HONG KONG BAPTIST UNIVERSITY
FACULTY OF SCIENCE

Department of Computer Science Colloquium
2016 Series

Oberon = Education + Innovation

Dr. Fyodor Tkachov
Leading researcher,
Russian Academy of Sciences

Date: November 30, 2016 (Wednesday)
Time: 2:30 - 3:30 pm
Venue: SCT909, Cha Chi Ming Science Tower, Ho Sin Hang Campus

Abstract

The Project Oberon by Niklaus Wirth (Turing Award, 1984) and Jurg Gutknecht (ETH, Zurich) summarizes the essence of programming in a minimalistic language and IDE that are in a stunning contrast with mainstream tools in regard of size, robustness, flexibility and smooth workflow.

Oberon has been influencing the IT industry (Java, C#, Golang ...) and proves to be a perfect backbone for systematic programming education encompassing students of ages 12 (introduction) to 20+ (software architecture, compiler construction etc.).

It also proves to be a great tool (even a silver bullet in a number of cases) for various projects that are cost sensitive OR require a stringent quality of software OR are immune to the pressures of IT mainstream OR cannot push the costs of the mainstream complexity on to unsavvy customers, as in application areas such as:

drones, nuclear power stations, various industrial automation, theoretical and experimental scientific research, teaching programming to 12-year old kids ...

The talk is based on the diverse experiences accumulated by the participants of the educational project Informatika-21 since the late 90s and continues a series of domestic and international presentations, including the following ones:

Austria, 2003: http://www.inr.ac.ru/~info21/texts/2003-08-JMLC/en.htm
Zurich, 2011: https://www.ethz.ch/content/vp/en/conferences/2011/oberon.html
Zurich, 2013: http://ictk.ch/content/informatik-und-die-schule-des-21-jahrhunderts
Beijing, 2013: http://indico.ihep.ac.cn/event/2813/session/3/contribution/38/material/slides/2.pdf

The fresh (Nov. 30) issue of The Standard provides a perfect way to start this talk:

New air system blanks out for 26 seconds
http://www.thestandard.com.hk/section-news.php?id=176914

Editorial: Maximum control needed in the air
http://www.thestandard.com.hk/section-news.php?id=176902

The new and expensive air traffic control system at the Hong Kong airport blanked out dangerously for 26 seconds, following a series of other incidents since it became fully operational.

Quite obviously, the problem is due to buggy software.

Efficient construction of high-quality software is the problem addressed by Oberon.

Compare: a monitoring system at the Rostov-on-Don nuclear power station, written in C++/CORBA, defied a complete debugging for a number of years;

switching to Oberon eliminated all the problems and resulted in a measurably higher-quality system.

http://www.inr.ac.ru/~info21/oberon_innovation/oberonRostovAES.htm

This talk is a privilege for two reasons:

One -- Hong Kong is #1 venue of a cross-cultural exchange

Two -- the Chinese educational ethics

Thanks to Professor Yuen for extending this privilege to me
.. at a symposium in Zurich, Switzerland, in 2014. The symposium to honor Niklaus Wirth on the occasion of his 80th birthday.

Niklaus Wirth is arguably the best programming language designer ever,
awarded the Turing Prize for his Pascal and Modula-2 in 1984,
but managed to leave them far behind with **Oberon** (1988) which is the subject of this talk:

there must be a few thousand IT departments in the universities around the world,
but I did not notice *thousands* of IT department heads at that symposium:

This photo illustrates nicely the two points I made in the beginning.

Oberon

1 programming language -- only a starting point of discussion

2 IDE(xecution)E(nvironement) ("workflow")
on top of another OS or as a native OS

3 ideology
a "theory" of programming based on:
complexity = the primary concern
"**less is more**"

what's so special about it (small, clean, safe, fast compiler)

what went into its design

experience of an engineer
25 years of language design,
30 years of compiler construction
N.Wirth. Compiler Construction, Addison-Wesley, 1996

how it helps with education

who are the "non-professional" programmers

how programming can be taught with Oberon efficiently and in a systematic fashing to 12-20 year old students

how it promotes innovation

allows to handle the niches where the mainstream tools fail, completely, due to excessive complexity >> **silver bullet**

Project & site: **Informatika-21** (since 2002)
promotes Oberon for education >>> site

http://www.inr.ac.ru/~info21/
proposal: an efficient backbone system
of introductory programming and basic CS courses
based on a single platform
encompassing students of ages 12-20

mainstream languages are then to be taught with a focus on specific features

DIVIDE AND CONTROL:
separate fundamentals from language specifics (=defects)

it is simply more efficient (proven by experience)

Who are "non-professional" programmers

those who do a lot of application programming but do not sell programs, so they
cannot benefit from the asymmetry of market information generated by the excessive complexity of IT tools.

Closely related: a fraction of system-level IT professionals who cannot benefit ... etc. (including some -- not all -- embedded systems programmers).

Concept formulated in **2002** in the very first edition of the Informatika-21 site.
Oberon was indicated as a preferred tool.

Microsoft reacted within 23 hours after the site had been announced,
then started a campaign promoting C# for "non-professionals".
2005: Express edition of their tools, targeting "amateur" programmers ("amateur" is a wrong word >> wrong implementation).

That MS campaign fizzled out -- Informatika-21 lives.

Now MS devises application domain languages targeting specific niches of business applications.
They keep attempting to exploit the niche of "non-professionals" without loosing hold on the market.

The speaker is a "non-professional" ... but no "amateur".

>>>>> the Hello, Hong Kong! program with keywords in Chinese ⇨

```
模組 x;
    连接  StdLog;
开端
    StdLog.String("Hello, Hong Kong!")
结束 x.
```

English equivalent:

```
MODULE x;
    IMPORT  StdLog;
BEGIN
    StdLog.String("Hello, Hong Kong!")
END x.
```

an obvious similarity to Pascal and Modula-2

this dialect of Oberon: "Component Pascal"

the behind the scene translation is made possible by:

-- very small and very regular language
-- lightning-fast compilation
-- "text as interface" in place of command line
-- Oberon is a preprocessor for itself
>> program Self

Implemented by a "non-professional" programmer!
if only 1 out of 100 teachers were able to do such things ... ⇦

Grammar of Oberon

⇨

Appendix B: Syntax of Component Pascal, popular dialect of Oberon

Module = **MODULE** ident ";" [ImportList] DeclSeq
[BEGIN StatementSeq]
[CLOSE StatementSeq] END ident ".".

ImportList = **IMPORT** [ident ":="] ident {"," [ident ":="] ident} ";".

DeclSeq = { **CONST** {ConstDecl ";" } | **TYPE** {TypeDecl ";"} |
VAR {VarDecl ";"}} {ProcDecl ";" | ForwardDecl ";"}.

ConstDecl = IdentDef "=" ConstExpr.

TypeDecl = IdentDef "=" Type.

VarDecl = IdentList ":" Type.

ProcDecl = **PROCEDURE** [Receiver] IdentDef [FormalPars] MethAttributes
[";" DeclSeq [BEGIN StatementSeq] END ident].

MethAttributes = ["," NEW] ["," (ABSTRACT | EMPTY | EXTENSIBLE)].

ForwardDecl = PROCEDURE " ^ " [Receiver] IdentDef [FormalPars] MethAttributes.

FormalPars = "(" [FPSection {";" FPSection}] ")" [":" Type].

FPSection = **[VAR | IN | OUT]** ident {"," ident} ":" Type.

Receiver = "(" [VAR | IN] ident ":" ident ")".

Type = Qualident
| **ARRAY** [ConstExpr {"," ConstExpr}] OF Type
| [ABSTRACT | EXTENSIBLE | LIMITED]

```
                    RECORD ["("Qualident")"] FieldList {";" FieldList} END
                  | POINTER TO Type
                  | PROCEDURE [FormalPars].
FieldList       = [IdentList ":" Type].
StatementSeq    = Statement {";" Statement}.
Statement       = [ Designator ":=" Expr
                  | Designator ["(" [ExprList] ")"]
                  | IF Expr THEN StatementSeq
                    {ELSIF Expr THEN StatementSeq}
                    [ELSE StatementSeq] END
                  | CASE Expr OF Case {"|" Case}
                    [ELSE StatementSeq] END
                  | WHILE Expr DO StatementSeq END
                  | REPEAT StatementSeq UNTIL Expr
                  | FOR ident ":=" Expr TO Expr [BY ConstExpr]
                    DO StatementSeq END
                  | LOOP StatementSeq END
                  | WITH [ Guard DO StatementSeq ]
                    {"|" [ Guard DO StatementSeq ] }
                    [ELSE StatementSeq] END
                  | EXIT
                  | RETURN [Expr]
                  ].
Case            = [CaseLabels {"," CaseLabels} ":" StatementSeq].
```

```
CaseLabels  = ConstExpr [".." ConstExpr].
Guard       = Qualident ":" Qualident.
ConstExpr   = Expr.
Expr        = SimpleExpr [Relation SimpleExpr].
SimpleExpr  = ["+" | "-"] Term {AddOp Term}.
Term        = Factor {MulOp Factor}.
Factor      = Designator | number | character | string | NIL | Set |
              "(" Expr ")" | " ~ " Factor.
Set         = "{" [Element {"," Element}] "}".
Element     = Expr [".." Expr].
Relation    = "=" | "#" | "<" | "<=" | ">" | ">=" | IN | IS.
AddOp       = "+" | "-" | OR.
MulOp       = " * " | "/" | DIV | MOD | "&".
Designator  = Qualident {"." ident | "[" ExprList "]" | " ^ " | "(" Qualident ")"
              | "(" [ExprList] ")"} [ "$" ].
ExprList    = Expr {"," Expr}.
IdentList   = IdentDef {"," IdentDef}.
Qualident   = [ident "."] ident.
IdentDef    = ident [" * " | "-"].
```

⇦

This is the **complete** grammar.
Very few "interesting" features
but every detail polished (cf. OR, &)

Oberon is a thinnest layer that:

converts machine language into one allowing logical reasoning

+ provides maximal protection from human mistakes
(strict static type safety **extended to pointers >> garbage collection**)

+ provides a basic set of tools to grow programs arbitrarily (modules + type extension).

The basic single-inheritance type extension is treated as a way to divide functionality between independent modules (polymorphism etc. is then a consequence).
Multiple inheritance is left for application programmers -- same as dynamical data structures.

Why complex "interesting" constructs are avoided in Oberon:

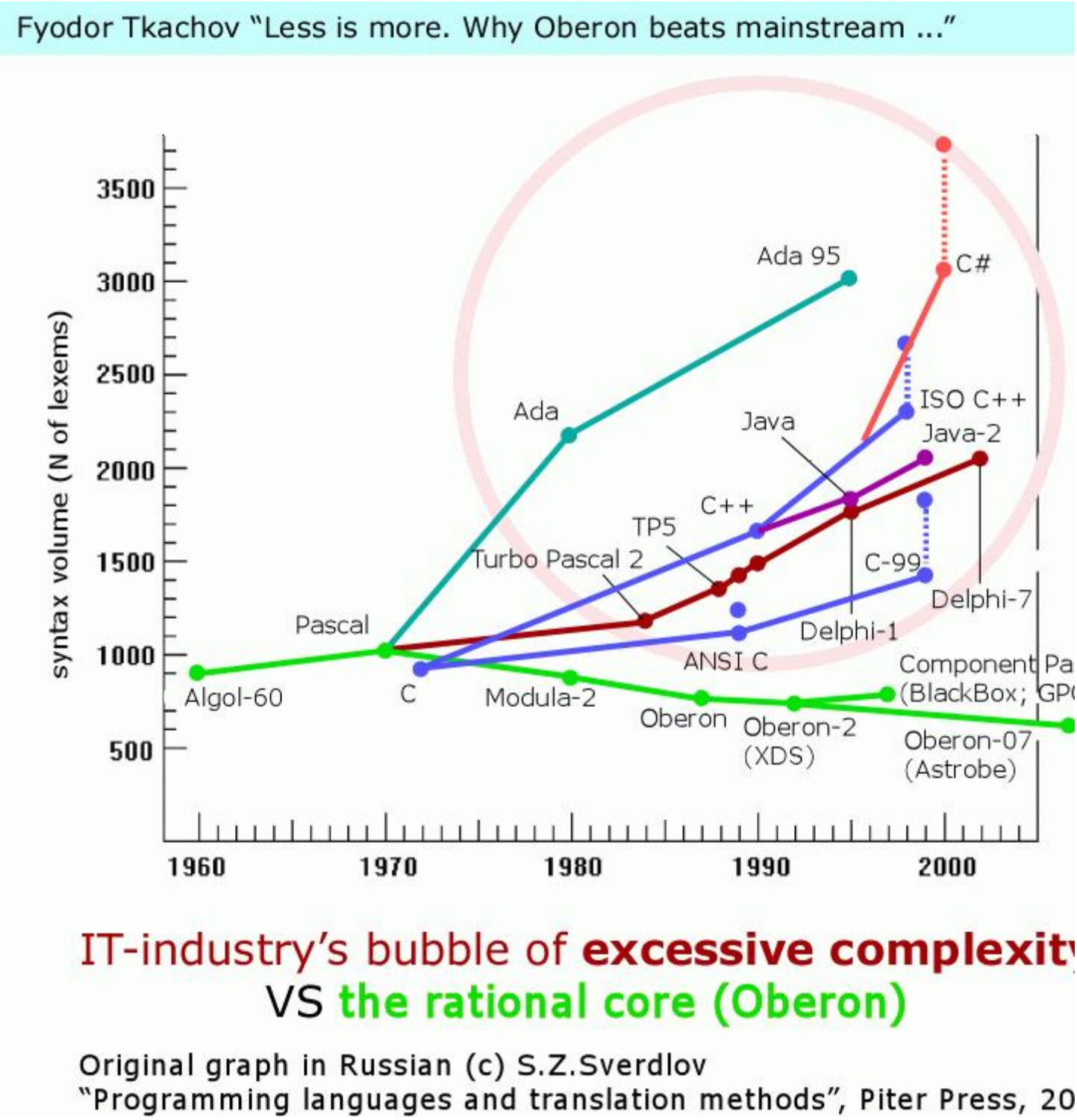
Complex constructs have many variations, depending on specific applications.

The more complex, the less are the chances that the choice of the language designer would be optimal in specific applications.

Already complex numbers proved problematic:
  two possible representations;
  real and imaginary parts often computed separately -- then what's the point of having the built-in type?

Multiple inheritance is treated in the spirit of complex dynamical data structures:
the language provides tools to construct any such thing efficiently -- and leaves all decisions on details of a specific implementation to programmer's discretion.

Picture #1 ⇨



the graph is not intended for comparison of languages in regard of their complexity -- only **Oberon VS the rest** of the world

the IT bubble of excessive complexity -- (almost) everywhere

this is a fundamental phenomenon, to be understood

cognitive roots -- the way humans solve problems ("combinatorial intellect" -- the first level of intellect)

philosophers and psychologists did not recognize it because they don't know what combinatorics is

economic roots:

creates and feeds on, asymmetry of information

Nobel for economics, Akerlof, Spence, Stieglitz (2001);

theorem:

with AoI, **the worst wins** in the market competition

Combinatorial intellect -- the basic/foundational form of intellect ⇨

CI turns on when a goal is seen but is not immediately reachable
scan the scene, identify (familiar) objects
recall possible actions
find a combination of actions with objects to reach goal

humans rarely go beyond CI⇦

the next level of intellect: **reflection + critical thinking** ⇨

every element of CI is subject to reflexive critique:
why that goal; what are those object; why those actions

means going back towards the basics/elements/**foundations**
convenient musical term:
нем. **Krebsgang** = cancer walk
лат. cancrizans, per motum retrogradum;
en. retrograde = **stepping backwards**

may develop slowly, with age -- or not at all
requires (self-) training⇦

combinatorial intellect: combine what's available (if unsuccessful, search for new things to add to combinations)

= accumulation -- patch upon patch, prop for prop, etc.

critical intellect: one decomposes what's available into elements and examines relevance, etc.

= simplification -- eliminating what's irrelevant, etc.

Two meanings of "simplicity" ⇨

simple-minded -- simplicity of an ignoramus

sophisticated simplicity of a sage -- NO WORD FOR IT⇦

perhaps 哲朴 ?

Excessive complexity = natural result of unbridled combinatorial intellect at work

just look at kids' programs: it does not occur to them to go over their programs again, in order to clean up the mess after trial-and-error -- one must teach them to do so

G3 : three big principles related to complexity ⇨

1 The Kalashnikov Principle:

Excessive complexity = vulnerability

explains **why** excessive complexity is bad (vulnerability may be client's due to the economic asymmetry of information)

2 DIVIDE AND CONTROL

the single most important recipe to control complex systems

explains **how** to hold complexity in check
>> non-obvious examples in Hello ... program (all-caps keywords ...)

#1 example: automatic garbage collection
the mapping of data structures to static memory
is postponed until the program works correctly

By default "divide".
One must explain why no "divide", not vice versa.

Example: keywords in all-caps are visually "divided" from identifiers and unconditionally recognized as keywords by the compiler.

3 The Pareto Principle: **80/20**

non-uniformity of real-world distributions

explains **how** to determine where to divide (optimize only part of code etc.)⇦

Why the Oberon message may be misinterpreted by mainstream

Tools and techniques go hand in hand

mainstream programmers project onto Oberon their experiences with hugely complex tools

For instance, when programming in Oberon:

sintax coloring is superfluous >> color can be used much more usefully

very frequent compilation catches typos early

heavy reliance on static type safety >> much facilitates the polishing of code

highly interactive (similar to lisp, smalltalk etc.) >> text as interface enormously useful for application programmers (data analysis etc.)

no stepwise debugger -- a symbolic, human-readable stack dump + discipline of programming

>> Edsger W. Dijkstra, Turing 1972
a classic "A Discipline of Programming"

obligatory: D.Gries "The Science of Programming"

Traditions that influenced Oberon ⇨

"Swiss quality"

Swiss watches, knives, (formerly machinery), food (Nestle),
pharmaceutical industry (Novartis et al. -- 30% of Swiss exports),
etc.

Jurg Gutknecht (ETH, Zurich, retired; now in South Africa)
began as programmer in aviation industry
realized importance of math >> obtained PhD in mathematics
~1980 joined N.Wirth
made significant contributions to Project Oberon (message bus etc.)

Niklaus Wirth

as a teenager, was fascinated by model airplanes;
electronics unreliable >> entered electronics department
got interested in computers and programming languages and compilers:
euler, algol-w,
1969: presented **Pascal** -- after breaking with the Algol-68 committee
1975/66: sabbatical at Xerox PARC (Palo Alto

Research Center) -- **Xerox Alto** system

1980: refined Pascal to **Modula-2** (very good for embedded systems, used in Russian communication satellites)

Lilith workstation

1985/86: sabbatical with Gutknecht at Xerox PARC -- experimental computer system **Mesa**

1988: programming language Oberon

1990: **Oberon System** on **Ceres** workstation

Oberon System is fully described in:

N.Wirth and J.Gutknecht "Project Oberon" (1992, 2013)
http://www.projectoberon.com/

Oberon dialects ⇨

ETH Oberon ("classical") -- within an OS that ran natively; available now under Windows.

Oberon-2 = ETH Oberon + methods (not thought out well enough)

XDS Oberon-2 (optimizing compiler from best experts in Novosibirsk) -- available freely (but no code)

Component Pascal (1997) -- fine-tunes Oberon-2

BlackBox Component Builder -- IDEE under MS Windows and GNU/linux+wine

authors: Oberon microsystems, Inc.; last release 1.5 (free and very open -- BSD licence)

version 1.7 of BBCB fully supports **Unicode** identifiers

first fruit of international collaboration

>> BlackBox Center

http://blackboxframework.org/index.php?cID=home,en-us

Chinese participation (Luo WengYing)

the most popular and practical dialect due to interoperability with MS Windows (however, this creates problems too)

GPCP (Gardens Point Component Pascal) under .NET

Oberon-07/11/13 (NW keeps polishing his best invention)

popular with embedded systems (ARM etc.):

Astrobe (Australia) -- a commercial implementation

an opensource **AVS** cross-compiler under BlackBox

Active Oberon, Zonnon, Bluebottle -- research projects at ETH, Zurich, building upon Oberon

a bunch of compilers, e.g. **Oxford Oberon-2** compiler

⇦

Oberon influence ⇨

Java

Sun team studied Oberon compiler in 1991

Java bytecode is based on Pascal P-code (ancient technology);

Java rollout was hurried up by a presentation at Sun by NW's student Michael Frantz -- "slim binaries"

restrictions on multiple inheritance, type safety -- reminiscent of Oberon

C# (similar to Java)

funny: one cannot live without modules >> namespace etc.

factoid:

at the presentation of MS.NET in 2000, out of 12 compilers presented, only 2 actually worked (more or less): Oberon and Component Pascal

Golang >>> ⇨

http://www.ocp.inf.ethz.ch/forum/index.php?topic=3

New language from Google: "GO". Includes rant about Oberon. (1/1)
soren renner:
http://scienceblogs.com/goodmath/2009/11/googles_new_languag
o.php

comment #26:
Quote

You know, all this hoopla over "Go" and how wonderful it is
and yadda yadda yadda
was being pushed by the ETH Oberon group and all Oberon users back in the late 90s.
With few exceptions,
certainly none large enough to fundamentally change the language for,
Go does _NOTHING_ different than Oberon-2.

Why can't people just use Oberon and move on with life?
I'll tell you why -- because it was designed by Niklaus Wirth.
People HATE Wirth. He's the anti-Christ. He's an abomination.
Nobody loves him, and neither should you.
He's communist, fascist, and a capitalist pig all at the same time.
And, get this folks!, he writes his languages using all-caps keywords! OH NOES!!

Except, of course, the folks who want to cash in on Wirth's ideas while retaining C/C++'s basic, ugly, highly error-prone, and utterly retarded syntax.
I have no respect what-so-ever for **Pike and Thompson** here
-- they're **just plagiarizers** and have actually made the language WORSE.
And, unbelievably hard to read and maintain in comparison.

They should be ashamed of themselves.

I am so sick and tired of this uber-opportunist and politicized bull$#!+ in the comp-sci/comp-eng fields.
If it's one thing that pisses me off more than any other is
when awesome technology comes out,
gets trash-talked fifteen ways 'til Monday,
then some jackass who lists "Bell Labs" on his resume
comes out with the EXACT SAME LANGUAGE
but with a different syntax and a handful of bells and whistles
(which EASILY could have been retrofitted into Oberon-2 had anyone gave a damn about it),
and he somehow gets all cuddly kudos and awesome street-cred for it.

BULL$#!+.

Posted by: Sam
soren renner:
http://www.math.bas.bg/~bantchev/misc/on-go.html

Quote

And finally, the language **Oberon and N. Wirth should have been acknowledged by Go's authors much more explicitly** than with the single phrase
sage:
Yep, another piece of BULL$#!+ >:(

Navigation

[0] Message Index

Education

1992: the speaker learned about Oberon.
1995: downloaded BlackBox Oberon.
1997: realized it solves all my old SW problems (reliability, memory management).
1999: developed a computer algebra engine for my problems.
2001, Spring semester: tried to read a course to graduate students -- discovered they know ~nothing.
2001, Fall semester: started a course in a nearby lyceum.
2002, Fall: site Informatika-21.

The university course retargeted as an introductory programming.
Since 2008: Friday afternoons, extra-curriculum courses of programming for kids 12-15, in a nearby educational center.

A number of teachers have been using it (Belorussia, Kazahstan).
Some textbooks have been published.

It is completely realistic to take students from introductory programming to compiler construction, to S/W architecture within a single environement, without loosing tempo.

Impressive testimony last year from a former student: an advanced course for school-level students (recursion, pointers, lists...) with Oberon proved to be clearly superior to the university courses in C and C++.⇦

Innovation ⇨

examples below only from Informatika-21 collection, for more see:
2004: http://www.inr.ac.ru/~blackbox/Oberon.Day/
2011: http://www.oberonday2011.ethz.ch/talks/

Kaliningrad (westernmost Russian city)
electricity network of a township (population 35K)
examined practically all the available options -- BlackBox proved the only working option -- **silver bullet**
found BlackBox via Informatika-21
more:
http://www.inr.ac.ru/~info21/oberon_innovation/oberonKaliningrad.htm

Bryansk region
automation of a grain storage facility (part of Miratorg Corp.)
unusual: highly interactive
replaced cumbersome and expensive software from Siemens AG and OMRON Corp. (Japan) -- **silver bullet**
team: participants of Informatika-21 since 2005
more: http://www.inr.ac.ru/~info21/oberon_innovation/oberonAgro.htm

Rostov-on-Don nuclear power station
monitoring system

Oberons replaced "industry standard" C++/CORBA
solved the problems that defied solution for a number of years -- **silver bullet**
found BlackBox via Informatika-21
more: http://www.inr.ac.ru/~info21/oberon_innovation/oberonRostovAES.htm

UAV (drones)
cross-platform Oberon-07 compiler for embedded applications
runs under BlackBox
analogue: Astrobe
author: Alexander V. Shiryaev, Moscow, a former student of my university course
more: http://www.inr.ac.ru/~info21/oberon_innovation/oberonBPLA.htm

FT
a bunch of unique solutions for elementary particle physics
most recent: best upper bound on neutrino mass -- **silver bullet**

understanding how formulae are mapped to programs narrows down the options on the math level something wonderful
(my PhD work of 1981 generates new citations every week).
more:
http://indico.ihep.ac.cn/event/2813/session/3/contribution/38/material/slides/2.pdf
⇦

Thank you for your attention.

Godspeed with **Oberon**!

This presentation was prepared directly within BlackBox Oberon.