

# BlackBox

## An Industrial-Strength Oberon Implementation

Dr. Cuno Pfister

10-Mar-04

[www.oberon.ch](http://www.oberon.ch)

# Oberon microsystems, Inc.

- Founded in 1993 as an ETH Spin-Off
  - By a **physicist**, an electrical engineer and a computer scientist
- Goal:  
High-Quality Software for Industry
  - Interest from science is a positive surprise!

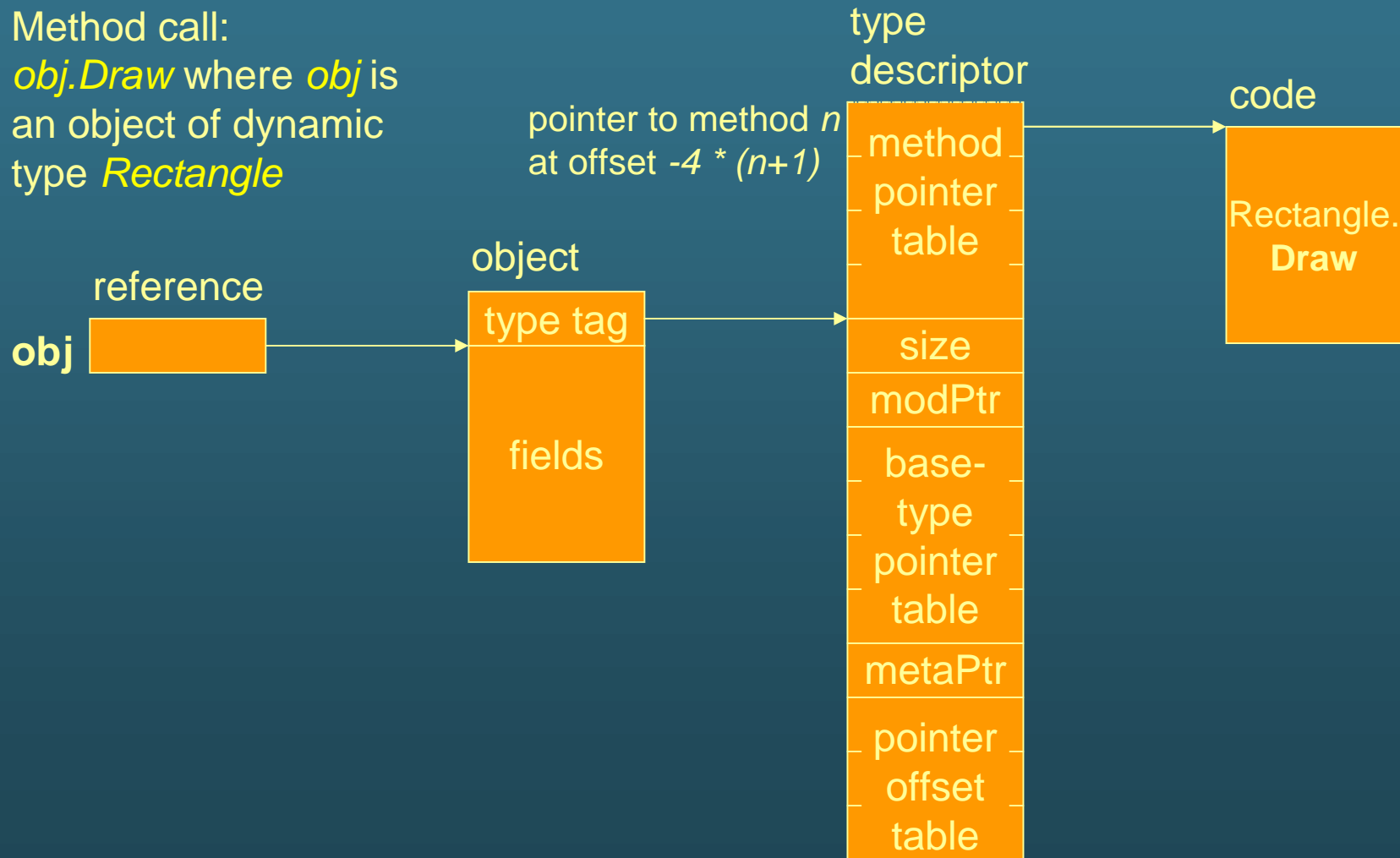
# BlackBox Component Builder

- Industrial-strength Oberon
  - Retains general Oberon strengths ...  
(speed, simplicity, safety, cross-platform capability, run-time loading of new components)
  - ...*and* hosted by **most popular OSes & GUIs**

# The Language

# Method Calls in Oberon-2

Method call:  
*obj.Draw* where *obj* is  
an object of dynamic  
type *Rectangle*



# Component Pascal Syntax

TYPE

GraphicsObject = POINTER TO ABSTRACT RECORD END;

Rectangle = POINTER TO RECORD (GraphicsObject)

... field declarations ...

END;

PROCEDURE (obj: Rectangle) Draw;

BEGIN

... implementation of class *Rectangle's Draw* method...

END Demo;

PROCEDURE Demo;

VAR obj: GraphicsObject; rect: Rectangle;

BEGIN

NEW(rect); (\* create an instance of class *Rectangle* \*)

obj := rect; (\* assign *rect* to generic graphics object *obj* \*)

**obj.Draw** (\* call *Draw* method \*)

END Demo;

(The “Component Pascal” name for Oberon-2 is used for marketing reasons)

# The Environment

BlackBox Component Builder

# Key Assets of BlackBox (1)

- Instantaneous edit / compile / run cycle
  - No switching between development & run-time environment necessary
- Scope:  
From small programs to large systems
  - **No compromises**, even the garbage collector is written in Component Pascal!



## Key Assets of BlackBox (2)

- **Powerful**
  - “Compound Document Framework”
    - Simplifies construction & modification of GUIs
    - Enables novel user interfaces (see last demo)
- **Robustness**
  - Software that crashes would not be permissible in industry

# Factors that contribute to BlackBox robustness

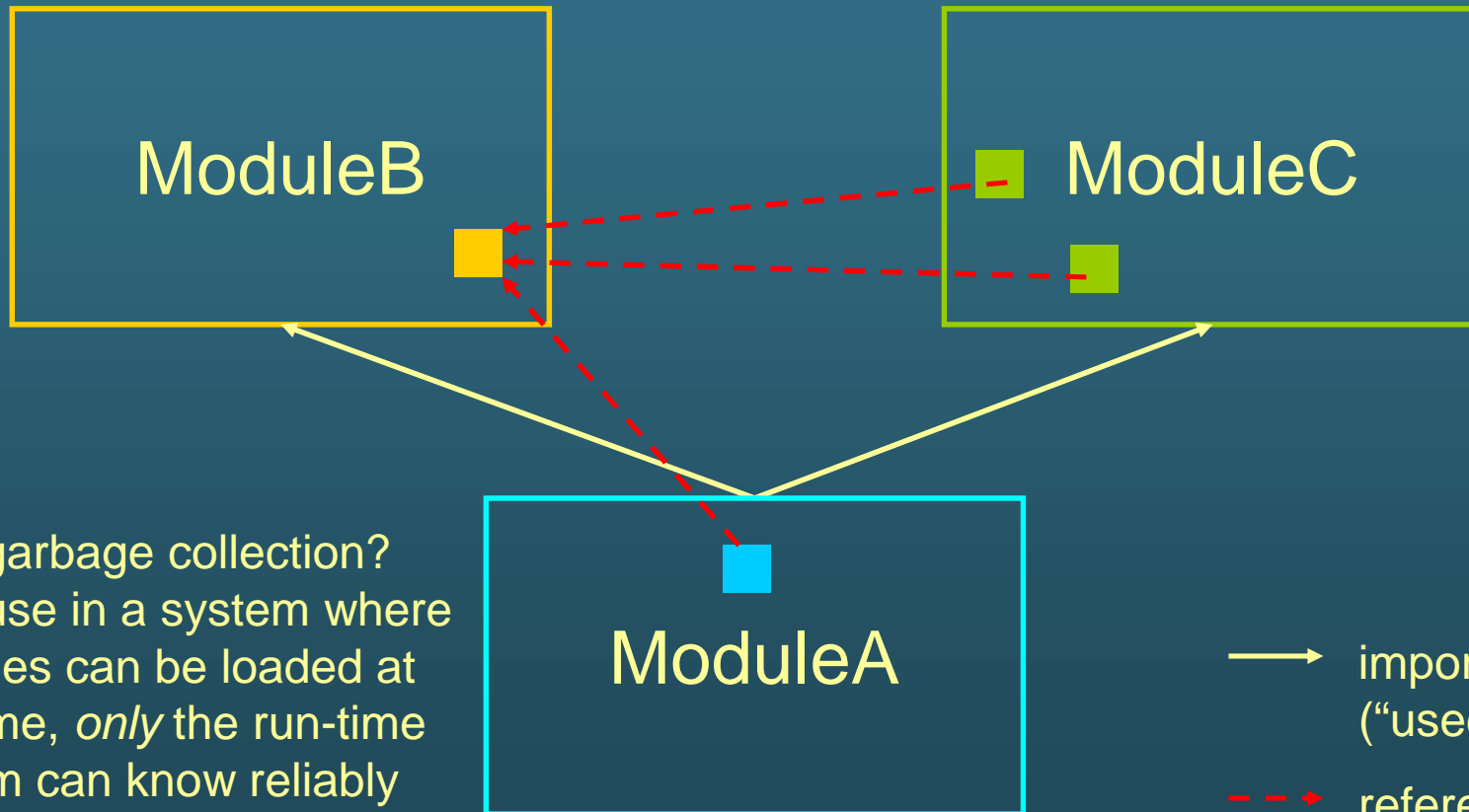
# Robustness: Language

- **Simplicity**
  - Language definition: only 35 pages
  - Language is easy to learn, easy to use
- **Strong typing**
  - Prevents **memory corruption**
    - Array index checking prevents buffer overruns, etc.
  - Limits “entropy increase” during maintenance
    - “Refactoring without fear”: improve code structure
    - Compiler shows locations affected by a change

# Robustness: Language

- Automatic memory reclamation
  - Prevents dangling pointers (memory corruption)
  - Prevents memory leaks
  - **Garbage collector**
    - Important for **dynamic data structures**
    - Indispensable for **dynamically extensible systems**
    - Additional benefit: you have less work to do

# Garbage Collection



Why garbage collection?  
Because in a system where modules can be loaded at run-time, *only* the run-time system can know reliably when the last reference to an object has been removed!

→ imported by ("used")  
- - - → references

# Robustness: Language

- Modules are
  - Independently compiled
    - Fast compilation
  - Independently developed
    - Team work
  - Dynamically loaded
    - Run-time extensibility
  - Dynamically unloaded
    - Easy debugging and run-time updates

# Robustness: Language

- Strong modularity
  - Information hiding
    - Prevents inadvertent access to private variables, procedures, etc.
      - Simplifies maintenance work (e.g., refactoring of code)
  - Cycle-free import relations
    - Simplify program comprehension

# Robustness: Design

- Unsafe or unportable code is confined explicitly to “unsafe modules”
  - Minimize and isolate unsafe spots in the code
  - Make unsafe spots easy to find
- Common design patterns
  - Carriers / Riders / Mappers, Directories, etc.
    - Simplify understanding and correct use of libraries



# Robustness: Design

- “Design by Contract”
  - Introduced by Prof. Bertrand Meyer
  - Minimum requirement:  
Checking preconditions that are inexpensive
    - Do not trust your callers (in an extensible system)
    - Example of the “Stop Sooner” principle
      - Minimizes domino effects
      - Simplifies “blame assignment”
      - Example:      PROCEDURE Quotient (x, y: REAL): REAL  
                  *Precondition: y # 0*

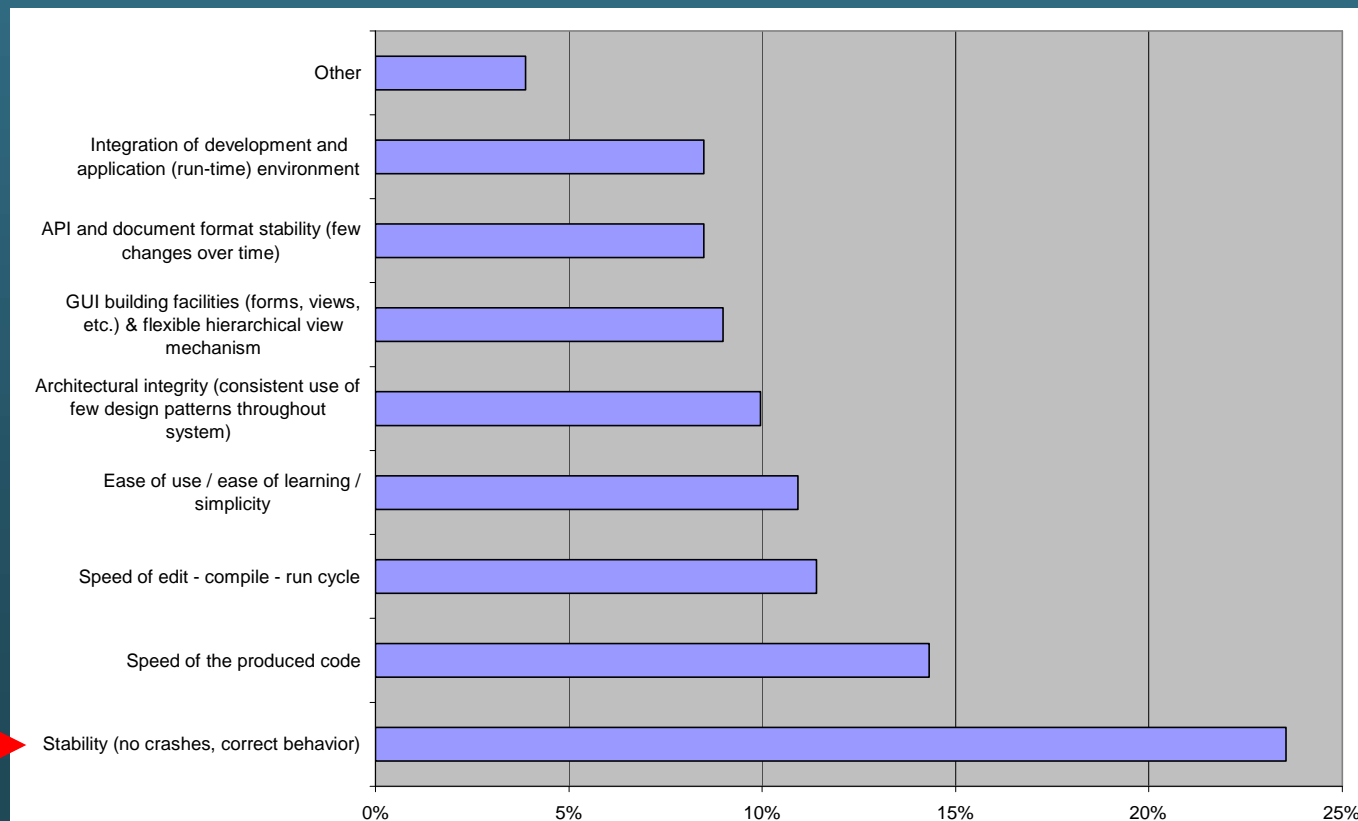
# Robustness: Design

- Object reentrance
  - Reentrance problems can make complex OO systems difficult to debug and maintain
    - We avoid them wherever possible
  - For further reading, see “Component Software – Beyond Object Oriented Programming”
    - Written by our co-founder Clemens Szyperski



# Why is Robustness Important?

- It is the most valued feature of BlackBox :



Source: BlackBox survey, February 2004

# Why is Robustness Important?

- To get results sooner
  - Less time for debugging during development
  - Crashing is *bad* → we help avoid it
- To make verification of research results easier
  - When analyzing a program, collaborators need not worry about index overflows etc.,
  - Modularity makes program analysis easier

# Engineering Unit Converter Demo

```
MODULE CernForceConverter;
```

```
IMPORT Dialog;
```

```
CONST k = 4.45;
```

```
VAR
```

```
  pound*: REAL;
```

```
  newton*: REAL;
```

```
PROCEDURE PoundToNewton*;
```

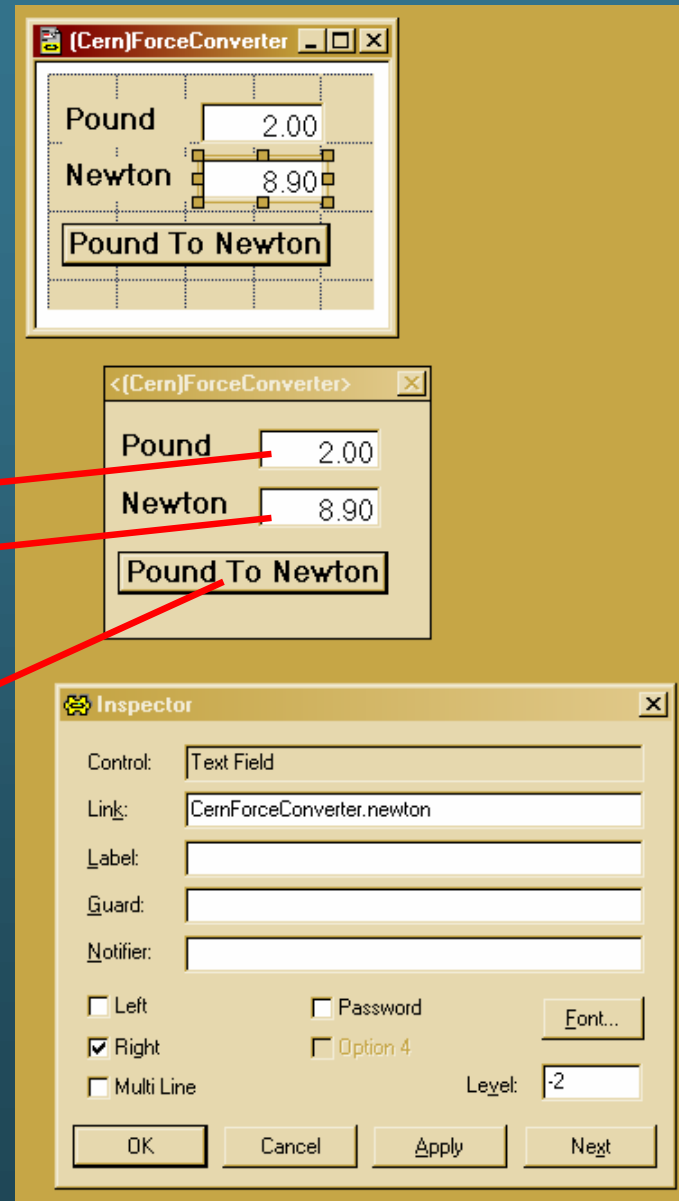
```
BEGIN
```

```
  newton := pound * k;
```

```
  Dialog.UpdateReal(newton)
```

```
END PoundToNewton;
```

```
END CernForceConverter.
```



# MONDIG Demo

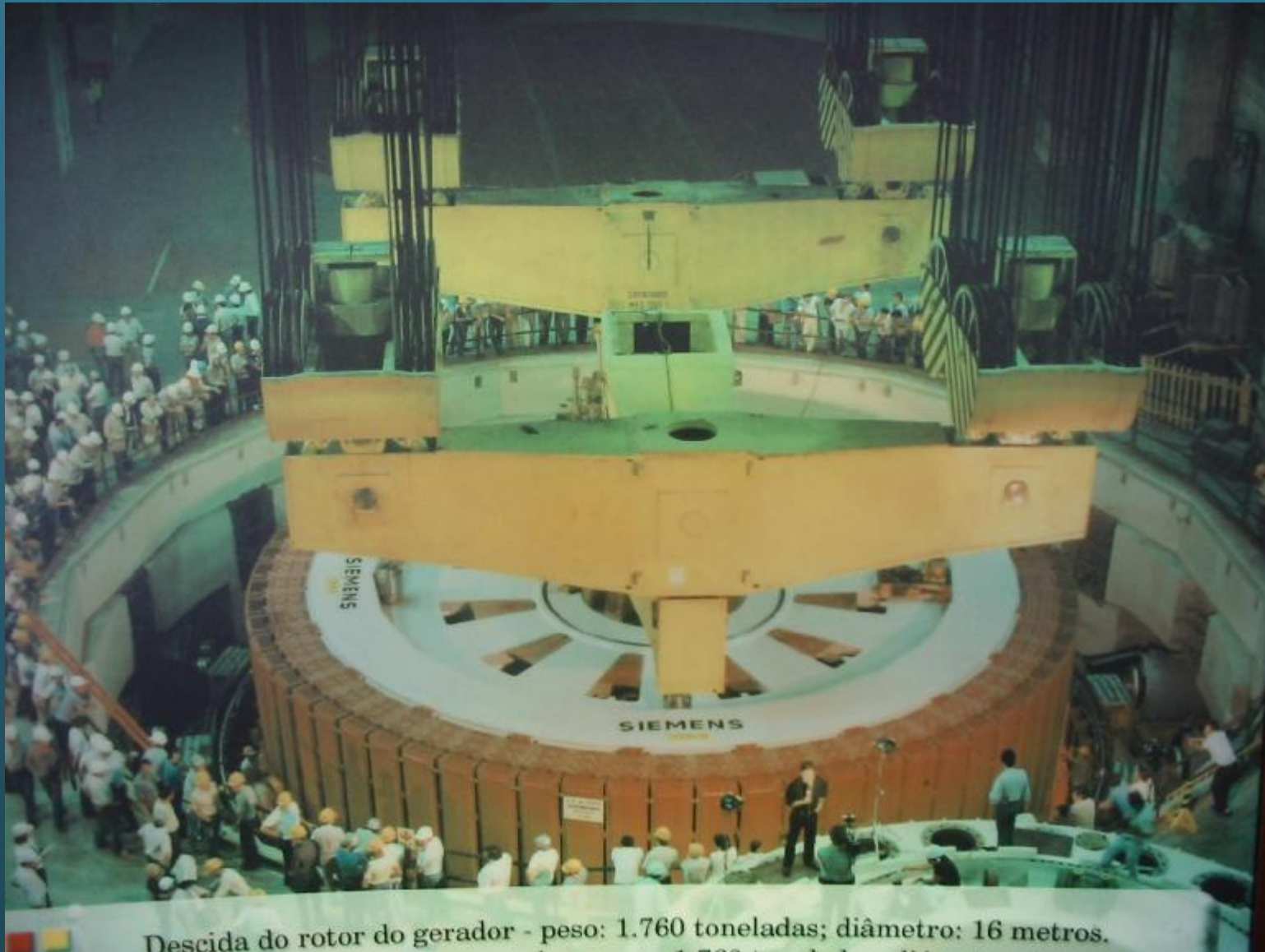
Monitoring the World's Largest  
Hydro Power Plant

# Oberon Day at CERN

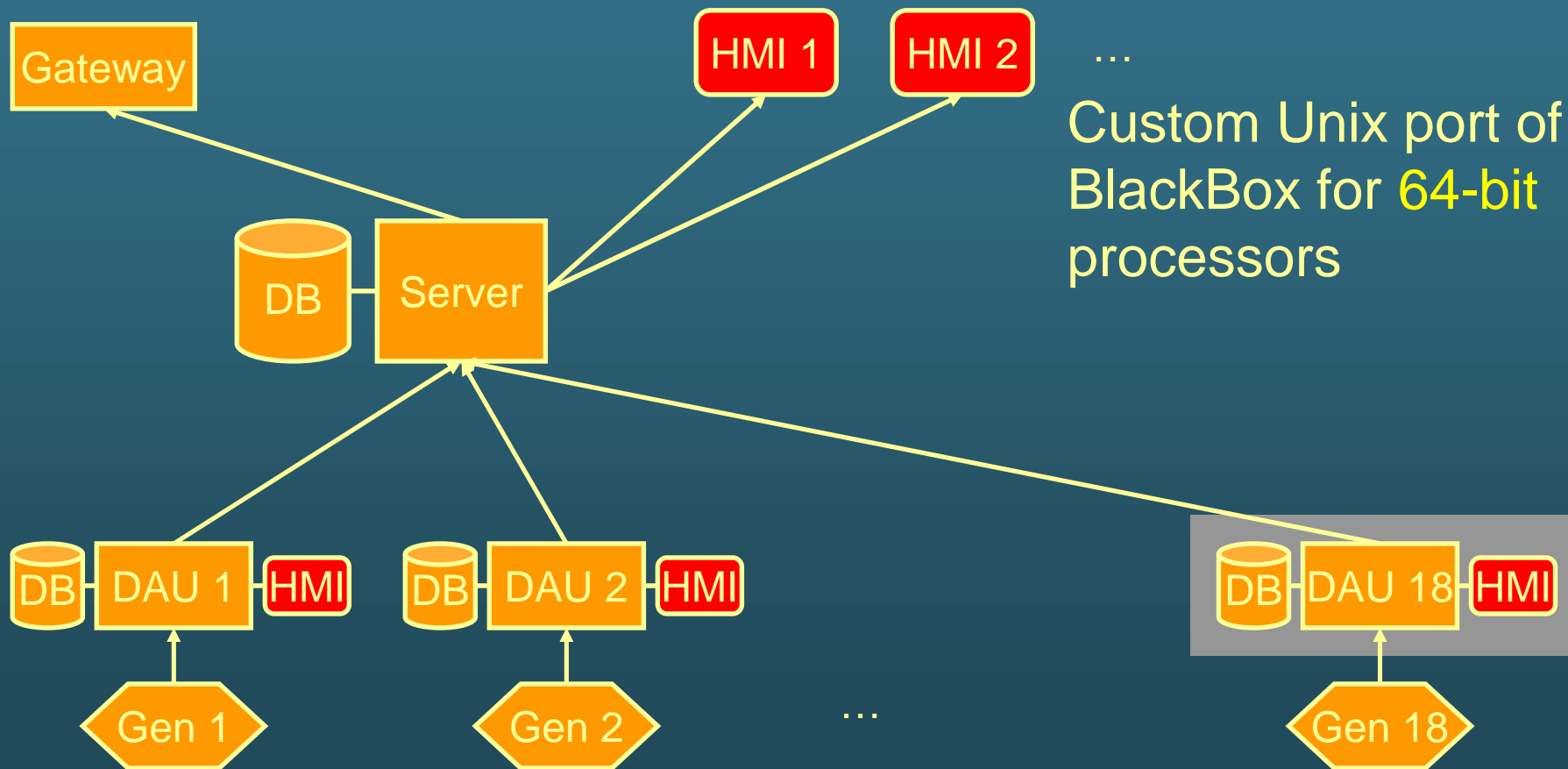




# Oberon Day at CERN



# Overview



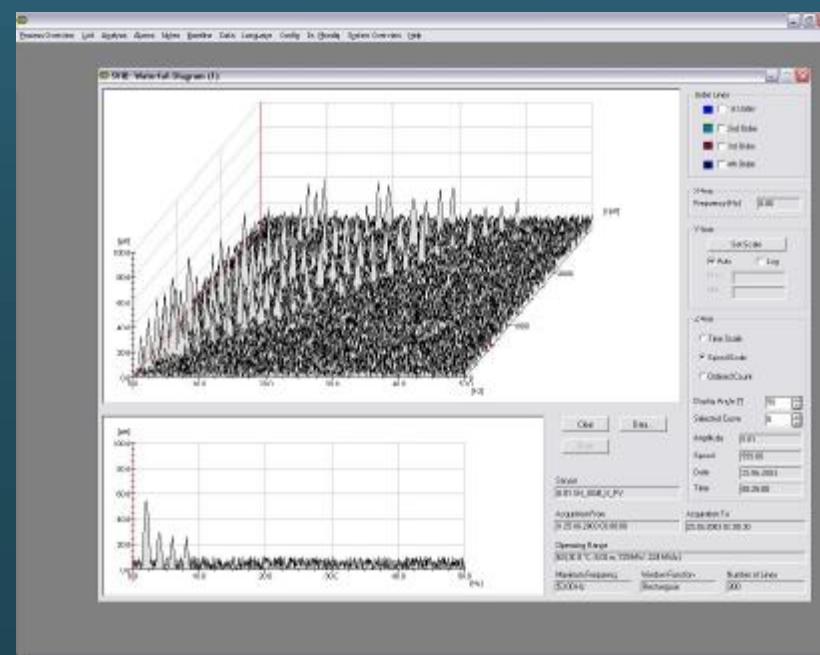
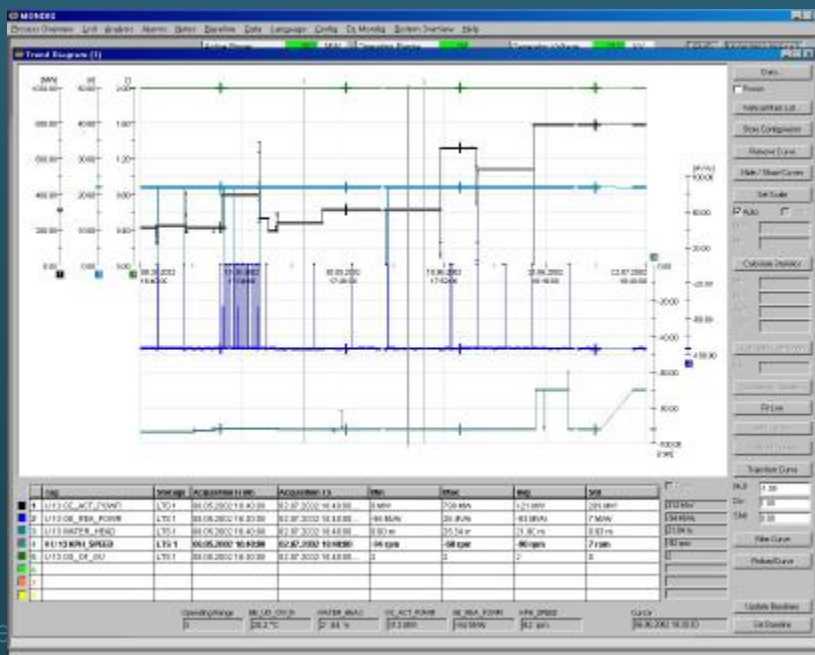
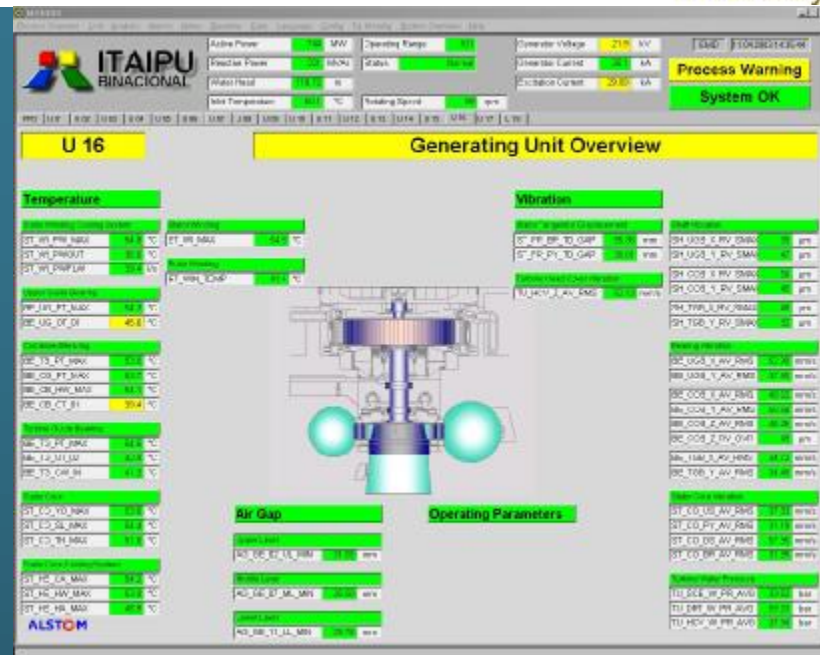
# Oberon Day at CERN



# MONDIG Statistics

- Integrated application suite
  - Process data monitoring
  - Historical data analysis
  - Alarm, baseline, note & configuration management
  - System autodiagnosis
- 30 overview screens
- 22 analysis functions
- 168 modules
- 250 compound documents



# Oberon Day at CERN






# “Physicist’s Workbench” Experimental User Interface Demo




Stator Core Vibration and Tangential Displacement 040310

Measurements of March 10, 2004 10:12 am:

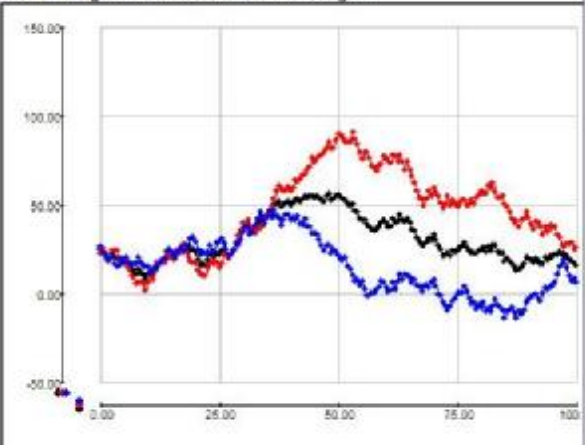
Left Direction  Right Direction 




Some computations:

Left Direction  avg Right Direction  = Left Direction avg Right Direction 


Left Direction  max Right Direction  = Left Direction max Right Direction 

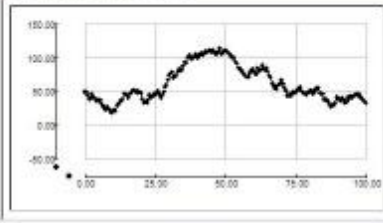
Left & Right Direction, Average:



Left Direction  + Right Direction  = Left Direction 

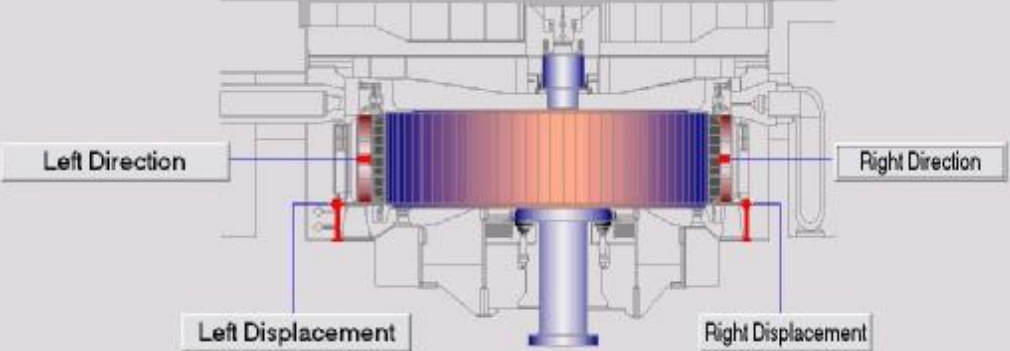
untitled16

Right Displacement 



Data Acquisition Unit

**Stator Core Vibration and Tangential Displacement**



Paste Measurement To Focus

# Some Examples from the BlackBox Community



# Example: Statistics

- Imperial College, London & Medical Research Council, Cambridge
  - Epidemiological Mapping Studies
  - Advanced statistical modeling for non-experts, using Markov chain Monte Carlo methods

# Example: Aerospace Engineering

- BAE Systems, Edinburgh
  - “How to fit a 200 m radar dish into a plane”
  - Synthetic Radar Aperture Research
  - Rapid algorithm design
- See Robert Campbell’s presentation this afternoon!



# Example: Experimental Physics (1)

- University of Rochester,  
Department of Physics and Astronomy
  - Data acquisition from CAMAC
  - Interfacing to custom waveform digitizers
  - Data analysis and Monte Carlo simulations

## Example: Experimental Physics (2)

- Laboratory for Laser Energetics
  - Rapid development of a closed-loop adaptive-optics control system
- See Wojtek Skulski's presentation this afternoon!

# Example: Theoretical Physics

- High Energy Physics
  - Perturbative quantum field theory
  - Large-scale mixed symbolic & numerical computations
  - Algorithmic experimentation
- See Fyodor Tkachov's presentation this afternoon!

# How can you use BlackBox?

- Rapid algorithm design
- Custom GUIs for equipment control and data management
- In the future:  
To run algorithms under Linux
- Developer community on the Internet
- We are close to CERN,  
we will come to work with you!

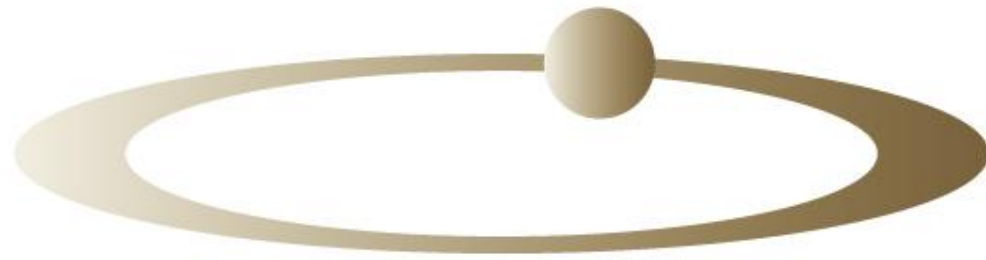
# Summary

- BlackBox is **reliable by design**
  - Made possible by good Oberon foundation
- Compound document framework
  - Enables **innovative user interfaces**
- Component Pascal as “Swiss Army Knife”
  - **Rapid algorithm design...**
  - ... to industrial-strength software engineering

# Questions







Oberon  
microsystems



SOFTWARE ARCHITECTURE