Dyalog User Conference 2006

Dyalog/Wine :

Freeing the POSIX port

About me

nicolas@dyalog.com

- Introduced to APL in 2003
- French Engineer degree in “scalar IT” and Electronics in 2006.
- Joined Dyalog Ltd. in late August 2006

About Wine

http://www.winehq.org

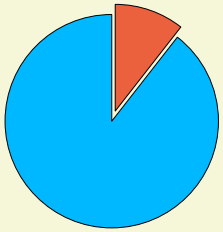
- **W**ine **I**s **N**ot an **E**mulator
- Emerged in 1993 as a Windows 3 API implementation
- Provides a Win32 API on top of X and POSIX
  - 90 % of the API is implemented
  - 90 % of the task remains to be done
- Runtime for Win32 binaries on x86
- Libraries for platform-specific compiling

Having Dyalog on a Platform

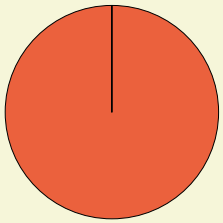
- Many Dyalog components use the Microsoft Windows API.
- Then requires two modules :
  - Dyalog itself
  - A implementation of the Windows API
- How ? For what cost ?

Dyalog/W

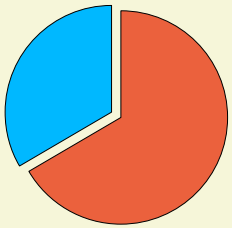
Developer License

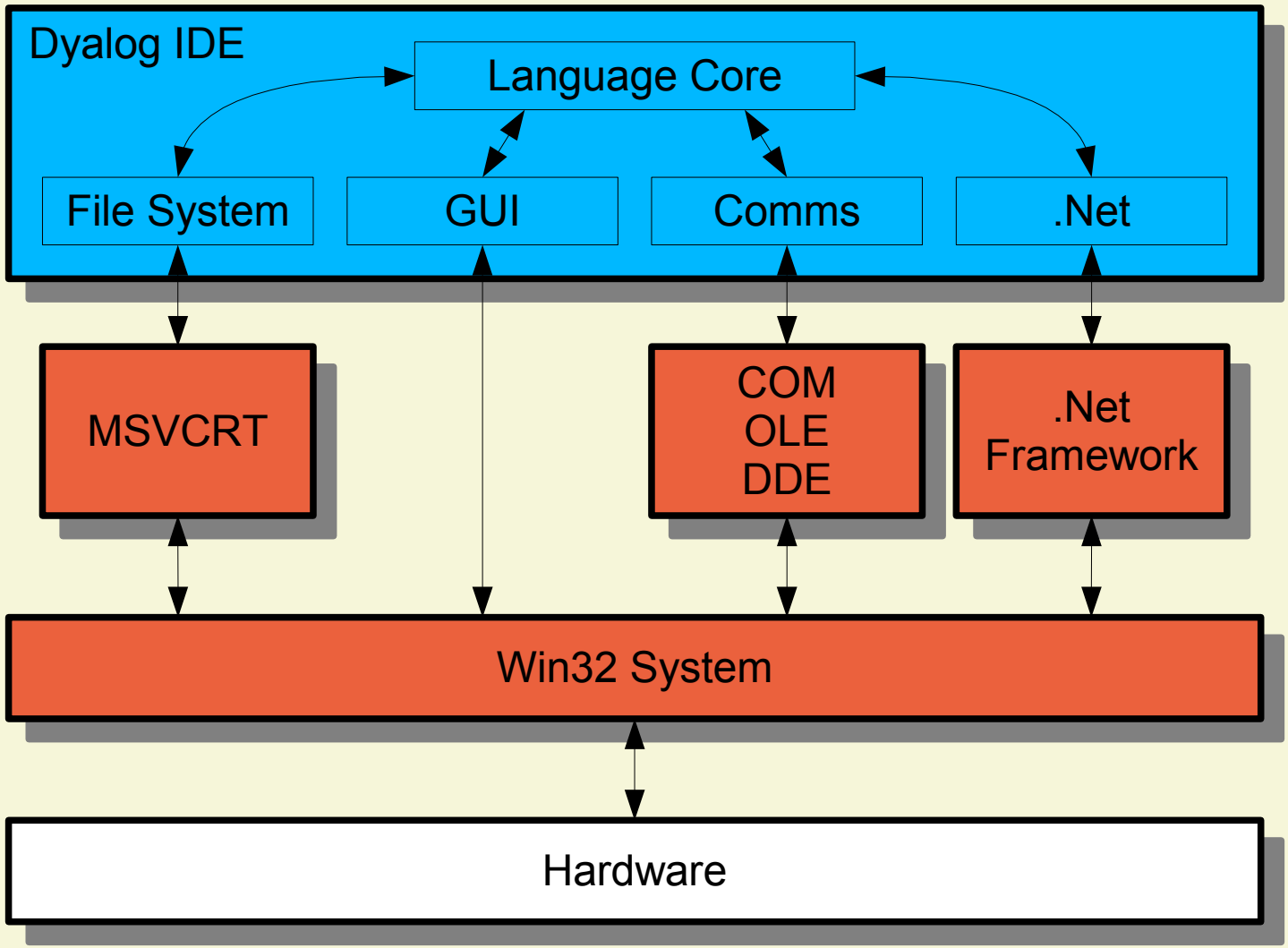


Educational License



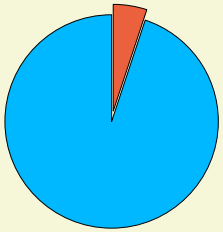
Runtime License



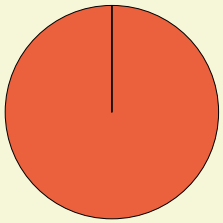


Dyalog/M

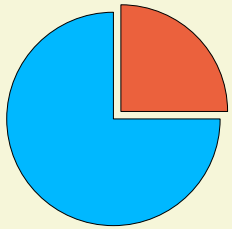
Developer License

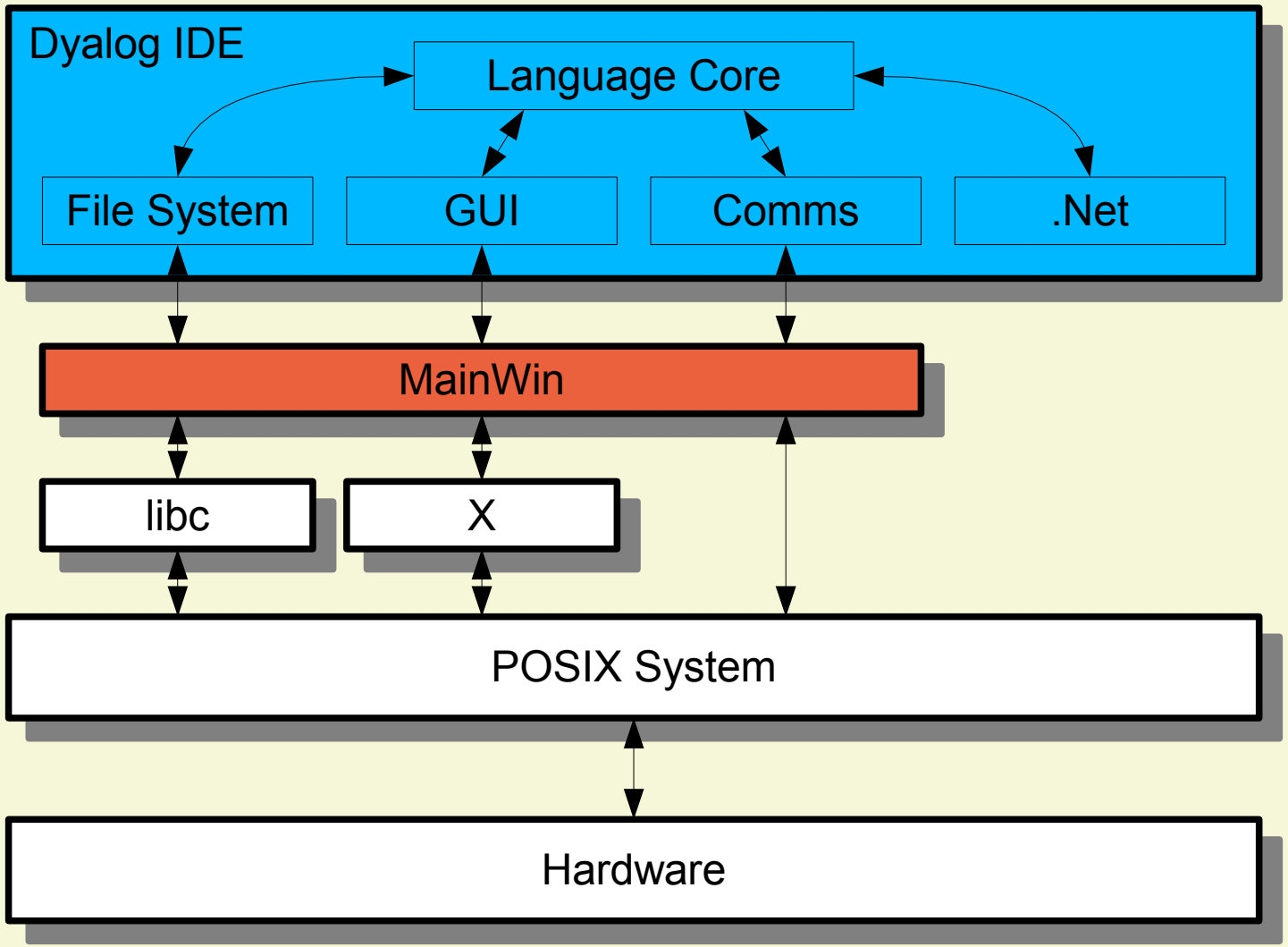


Educational License



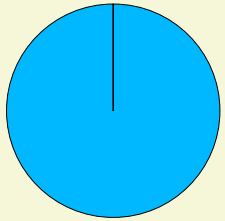
Runtime License



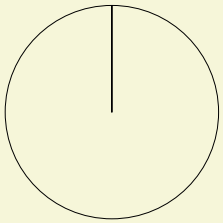


Dyalog/Wine

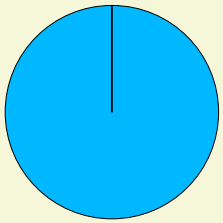
Developer License

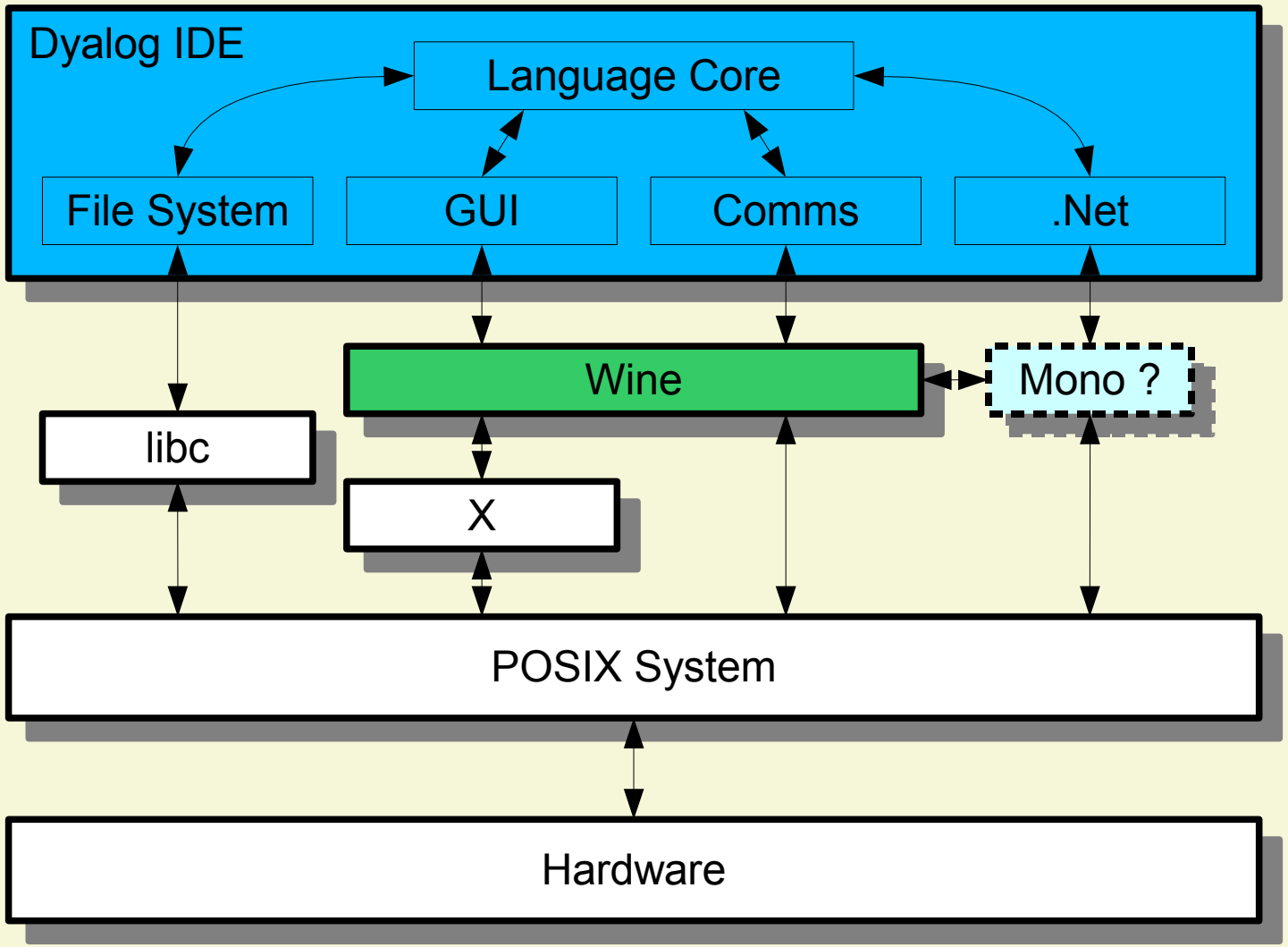


Educational License



Runtime License





Pros and Cons

- Wine vs MainWin:
  - Free of charge
  - Open Source
- MainWin vs Wine
  - Really Cross-Platform
  - Stability in evolution
  - Full implementation of Windows API
- Wine and MainWin vs Windows :
  - Using native UNIX libc and signals
  - X GUI for Remote execution
  - Support Team if needed (CodeWeavers ; MainSoft)

User aspects of Wine

- Wine is in development
- Input and Output
  - Windows-like keyboard and font handling
  - Fonts sent as graphical data to X server
- UNIX and Windows Dynamic libraries
- Discussion : support for COM, OLE and DDE

Looking Forward...

- Wine on other platforms than Linux/x86
  - Other UNICES : likely to work
  - 32-bit, little endian : likely to work
  - 64-bit : not for now
  - big endian : place your bets...
- Using Mono :
  - Natively beside Wine or MainWin
  - On top of Wine

And now...

"In theory, there is no difference between theory and practice. But, in practice, there is."

Jan L. A. Van de Snepscheut (1953-1994)

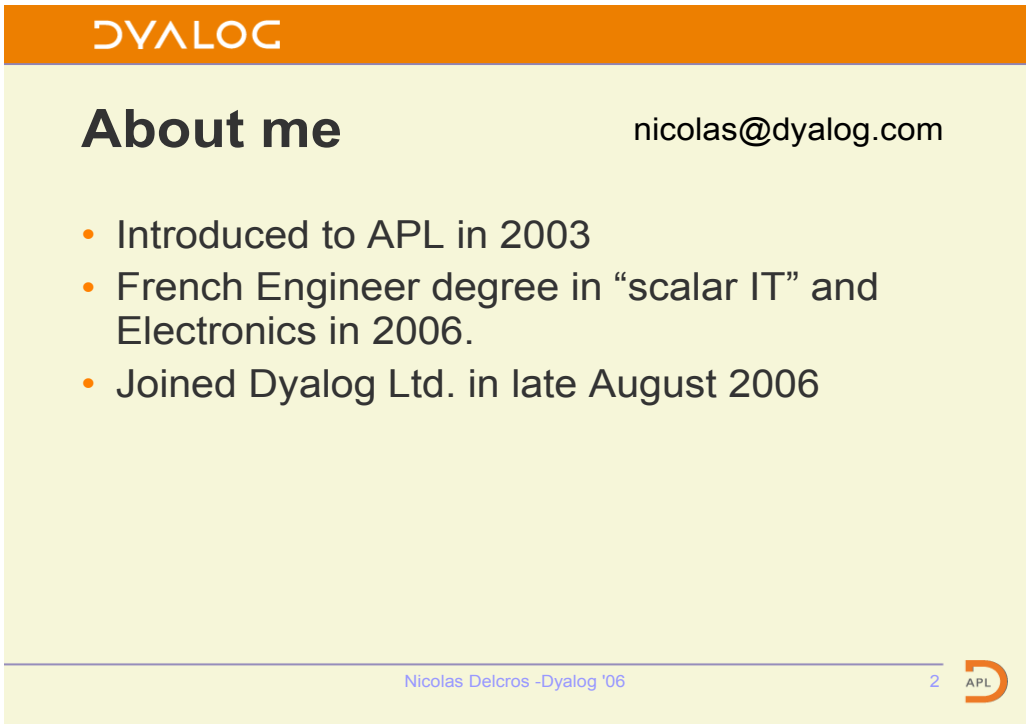
DYALOG

Dyalog User Conference 2006

Dyalog/Wine :

Freeing the POSIX port

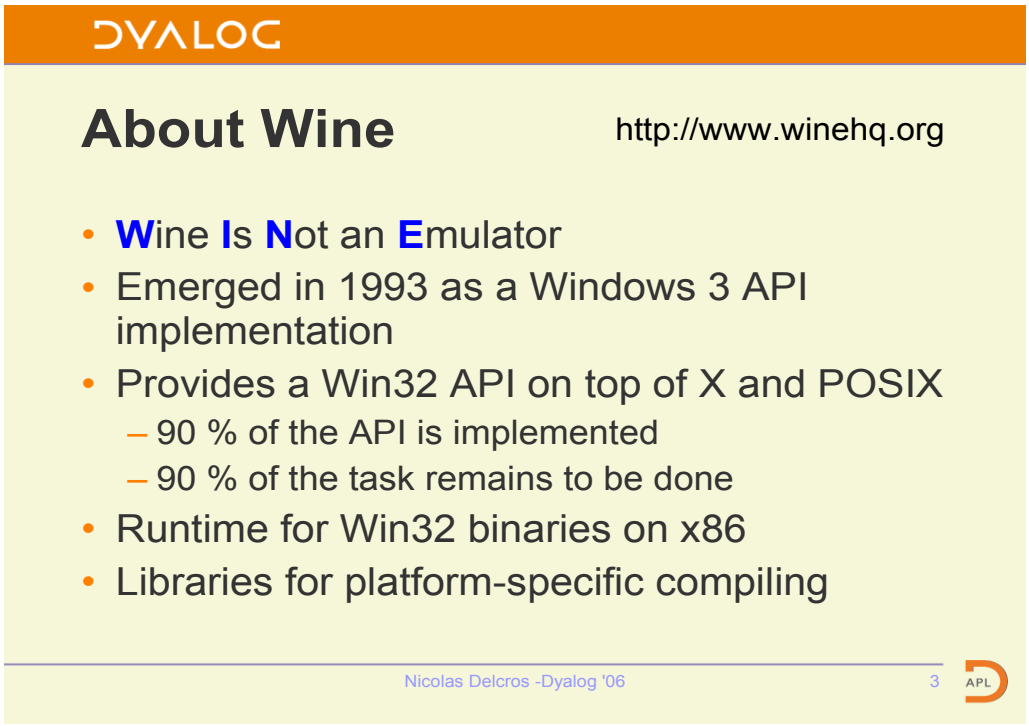
Please forgive my approximate English.



I have been more or less unsuccessfully looking for APL jobs since i know APL.

The Dyalog opportunity saved me from scalar IT.

Making the Wine port was a good way to getting used to working on the interpreter.

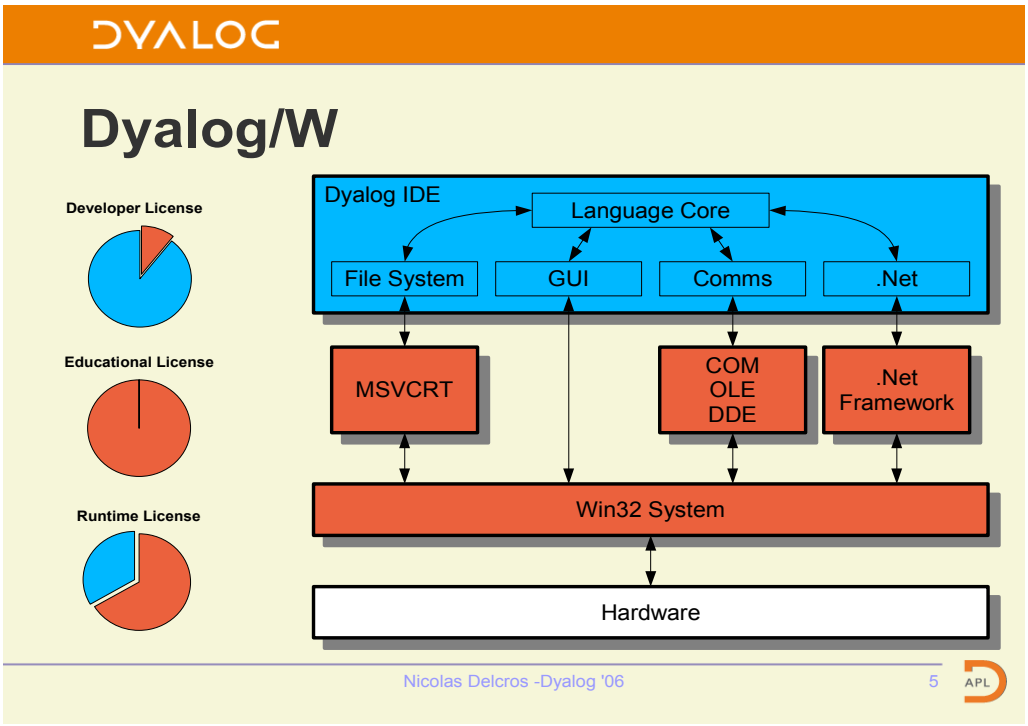


Runtime acts as an emulation of the Windows behaviour

Library is a tool to turn a Windows application into a real POSIX application, which is what MainWin was doing, and what we need.

Having Dyalog on a Platform

- Many Dyalog components use the Microsoft Windows API.
- Then requires two modules :
  - Dyalog itself
  - A implementation of the Windows API
- How ? For what cost ?



Costs :

In Blue : Dyalog itself

In Red : What Dyalog requires

In white : What is up to the user.

Figures for Windows are questionable, because nowadays many computers are shipped with Windows.

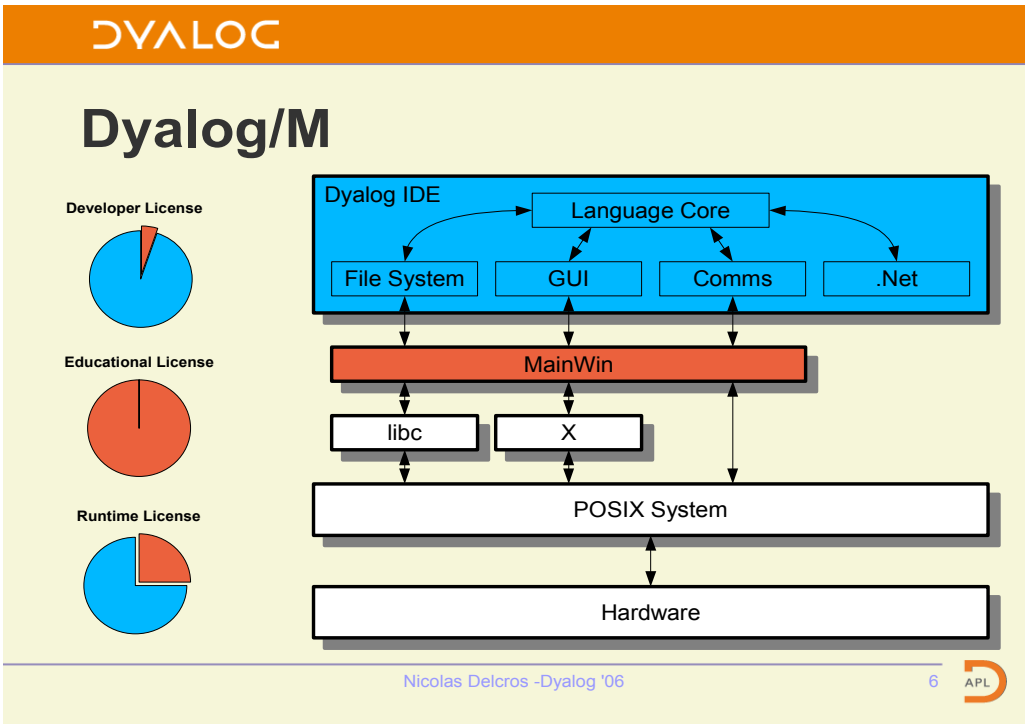
However, the cost of the Windows license cannot be ignored, especially when choosing the platform on which the runtime version of your application will be delivered.

Base :

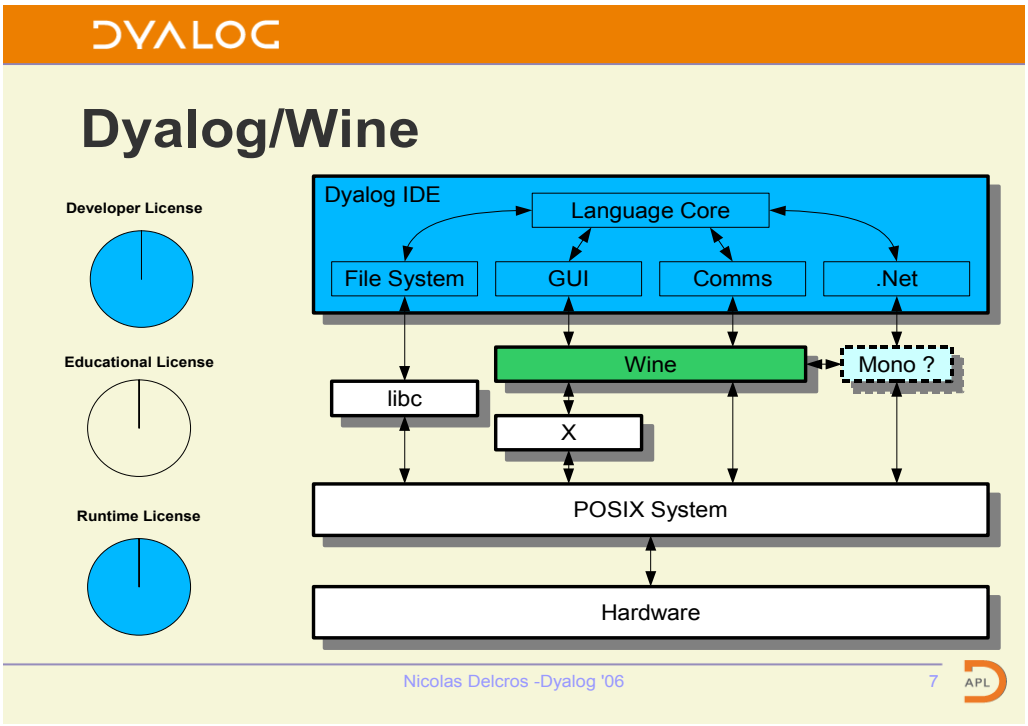
£100 for Windows OS

£850 for developer license

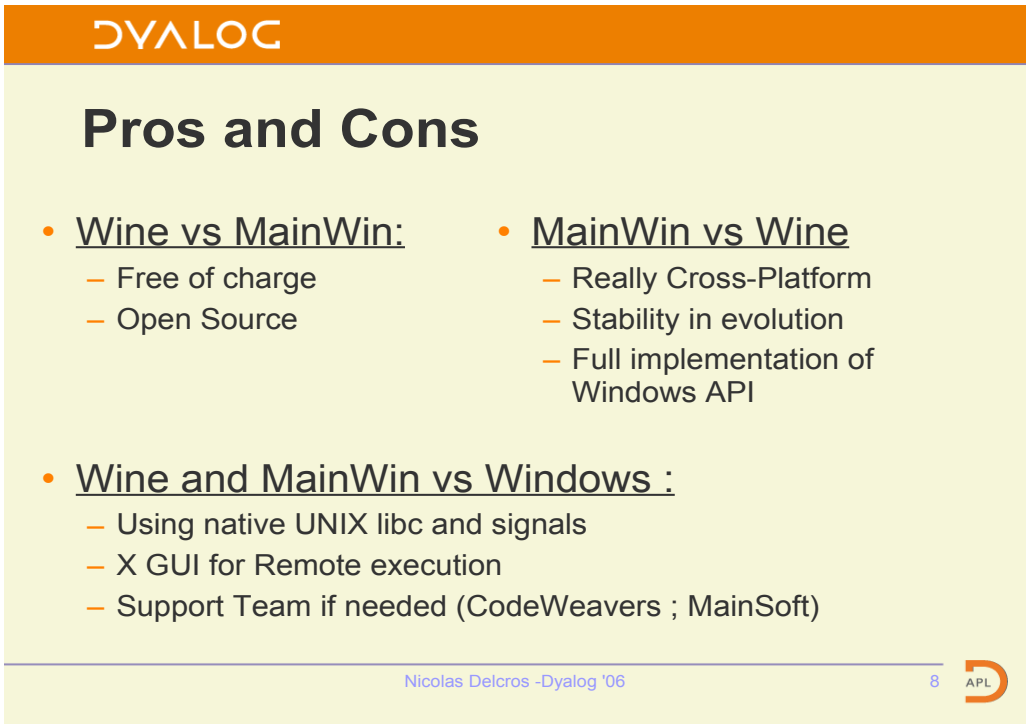
£50 for runtime license



These figures are estimations (the MainWin license policy is quite complicated and not so easy to compute per sold license)



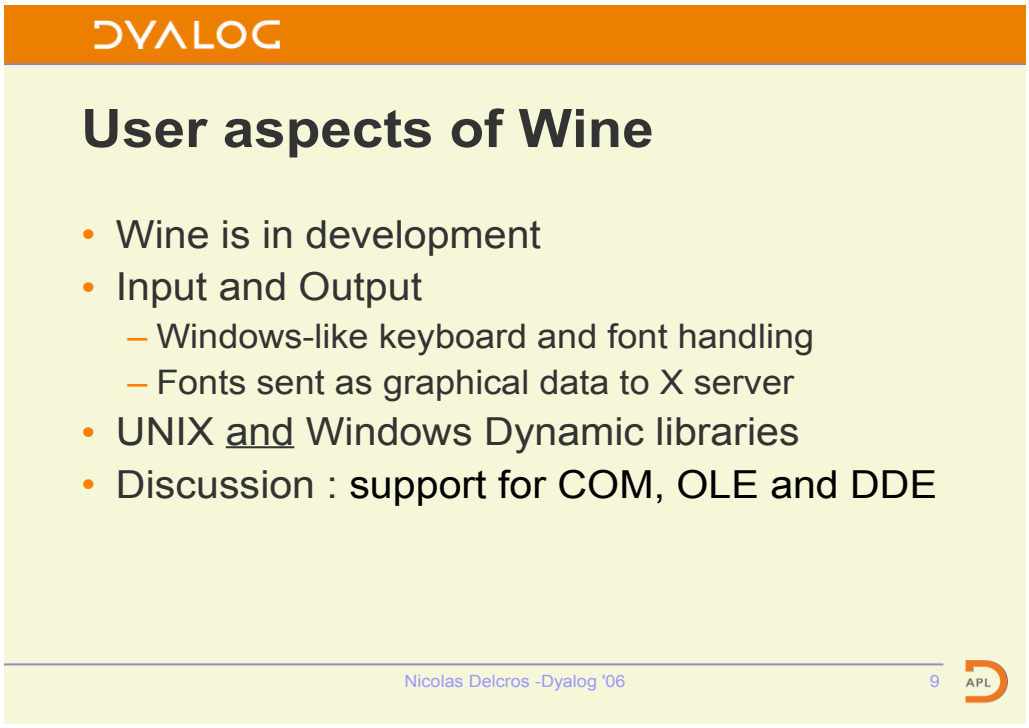
There is no fundamental difference between Wine and MainWin, apart from the Mono question, which is still open.



MainWin appears as the professional tool, and Wine as the Amateur Tool.
But both are powerful complementary tools.
(don't forget that the Titanic was built by professionals, and Noah's Ark by an amateur)

So at least we are going to support both ports for Dyalog, each one having strong advantages.

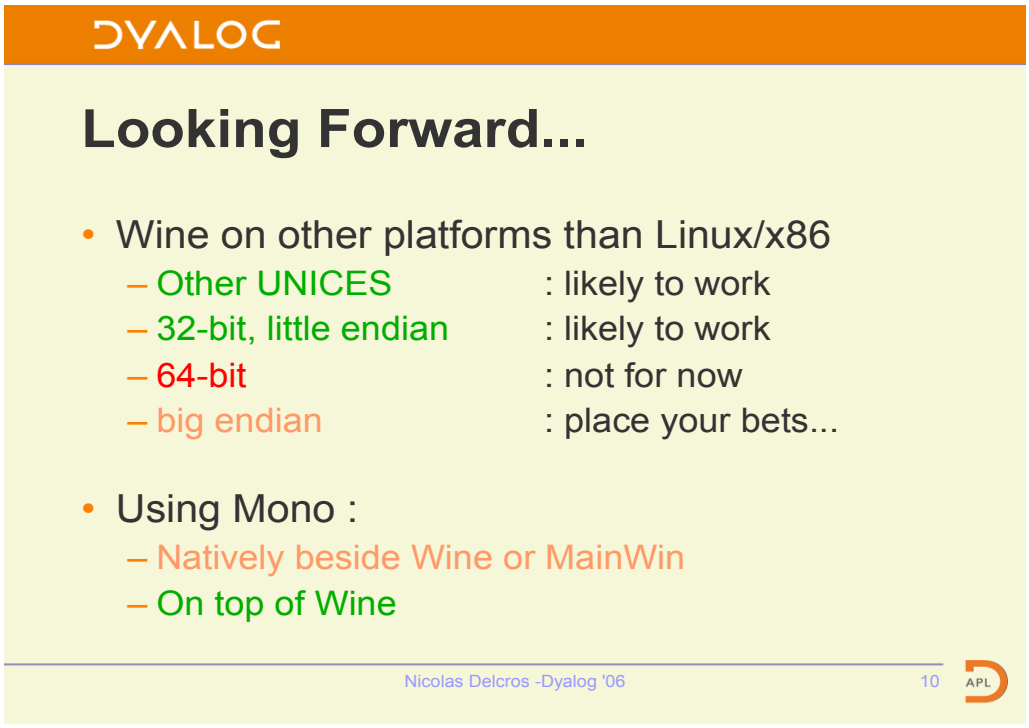
On word about Support Teams :
MainSoft will just have MainWin running,
where CodeWeavers will have your application ported.



Wine in development =
- bugs
- debugging issues
- lesser performances

Wine is more a “Windowsy” behaviour implementation
MainWin is more about building a POSIX application conforming to Windows requirements.

DDE : I had Shared Variables working between two Dyalog/Wine Sessions
OLE : Wine is supposed to support it
COM : never heard of COM support by Wine



Conclusion :Work In Progress

- Session = 90% of the app running, only interface issues.

- Wine port has now stepped from experiment to a serious port solution

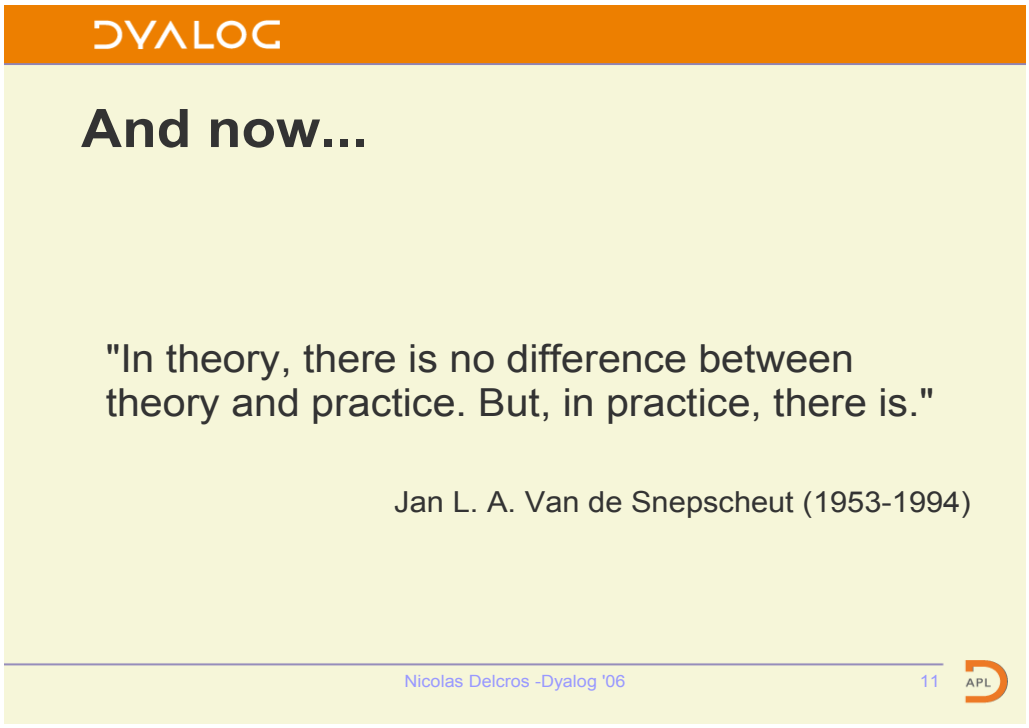
- Wine is suitable for non-professional license : learning and discovering Dyalog at no cost.

- But for professional systems (i.e. Reliability on every Dyalog component), we need more work to get a precise idea.

- A last word about Mono, based on a quick Googling :

Mono seems to have significant lacks of implementation when run natively on POSIX, and seems to run much better on top of Wine (where it provides Win32 Components)

This issue will be further investigated in a near future.



What we have running so far : Session, keyboard, fonts, ws load/save, language core (includes classes and latest v11 enhancements), GUI objects, TCP Sockets, Visual enhancements (Auto-completion, Tips, Syntax colouring...)

What we do not yet have : QuadNA, Consistent FileSystem, Files I/O (APL and native), guaranteed reliability/stability.

(don't forget this project is only 4 weeks old !)