Less is more. Why Oberon beats mainstream in complex applications

Fyodor Tkachov

Institute for Nuclear Research, Russian Academy of Sciences

Best ν mass bound from Troitsk-ν-mass (arXiv:1108.5034).

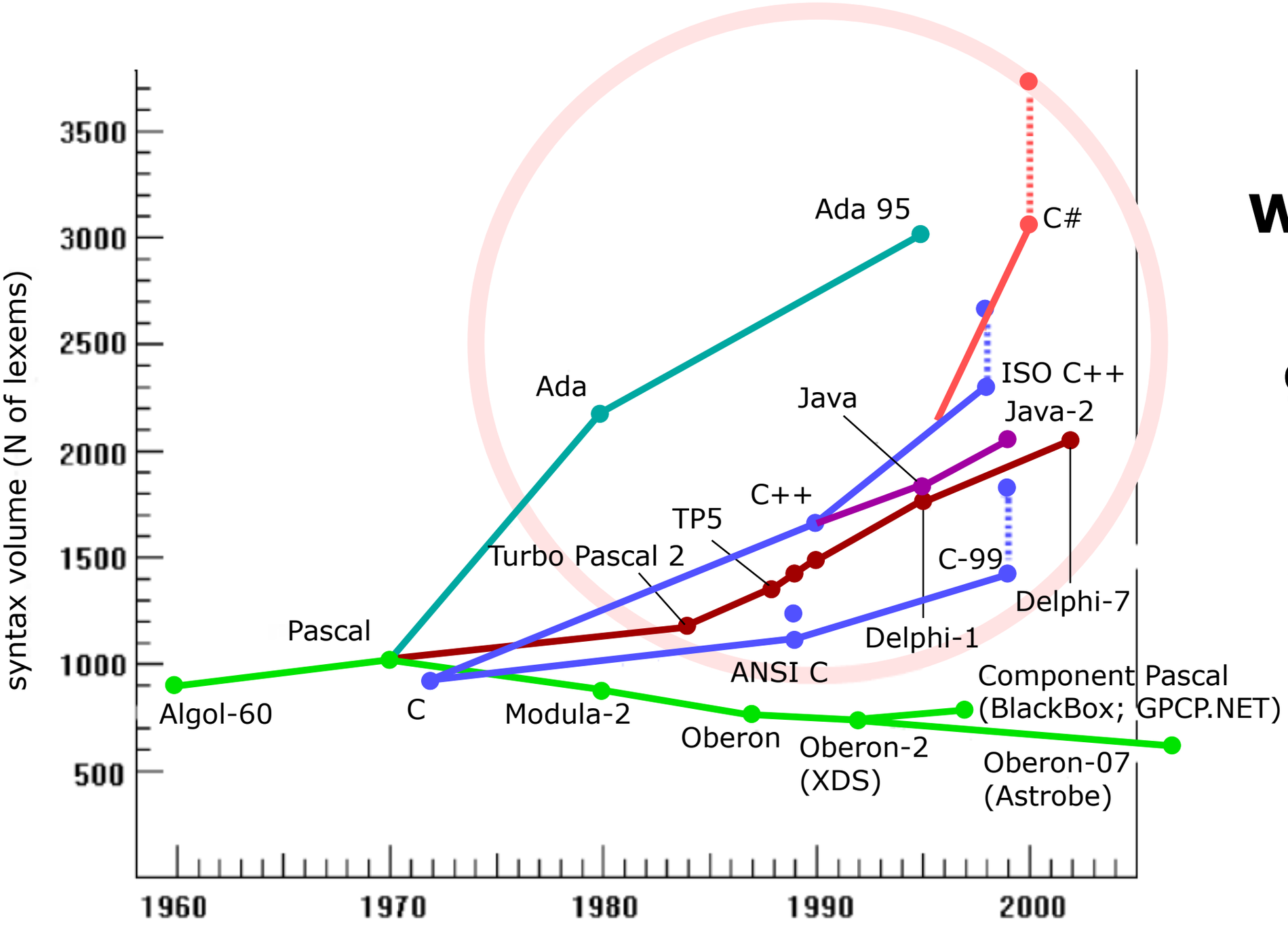
Had been plagued by "Troitsk anomaly" for ~10 years.

Anomaly went away after a reanalysis that was made possible by **two enabling technologies**:

Part 1 (prev talk) statistics: quasi-optimal weights

Part 2 (this talk) software: Oberon technologies

Application/"amateur"/non-professional programmer's perspective



Wirth’s Law:

Software gets slower faster than hardware gets faster.

IT-industry’s bubble of **excessive complexity**
VS **the rational core (Oberon)**

Original graph in Russian (c) S.Z.Sverdlov “Programming languages and translation methods”, Piter Press, 2007

Oberon ~ Pascal 2K

beware of myths and wrong associations

history

1970 Pascal
1980 Pascal-80=Modula-2
1986 Pascal-86=Oberon

dialects

– (classical | ETH) Oberon (1986)
– Oberon-2 (1992; XDS)
– Component Pascal (1997; BlackBox, .NET)
– Oberon-07 (2007; Astrobe)

Why Oberon "**technologies**"?

rather than simply "programming language" Oberon?

Never just tools,

but tools **+ techniques**

proper balance is key

If you are not aware of your techniques, then it is chiropractic.

To get the most from Oberon the language:
a development environment + a set of skills
(see below)

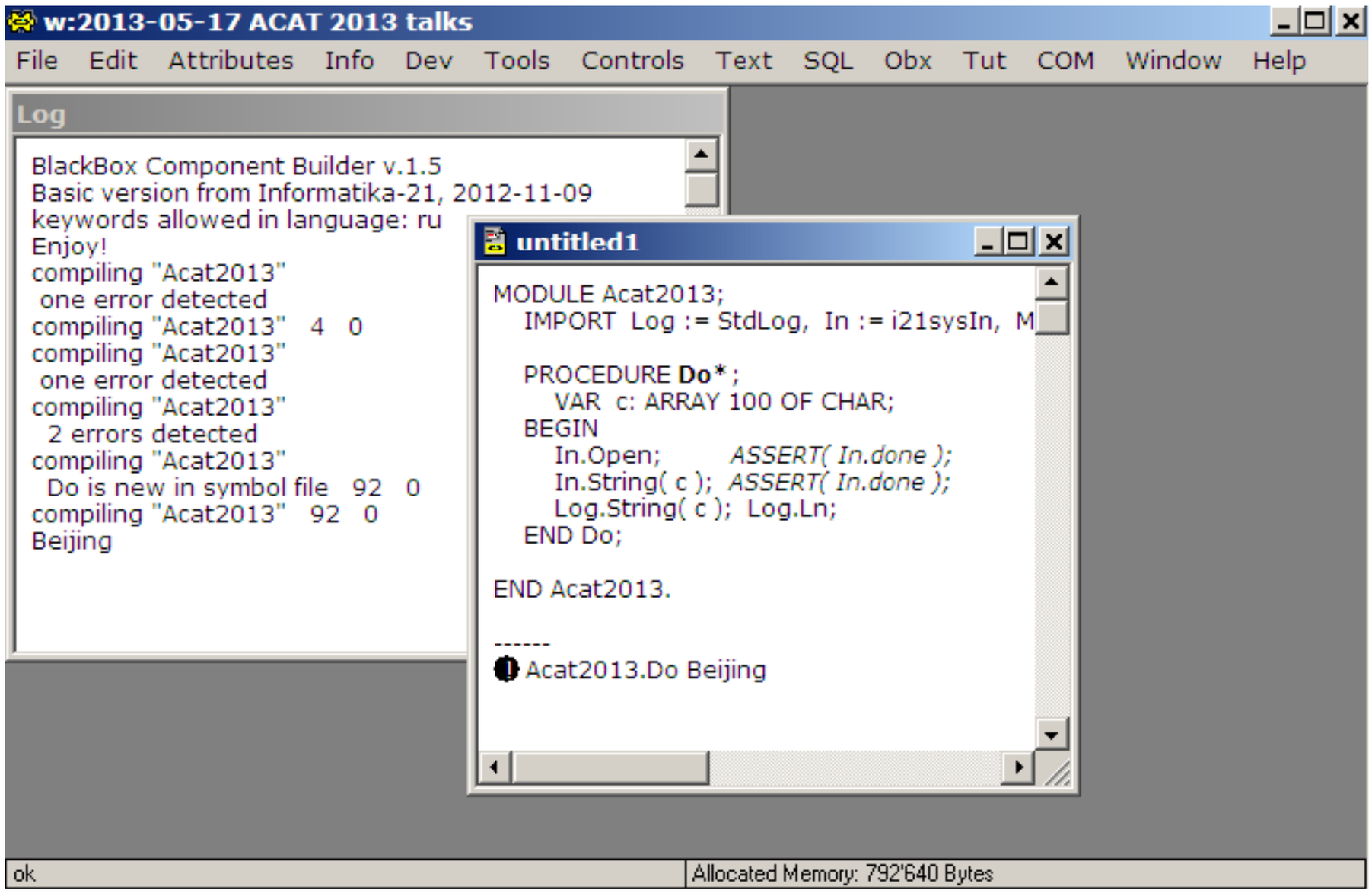
What *is* Oberon/Component Pascal

- A “vanishing” imperative programming language
(anything that can be put away into libraries is excluded from language;
what remains is designed with utmost care)
- Very small (language report ~20; dialects differ +/- a few pages)
- Pure compiled code (floating point optimization as external tool)
- Highly readable, robust (against typos etc.)
- Statically type safe (including dynamic records > no segviols, **ever**)
- Independently compilable modules (unit of information hiding, dynamic linking and (un)loading > “interactive” feel)
- Object-oriented (extensible records, very efficient)
- Garbage-collected (without affecting purely procedural programs)
- Object-oriented (extensible records)

Development environment, typically:

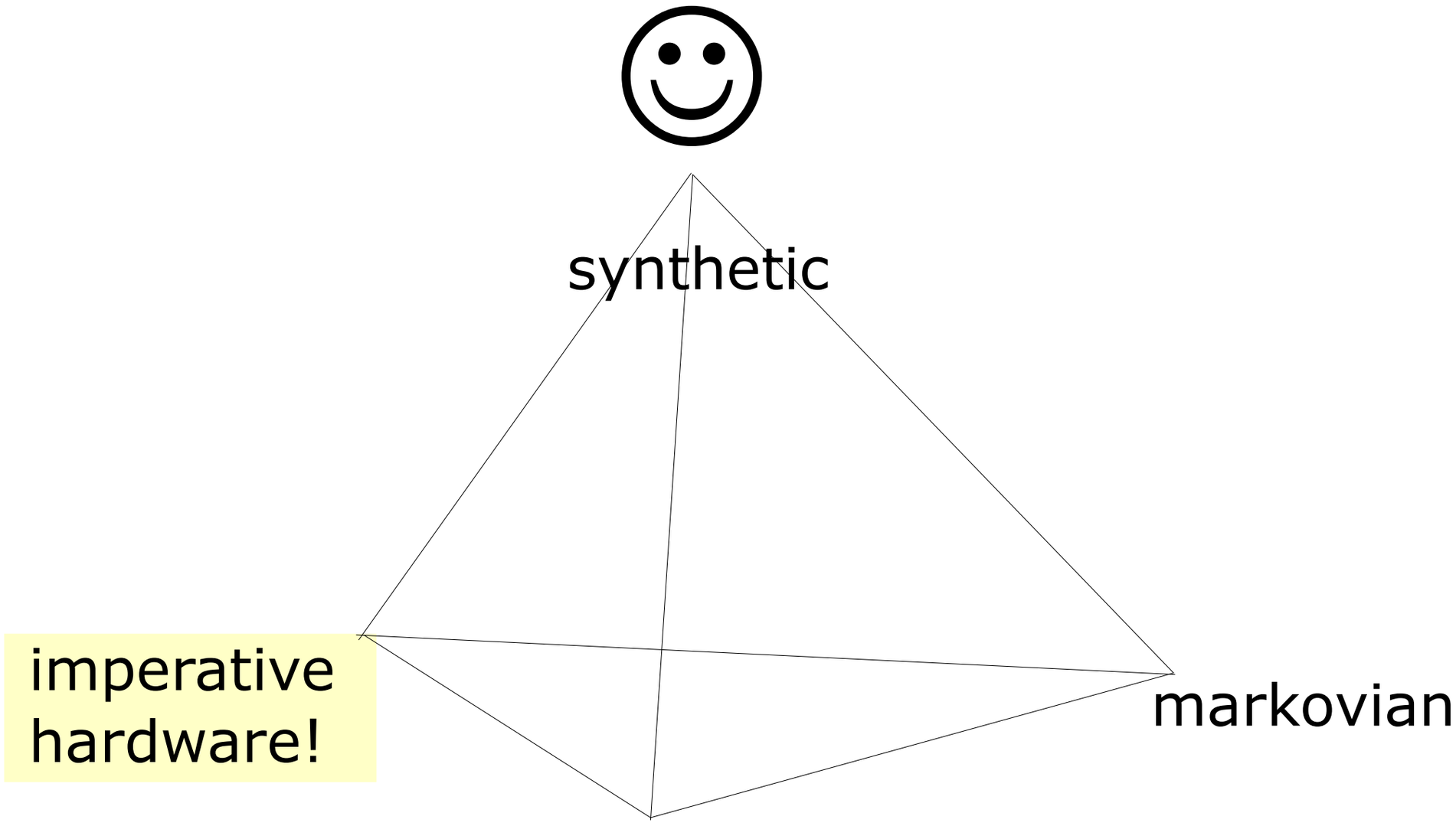
- Text-as-interface:
any text document can serve as a command prompt
+ input from any text
= hugely handy: text docs as flexible menus + storage for parameters ...

A simplistic interface, one becomes fully productive in a week:



92 in the log window means, the compiled code for this module is 92 bytes -- not kilobytes, just bytes.

Why imperative



farther from hardware # closer to human

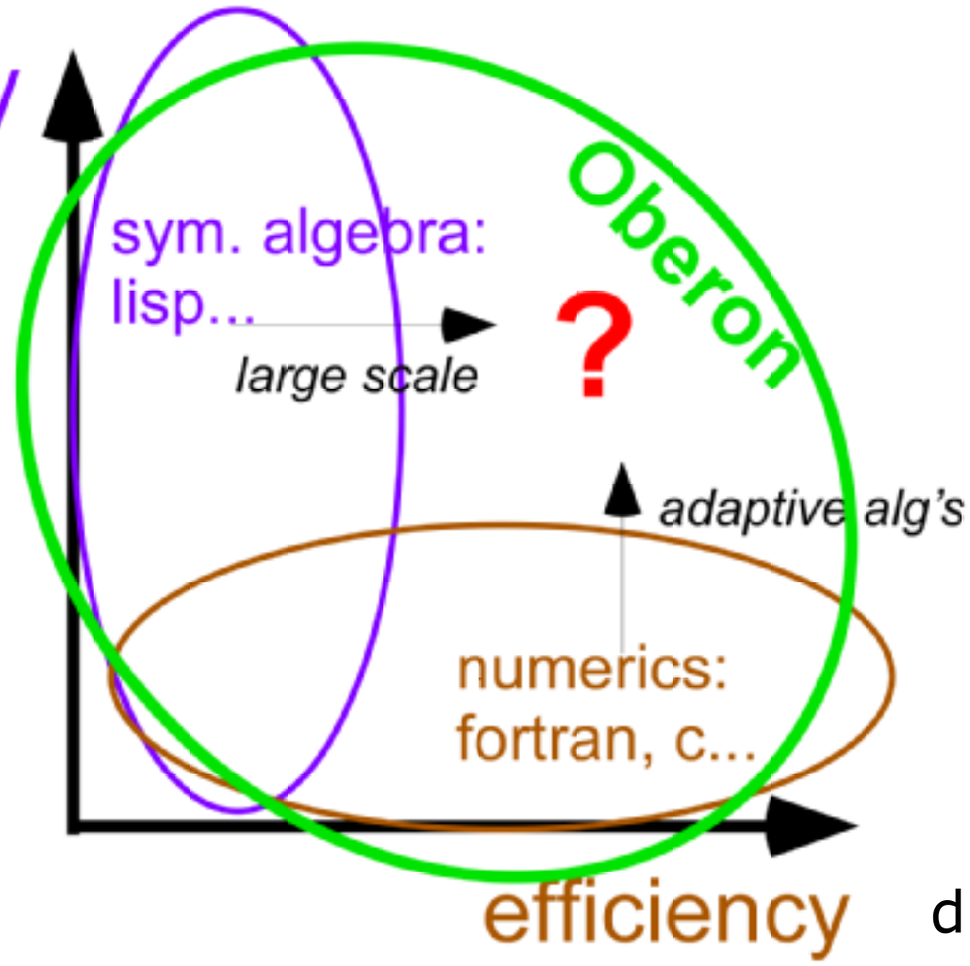
One Oberon does what is usually achieved via a combination of

C++ & python

Mathematica & Fortran

etc.

due to garbage collection



due to clean compiled code

non-professional's ego is not attached to IT toys

Oberon experience behind the assertions

BEAR algebra engine since 1997 *single-handedly*

one of the fastest engines, *the* most flexible

array of cutting edge calculations, A.Czarnecki et al.

hep-ph/0511004 Phys. Rev. Lett. (2006)
hep-ph/0506055 Phys. Rev. D
hep-ph/0503039 Phys. Rev. D
hep-ph/0403221 Phys. Rev. Lett. (2004)

Component framework implementing *single-handedly*

quasi-optimal weights (10K l.o.c. with all libraries)

used to reanalyse Troitsk-nu-mass data
arXiv:1108.5034 -- best direct neutrino mass bound

Continuous algorithm development work *single-handedly*

(Optimal Jet Finder etc. hep-ph/0301185; physics/0401012)

International educational project “Informatika-21”

coordinates leading experts from academia, aerospace, publishers ...
authorized revision of int’l bestseller “Algorithms and data structures”
by Turing Award winner N.Wirth www.inr.ac.ru/~info21/

Two causes of the excessive IT-complexity bubble:

1. **Combinatorial nature** of normal human intellect.

   primatologist W.Koeler 1919; zoopsychologists; cognitive sciences

   see a banana; scan the scene; identify familiar objects;
   find banana-getting combination of actions;

   if none, get angry and run around, this brings new
   objects into the scope of attention, with luck ... etc.

   Combinatory intellect (99.9% of all human activity**)**

   abstraction:
   magic wand

   cf. the belief that adding features to PL = "progress"

2. **Economic + social rent**
   derived by IT "experts" from the complexity.

   Well-known concept of **asymmetry of information**
   (e.g. George Akerlof, Nobel Prize for Economics, 2001
   for his analysis of "the market for lemons").

   Not a plot by IT, but a "collective effect" in the absence of
   proper education system:
   "normalized" group opinion/myth (see social psychology)

Alchemy came before rational Chemistry.

Astrology came before rational Astronomy.

IT is no exception.

We are currently at a parascientific stage of IT.

Choosing C++
was a gross failure of HEP community
as a *scientific* community.

C++ is best proof of natural origin of human intellect.
The choice of C++ is best proof of irrational forces within scientific community: **combinatorial intellect rules**.

The deal:

C++ (1K pp)

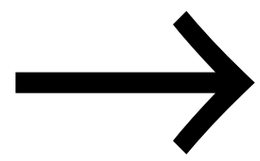
Fortran

Java

Form

Mathematica

.....



Oberon (20 pp)

General algorithmics

Dijkstra’s loop & invariant

Architecture patterns

-- Carrier-Rider

-- separation of interface from implementation

-- Oberon message bus

...

After one has learnt to program with Oberon, learning another language = learning its defects.

Oberon and open source

The world is much more complex than is imagined by software tool (libraries etc.) writers.
No library writer can foresee all the uses and applications.
Access to source for adaptation is essential.

Most open is the code that is most accessible.

Simpler language >> more open the source.

Oberon code is more open than e.g. C++ code.

The ill-recognized **problem of software complexity**
(cf. C++/Root/Mathematica ... crashes)

For the first time in the history of Humanity
the combinatorial/primatic intellect
has become freed from the restrictions of
the resistance of materials.

The Kalashnikov Principle:

Excessive complexity = vulnerability

Asymmetry of information >> customers pay.

Containing the gratuitous growth of complexity
must be a permanent concern
whenever IT is involved.

Oberon influence in the IT industry:

Java emerged after Sun licensed and studied Oberon compiler in 1991; the influence is obvious.

Google's **Go** is a C-syntax clone of Oberon with minor (unnecessary) extensions.

Wirth's student Clemens Szyperski is author of **best selling "Component Software"** and software architect at MS Research working on .NET.

(The books essentially describes the principles that a popular Oberon implementation the BlackBox Component Builder is built on.)

Only physicists are in the dark.

References

Prof. Jurg Gutknecht’s group at ETH Zurich: http://nativesystems.inf.ethz.ch/

Oberon Day @ CERN 2004: http://www.inr.ac.ru/~blackbox/Oberon.Day/

http://arxiv.org/abs/hep-ph/0202033 (see testimonies at end)

BlackBox Component Builder: www.oberon.ch, www.zinnamturm.eu

XDS Oberon (optimizing) http://www.excelsior-usa.com/xdsx86.html

Oberon-07 for embedded apps: http://www.astrobe.com/

Gardens Point Component Pascal: http://gpcp.codeplex.com/

Informatika-21 (educational) http://www.inr.ac.ru/~info21/

N.Wirth “Algorithms and data structures. Version for Oberon”
http://www.inf.ethz.ch/personal/wirth/books/AlgorithmE1/AD2012.pdf

Clemens Szyperski “Component Software: Beyond Object-Oriented Programming”
2nd ed., Addison-Wesley, 2011