

Mozart-Oz

Multi-paradigm Programming System

Boris Mejías and the Mozart community

`www.mozart-oz.org`

`boris.mejias@uclouvain.be`



Mozart-Oz

- ▶ Mozart is an implementation of Oz, a multi-paradigm programming language supporting
 - ▶ declarative
 - ▶ functional (lazy and eager)
 - ▶ object-oriented
 - ▶ concurrent
 - ▶ distributed
 - ▶ logic
 - ▶ constraint programming
- as part of a **coherent** whole



Mozart-Oz

- ▶ Mostly used in academia but also in industry
- ▶ It runs on GNU/Linux, Solaris, MacOSX and other operating systems
- ▶ From Mozart Consortium (Belgium, Germany, Sweden) to an open Mozart community organized by a Board governance model with MEPs
- ▶ It provides the Oz Programming Interface (OPI)
- ▶ Strengths:
 - ▶ **Concurrency**: ultra lightweight threads, dataflow synchronization
 - ▶ **Inferencing**: constraint and logic programming
 - ▶ **Distribution**: network transparent, open, fault tolerant
 - ▶ **Flexibility**: dynamically typed, incremental compilation



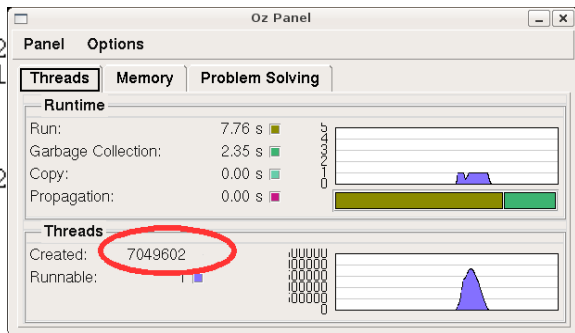
A bit of Oz code

```
declare
fun {Fibo N}
  if N < 2 then 1
  else
    Fm1 Fm2
  in
    thread Fm2 = {Fibo N-2} end
    thread Fm1 = {Fibo N-1} end
    Fm1 + Fm2
  end
end
{Browse {Fibo 32}}
```



A bit of Oz code

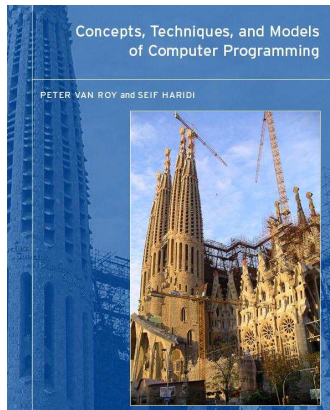
```
declare
fun {Fibo N}
  if N < 2 then 1
  else
    Fm1 Fm2
  in
    thread Fm2
    thread Fm1
    Fm1 + Fm2
  end
end
{Browse {Fibo 32}}
```



Teaching programming using Oz

“Concepts, Techniques, and Models of Computer Programming” by Peter Van Roy and Seif Haridi, published by MIT Press in 2004. (900 pages)

- ▶ One language to teach many concepts involved in all major programming paradigms
- ▶ Used for teaching in more than 20 universities worldwide
- ▶ The book is available in English, Polish, and soon in Spanish, Japanese and French



Strasheela

by Torsten Anders

- ▶ A constraint-based music composition system
- ▶ Users declaratively state a music theory model (as Oz code) – computer generates music which complies with this theory
- ▶ A theory model is implemented by a set of compositional rules (constraints) applied to a music representation in which some aspects are expressed by variables
- ▶ Results are output into various formats, e.g. music notation and sound synthesis
- ▶ Strasheela is highly programmable and extendable, e.g. users control what information is stored in the music representation



Strasheela

The screenshot displays the Oz Programming Interface (emacs@localhost.localdomain) with the following components:

- Code Editor:** Contains Oz code for generating a canon. The code defines a procedure `IsCanon` that takes two voices and a canon number, and a procedure `Canon` that generates a canon in a fifth. It also includes a solver call and a randomized solution.
- Oz Explorer:** Shows a search tree with nodes represented by colored squares and circles, and a red triangle indicating the current state.
- Musical Score:** A PDF viewer showing a musical score for two voices, with the title "03-FloridCounterpoint".
- Audio Player:** A window titled "Xpdf: /home/t/sound/tmp/out1-1.pdf" showing a waveform and playback controls for the file "out3.aiff - Ready".

```
end)
end
proc [IsCanon Voice1 Voice2]
%% The first CanonNo notes of each voice form a canon in a fifth
CanonNo = 10
in
for
Note1 in (List take (Voice1 getItems($)) CanonNo)
Note2 in (List take (Voice2 getItems($)) CanonNo)
do
(Note1 getPitch($)) + 7 =: (Note2 getPitch($))
(Note1 getDuration($)) =: (Note2 getDuration($))
end
end
%%
%% Call solver (A few different distribution strategies are proposed
%% to solve this CSP).
%%
%% Randomised solution
(SDistro exploreOne Canon
unit(order: startTime
value: random) []
/*
----- 03-FloridCounterpoint
overall samples of
0 errors in performance
Elapsed time at end of perfor
2647 1024-byte soundblks of
> sweep-audio-editor /home/t/
=U: ** *Oz Emulator* (C...
```


SCOLL Safe Collaboration Language

by Fred Spiessens and Yves Jaradin



Provide SCOLL Pattern

powered by 

Developed by:

[Fred Spiessens](#)

[Yves Jaradin](#)

| SCOLLAR Calculations | Saved Patterns | Saved Systems |
|-------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|-----------------------------------------|
| <p>Solutions</p> <p>1 Solution</p> <p>Min Fixpoint</p> <p>Max Fixpoint</p> <p>Recalculate every <input type="text" value="1"/> nodes.</p> | <p>choose</p> <p>save pattern as ...</p> <p>save "simplect1"</p> <p>remove "simplect1"</p> | <p>choose</p> <p>save system as ...</p> |

system

```
permission: access/2
behavior: may.sendTo/3 may.getFrom/2 may.return/2 may.receive/1
knowledge: did.sendTo/3 did.getFrom/3 did.return/2 did.receive/2
system
access(A,B) access(A,X) A:may.sendTo(B,X) B:may.receive()
=> access(B,X) A:did.sendTo(B,X) B:did.receive(X);
access(A,B) access(B,X) A:may.getFrom(B) B:may.return(X)
=> access(A,X) A:did.getFrom(B,X) B:did.return(X);
```

behavior

```
UNKNOWN: { => may.sendTo(A,B) may.getFrom(B) may.return(X) may.receive();}
MINIMAL: {}
ALICE: { isBob(B) isCaretaker(C) => may.sendTo(B,C);}
PROXY: {
=> may.receive();
isCarol(C) => may.getFrom(C);
isCarol(C) did.receive(X) => may.sendTo(C,X);
isCarol(C) did.getFrom(C,X) => may.return(X);
}
```

subject

```
alice: ALICE
bob: UNKNOWN
caretaker: PROXY
? carol: MINIMAL
```

config

```
access(alice,alice) access(alice,bob)
access(alice,caretaker) access(alice,carol)
access(bob,bob)
access(caretaker,caretaker) access(caretaker,carol)
access(carol,carol)
alice:isBob(bob) alice:isCaretaker(caretaker)
caretaker:isCarol(carol)
```

goal

```
!access(bob,carol)
```



SCOLL Safe Collaboration Language

by Fred Spiessens and Yves Jaradin

SCOLLAR INPUT

Provide SCOLL Pattern

powered by  Developed by:
[Fred Spiessens](#)
[Yves Jaradin](#)

| SCOLLAR Calculations | Saved Patterns | Saved Systems |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|
| <input type="button" value="Solutions"/> <input type="button" value="1 Solution"/> <input type="button" value="Min Fixpoint"/> <input type="button" value="Max Fixpoint"/> Recalculate every <input type="text" value="1"/> nodes. | <input type="button" value="choose"/> <input type="button" value="save pattern as ..."/> <input type="button" value="save 'simplect1'"/> <input type="button" value="remove 'simplect1'"/> | <input type="button" value="choose"/> <input type="button" value="save system as ..."/> |

system
permission: access/2
behavior: may.sendTo/3 may.getFrom/2 may.return/2 may.receive/1
knowledge: did.sendTo/3 did.getFrom/3 did.return/2 did.receive/2
system
access(A,B) access(A,X) A:may.sendTo(B,X) B:may.receive()
=> access(B,X) A:did.sendTo(B,X) B:did.receive(X);
access(A,B) access(B,X) A:may.getFrom(B) B:may.return(X)
=> access(A,X) A:did.getFrom(B,X) B:did.return(X);


behavior
UNKNOWN: { => may.sendTo(A,B) may.getFrom(B) may.return(X) may.receive(); }
MINIMAL: {}
ALICE: { isBob(B) isCaretaker(C) => may.sendTo(B,C); }
PROXY: {
=> may.receive();
isCarol(C) => may.getFrom(C);
isCarol(C) did.receive(X) => may.sendTo(C,X);
isCarol(C) did.getFrom(C,X) => may.return(X);
}

subject
alice: ALICE
bob: UNKNOWN
caretaker: PROXY
? carol: MINIMAL

config
access(alice,alice) access(alice,bob)
access(alice,caretaker) access(alice,carol)
access(bob,bob)
access(caretaker,caretaker) access(caretaker,carol)
access(carol,carol)
alice:isBob(bob) alice:isCaretaker(caretaker)
caretaker:isCarol(carol)

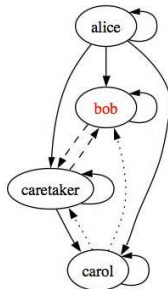
goal
!access(bob,carol)

SCOLLAR RESULTS

Successfully parsed SCOLL program powered by 

Search was completed

Calculated 2 solutions (all alive) in 0 Seconds.



| Solutions | 1 | 2 |
|-----------------------------|---|---|
| carol:may_receive() | 1 | 0 |
| carol:may_return(carol) | 0 | 0 |
| carol:may_sendTo(bob carol) | 0 | 1 |



LOGIS Caster Scheduler

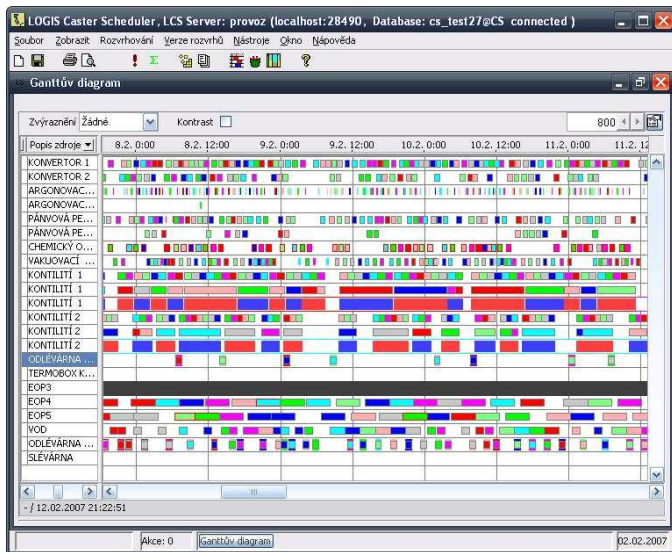
by Filip Konvička and LOGIS, s.r.o.

- ▶ It is a commercial planning/scheduling tool for continuous ingot steel casting plants
- ▶ Client/server application (Oz-based server, Java-based GUI clients)
- ▶ Users provides business and technological constraints from metal industry, and the application produce a schedule for the plant
- ▶ Able to produce a month's schedule for a medium-sized steel plant (about 200,000 tons/month) within 20 minutes. Previous methodologies never allowed plants to produce month's schedule.
- ▶ Developed and used in Czech Republic



LOGIS Caster Scheduler

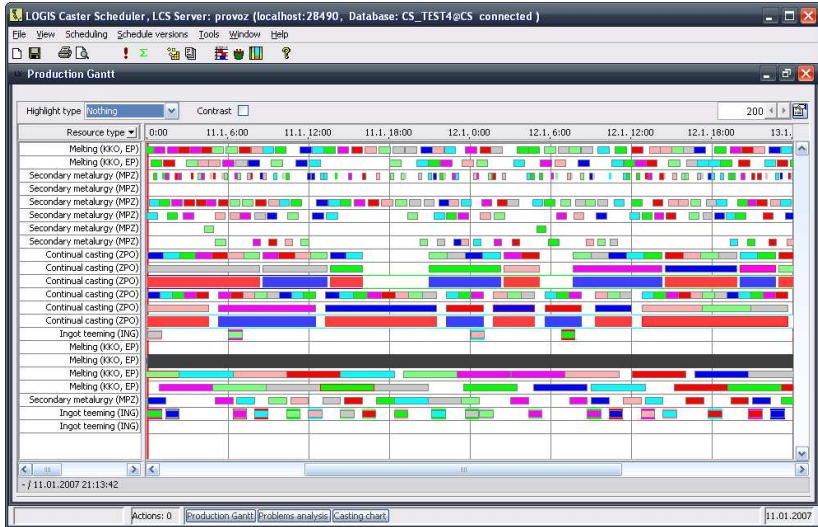
by Filip Konvička and LOGIS, s.r.o.



mozart

LOGIS Caster Scheduler

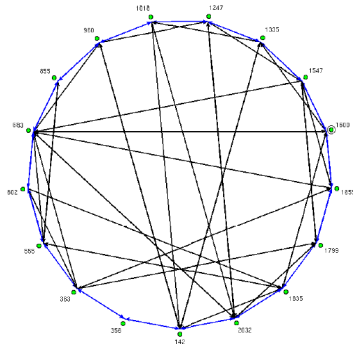
by Filip Konvička and LOGIS, s.r.o.



Peer-to-peer libraries P2PS/P2PKit

by Valentin Mesaros, Bruno Carton and Kevin Glynn

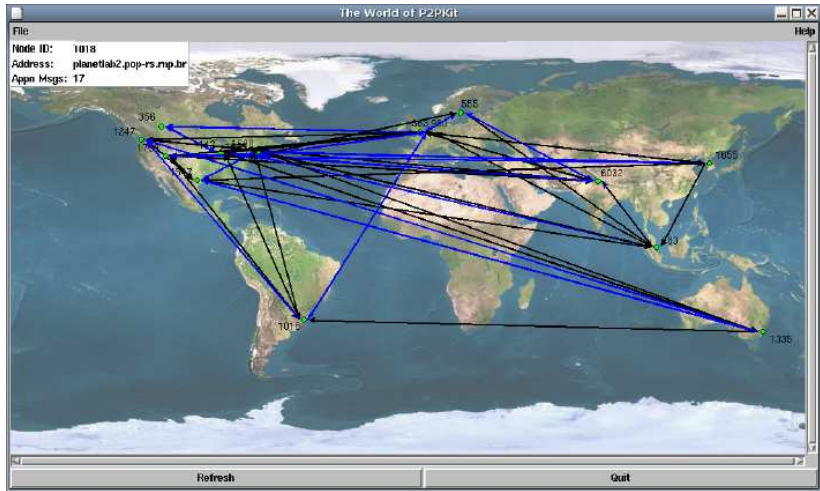
- ▶ Self optimized Chord-alike structured overlay network organized by successor, predecessor and finger-table
- ▶ Tolerant to link and processes failures



Peer-to-peer libraries P2PS/P2PKit

by Valentin Mesaros, Bruno Carton and Kevin Glynn

P2PS/P2PKit running on PlanetLab



Enhanced Binding Library EBL/tk

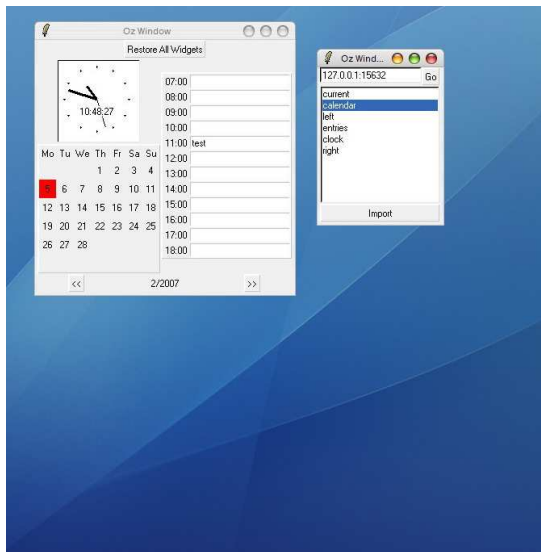
by Donatien Grolaux

- ▶ State-of-the-art toolkit for graphical interfaces
- ▶ It mixes declarative and object-oriented approaches
- ▶ 1/3 lines of code compare to standard toolkits (Swing, AWT, GTK, etc.)
- ▶ Each window component is freely detachable from its original place, and can be dynamically attached to any other EBL/tk window
- ▶ This dynamic migration process is completely transparent to the running application itself
- ▶ Migration of UI can be done to a different machine
- ▶ Seamlessly integrated in Mozart



Enhanced Binding Library EBL/tk

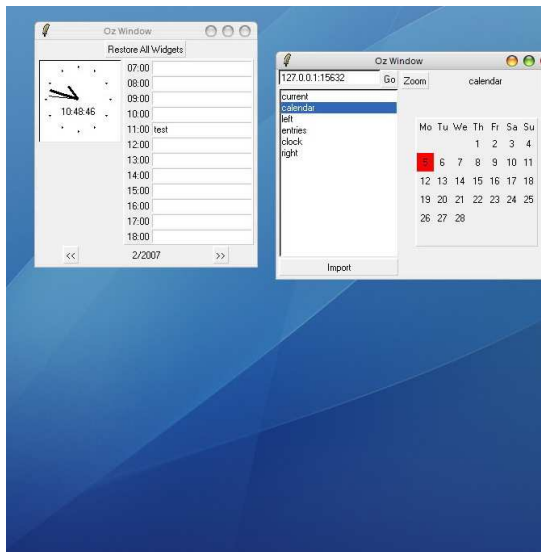
by Donatien Grolaux



mozart

Enhanced Binding Library EBL/tk

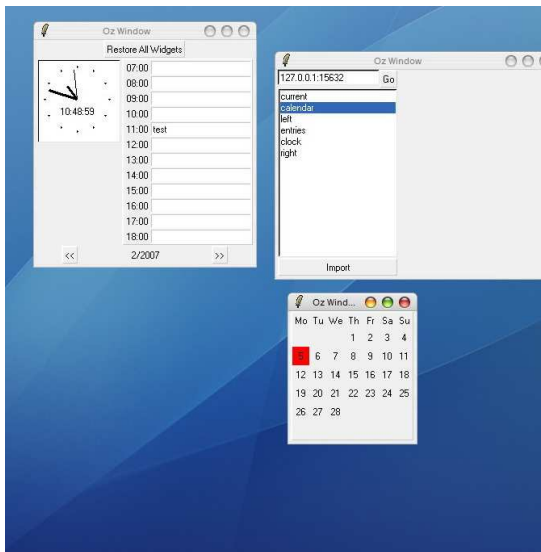
by Donatien Grolaux



mozart

Enhanced Binding Library EBL/tk

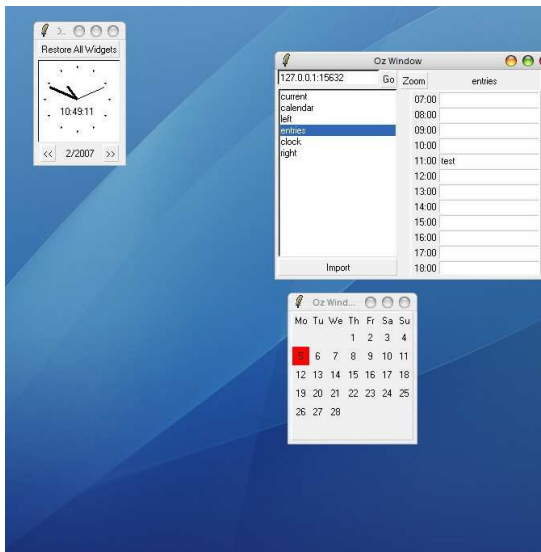
by Donatien Grolaux



mozart

Enhanced Binding Library EBL/tk

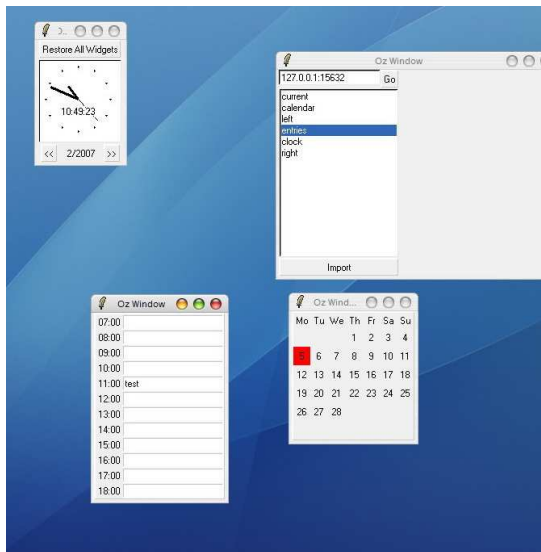
by Donatien Grolaux



mozart

Enhanced Binding Library EBL/tk

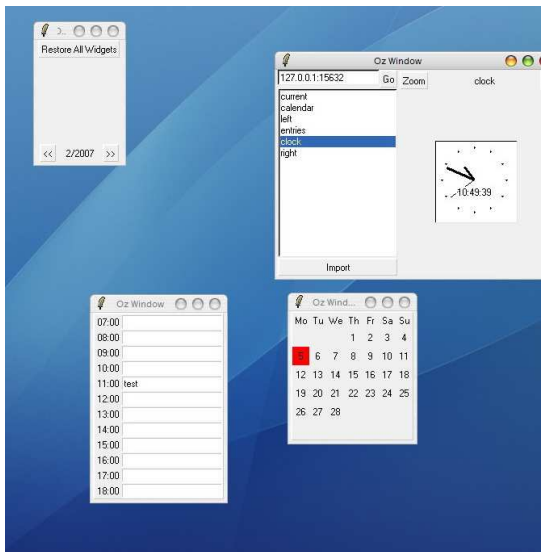
by Donatien Grolaux



mozart

Enhanced Binding Library EBL/tk

by Donatien Grolaux



m^{oz}art

Solving package installation problems

by Sébastien MOUTHUY

- ▶ Check that any package proposed in a distribution could be installed with respect to its dependencies requirements (same as aptitude, yum, etc)
- ▶ NP-Hard problem solved with constraint programming
- ▶ New search heuristics giving solutions for any package in less than 2 seconds
- ▶ It can find an installation solution to all packages of the **entire Debian distribution** (33200 different packages) in less than 1h50. Much faster than SAT solvers
- ▶ Simple implementation using cheap threads and data-flow synchronization



Current Projects

- ▶ **MozDSS**: Integration with middleware for transparent distribution support called Distribution SubSystem (Erik Klintskog, Raphaël Collet, Boriss Mejías)
- ▶ **GeOz**: Integration with Gecode, a state-of-the-art constraint programming library (Gustavo Gutierrez et al. in Colombia)
- ▶ **EVERGROW**: European Project supporting our peer-to-peer development
- ▶ **SELFMAN**: European Project to study large self-managing distributed applications based on structured overlay networks



Final Message

- ▶ **Mozart-Oz** is a powerful and mature programming system
 - ▶ Since 1995
 - ▶ Around 10^6 lines of code
- ▶ It supports all major programming paradigms giving you the possibility of choosing the right one for every problem
- ▶ Widely used for constraint programming
- ▶ It supports cheap concurrency and distributed programming transparently
- ▶ Not only for academia. Also professional software development.



Useful links

- ▶ **Mozart-Oz:** www.mozart-oz.org
- ▶ **Strasheela:** strasheela.sourceforge.net
- ▶ **SCOLL:**
www.info.ucl.ac.be/~fsp/scollardocmain.html
- ▶ **LOGIS Caster Scheduler:** www.logis.cz
- ▶ **P2PS:** gforge.info.ucl.ac.be/projects/p2ps
- ▶ **P2PKit:** p2pkit.info.ucl.ac.be
- ▶ **EVERGROW:** www.evergrow.org
- ▶ **SELFMAN:** www.ist-selfman.org

