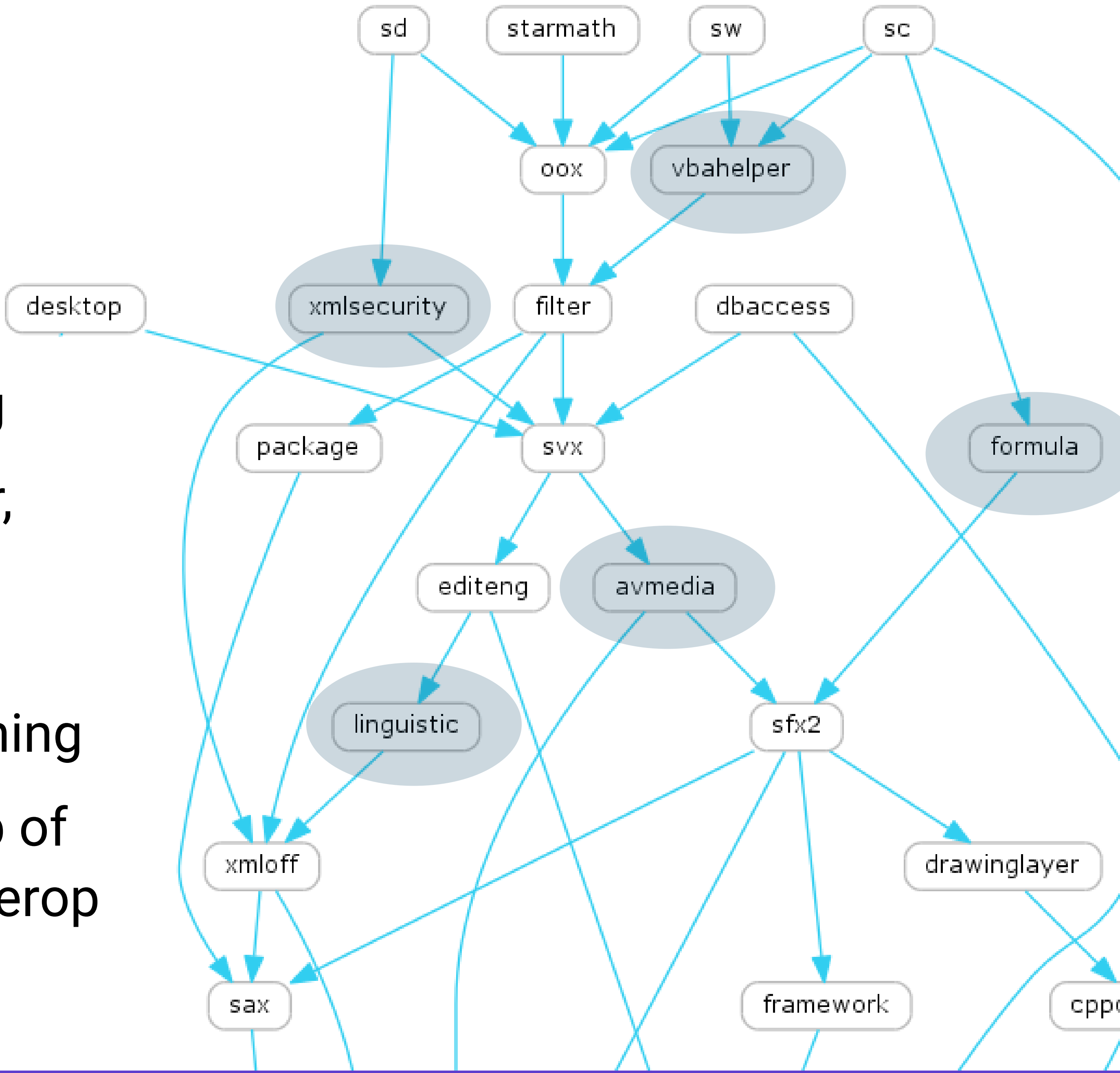
# Document / frame modules

- *framework*: docking, toolbars, menus, status bar, sidebars, task panes
- *sfx2*: core of the app
  - *SfxMedium*: load / save logic
  - Object / view management
  - Dialog helpers: tab pages
  - Document meta-data dialogs
  - Template management
  - Shared style code



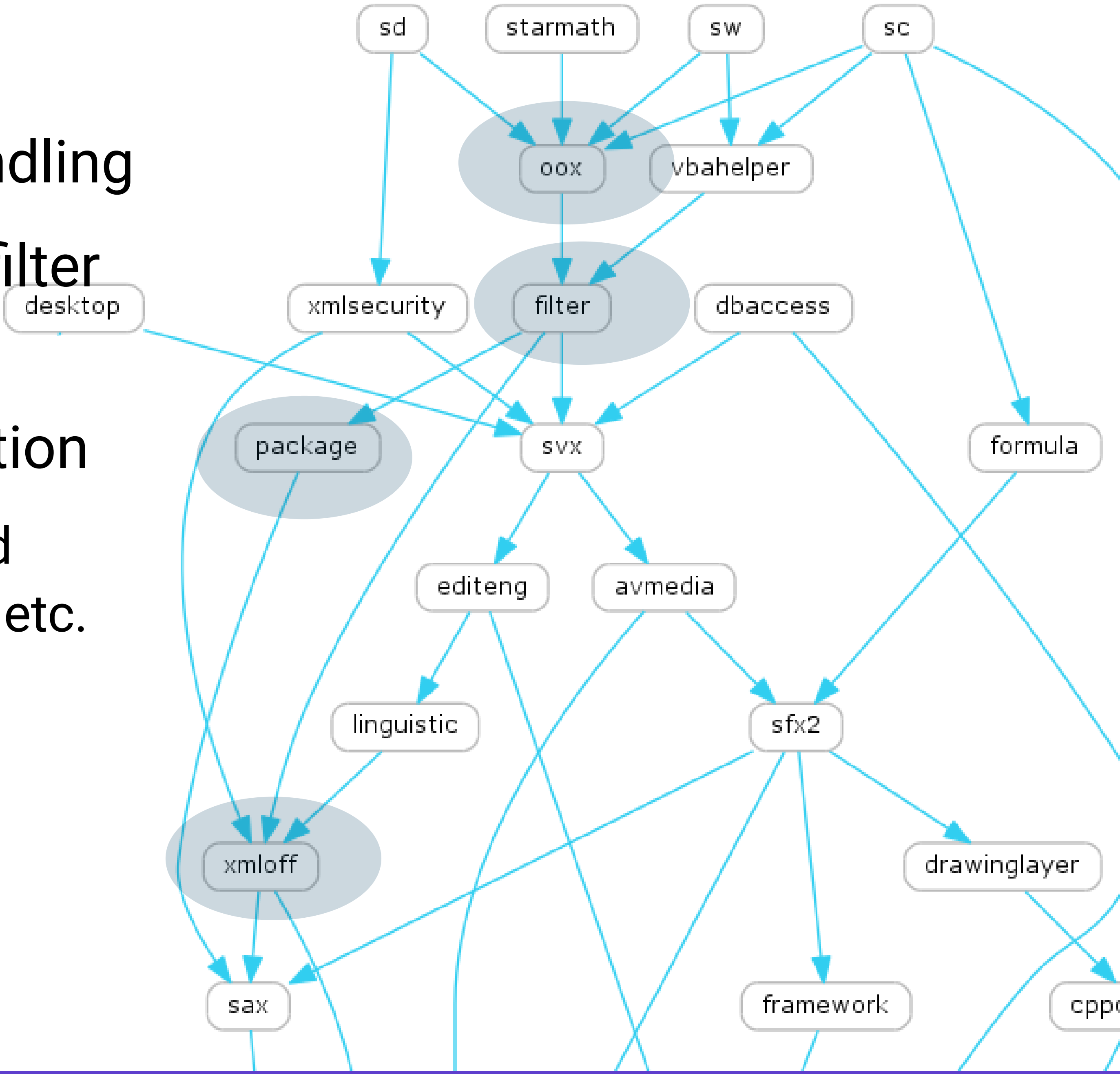
# Other document modules

- *formula*: shared code between sc and reportdesign
- *avmedia*: video playing
- *linguistic*: spellchecker, hyphenating
- *xmlsecurity*: ODF/OOXML/PDF signing
- *vbahelper*: code on top of *basic* for MSO VBA interop



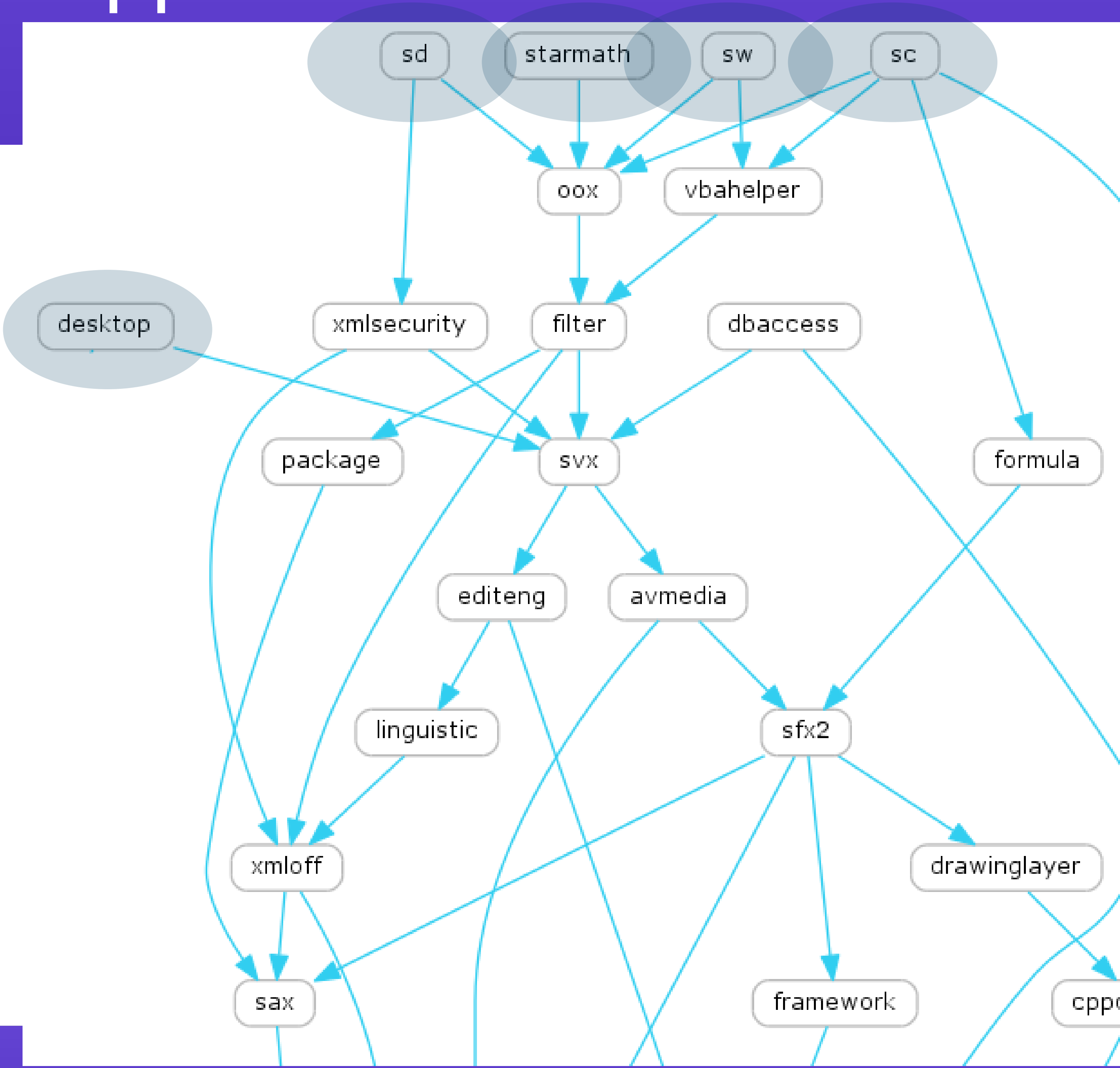
# Load / save (filter) logic

- *package*: ZIP file handling
- *xmloff*: shared ODF filter code
- *filter*: filter configuration
  - Also: flat ODF, shared binary MSO support, etc.
- *oox*: shared OOXML support:
  - VML, drawingML



# Applications

- *desktop*: StarDesktop
  - main() lives here
- *sd*: StarDraw (Draw, Impress)
  - drawings, presentations
- *sw*: StarWriter
  - Word processor
- *sc*: StarCalc
  - Spreadsheet



## This is a simplified picture

- These all were non-leaf nodes
- This is a linking dependency graph
  - UNO is a great dependency breaking tool
- Modules still missed:
  - *cui*: Common User Interface, common dialogs
  - *chart2*: charting support
  - *slideshow*: the piece that renders your Impress slideshow
  - *solenv*: build infrastructure

## Building, packaging

## Build: configure and compile

- autoconf / configure – pretty standard
- autogen.sh – a wrapper around autotools
  - Builds & runs the configure script
  - Keep your parameters in *autogen.input*
  - Builds:
    - config_host.mk from config_host.mk.in, contains all the environment variables
    - *config_host/*.h*, C++ headers

## Android and Online build

- Android
  - Inside *core.git*, configure with --*with-distro=LibreOfficeAndroid*
  - See *android/README*
  - Resulting .*apk* file under *android/*.
- Online
  - Uses autotools, in separate *online.git*
  - Link to *core.git*: --*with-lo-path*

## Build: gnumake

- Gnumake is used in creative ways
  - Code is in *solenv/gbuild/*
  - Each module has its own Makefile
    - You can build each independently after a full build
    - All rules are built by $(call Function,…) magic, we don’t use any of the build-in rules
    - If something is compiled, we have an explicit rule for it somewhere, you can find it
- Following the rules is expensive due to non-named function parameters (*$(1)*, *$(7)*)

# Build: output

- We build an installation set in *instdir/*
  - *instdir/program*
  - Contains something you can run in-place
  - *make && instdir/program/soffice* – it works
- *workdir/*
  - Object files, build intermediates here
  - Generated headers
  - Unpacked external source code
- So *make clean* can just remove instdir/workdir

## Build-related modules

- Postprocess
  - Packimages
    - Using solenv/bin/pack_images.py – build icon theme .zip and sort it by access pattern
  - CustomTarget_registry.mk
    - Builds configuration files from officecfg/.
  - Rdb_services.mk
    - Builds services.rdb file .component files
- Officecfg/
  - Home of all defaults / office configuration / settings

## Internal module organization

- *include/*
  - All global includes live in *include/<module>/*
- e.g. *sfx2/inc/* – these are includes local to a module
  - *sfx2/source/* – source code for the module
  - uiconfig/ – UI descriptions (dialogs, toolbars, menus)
  - sdi/ – descriptions of slots / actions (UNO commands)
  - qa/ – unit tests, test file data, etc.
- Lots of things moved over time:
  - git log -u --follow is your friend

## Summary

- This was very high-level
  - Intentionally, so you can get the big picture
  - Hopefully still useful
- We have a lot of modules
  - You can safely not know about the majority of them.
- Slides: https://vmiklos.hu/odp