SISCOG

a story written in Lisp

Tiago Maduro Dias
Luís Borges de Oliveira

June 2nd, 2013

Agenda

- Story so far
- Problem
- Products
- Lisp

Foundation (1986)

- Office
  - **90 square meters**
- Staff
  - **Two founders**
    - No salary
    - Did everything
  - Some **part-time programmers**
  - **One full-time** secretary
- Two Explorer Lisp machines with KEE
- One dream, looking for a goal
- First software exportation in Portuguese history (NS, 1993)

Our Clients

London Underground

Lisbon Metro

Suburban trains of Copenhagen

Barcelona Metro

Dutch Railways

Finnish Railways

Norwegian Railways

Danish Railways

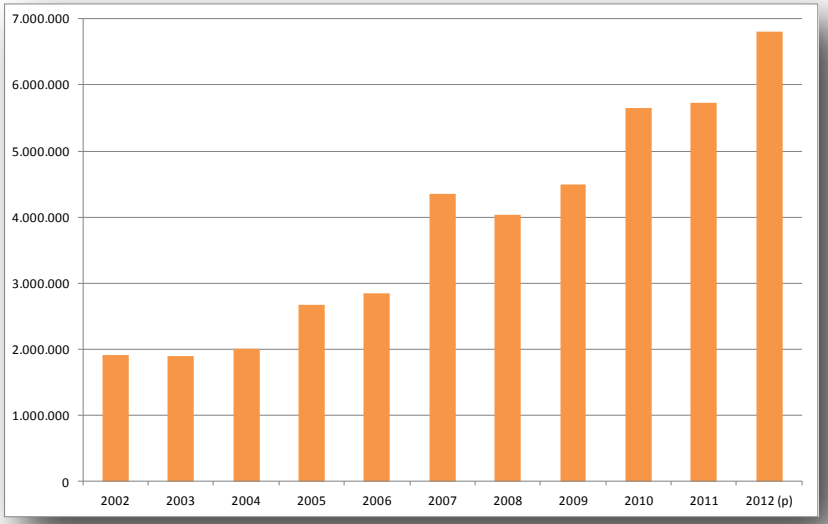
Company with largest number of systems in production in the rail domain

SISCOG
SISTEMAS COGNITIVOS

SISCOG (1986-2013)

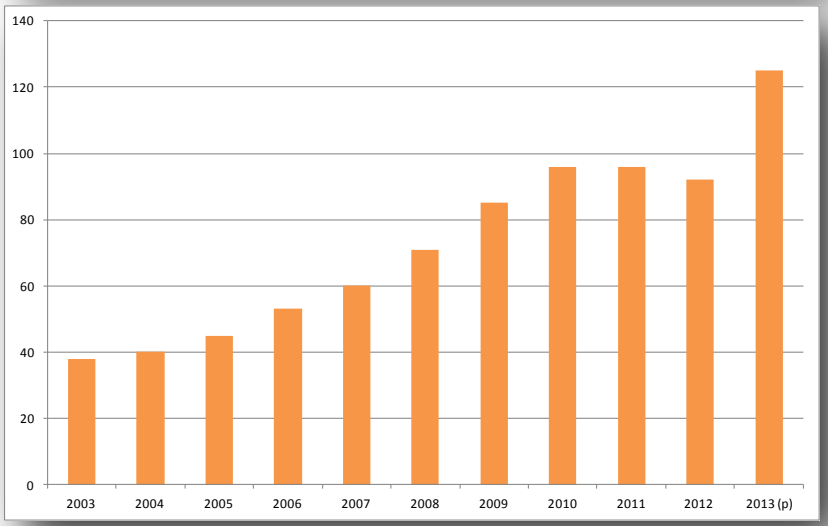
SISCOG
SISTEMAS COGNITIVOS

Present



Sales (€)

- Projected 7M€ surpassed in 2012



Staff (nr.)

- > 100
- 95% with studies in engineering and IT

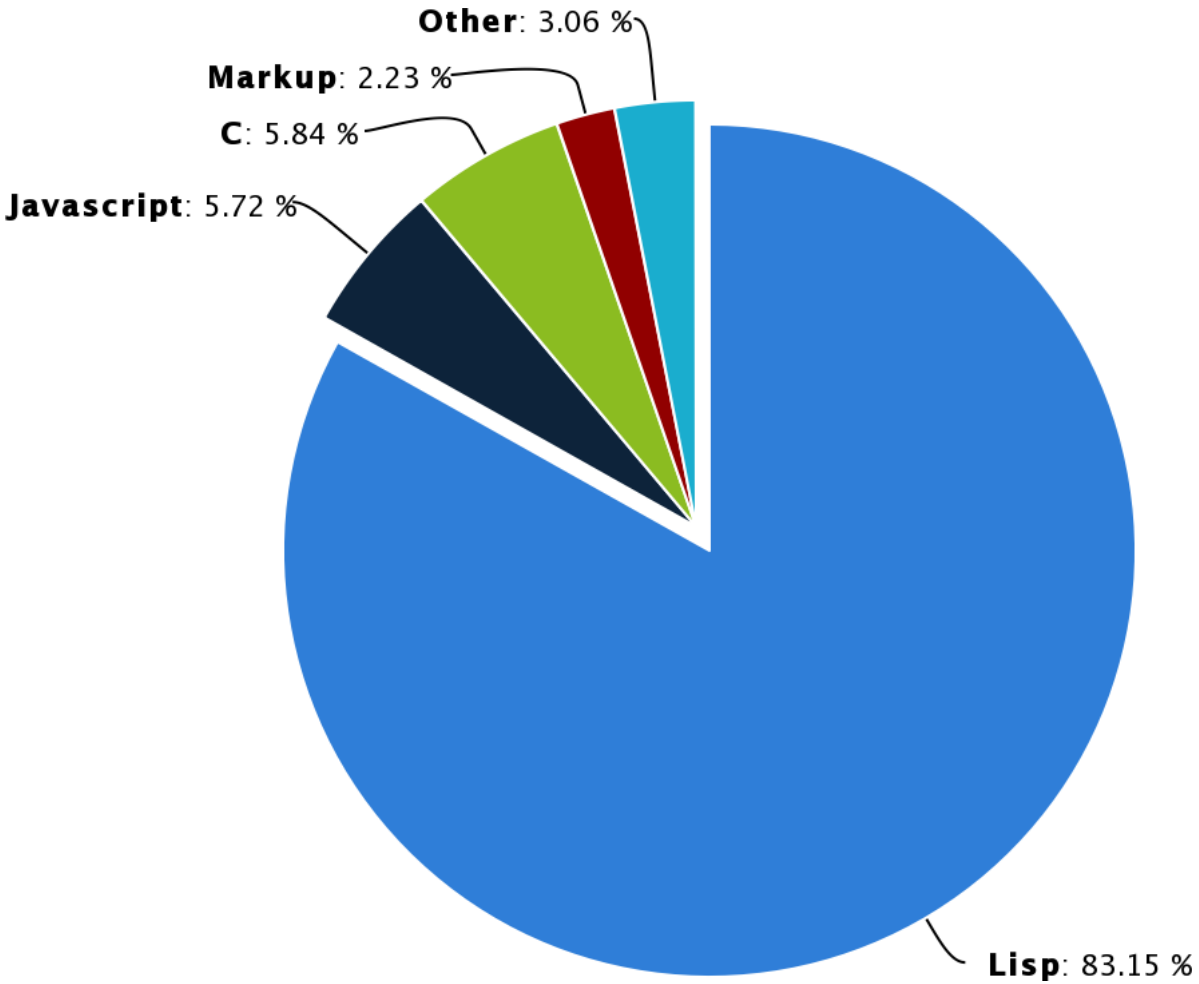
SISCOG
SISTEMAS COGNITIVOS

Present

- Offices in **Lisbon and Oporto**
- **Over 100** full-time employees
- **Product-based** company playing in a **niche market**
- **Customers** are railways and subway companies
- Our **competitors** are german, swedish, canadian and dutch (world championship)

SISCOG
SISTEMAS COGNITIVOS

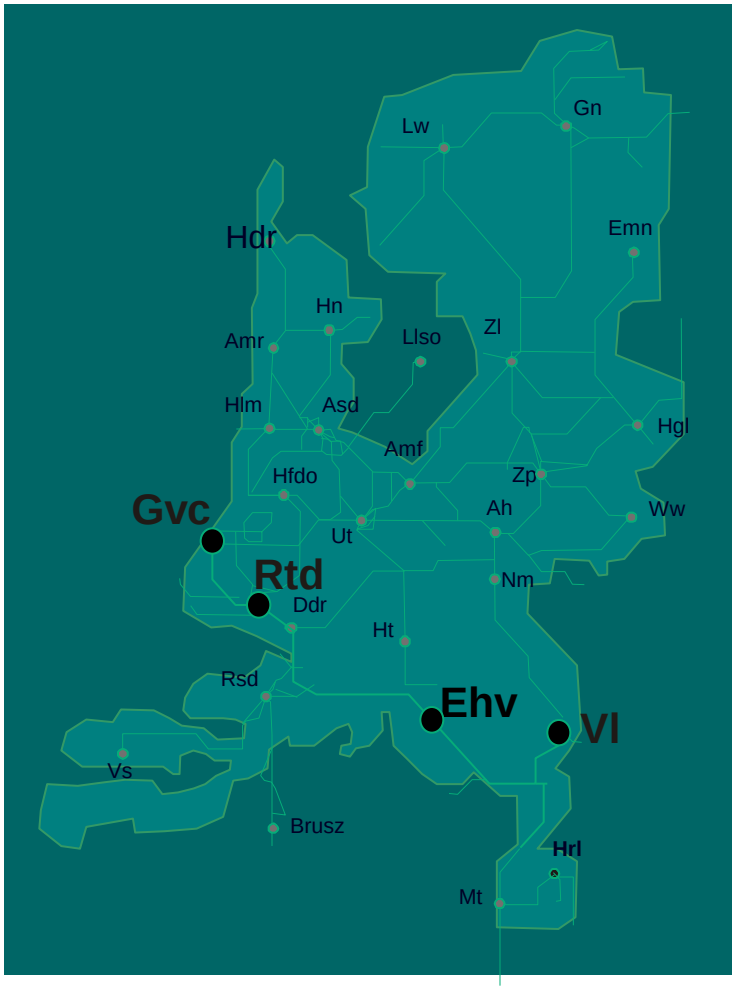
Language Distribution (May 2013)



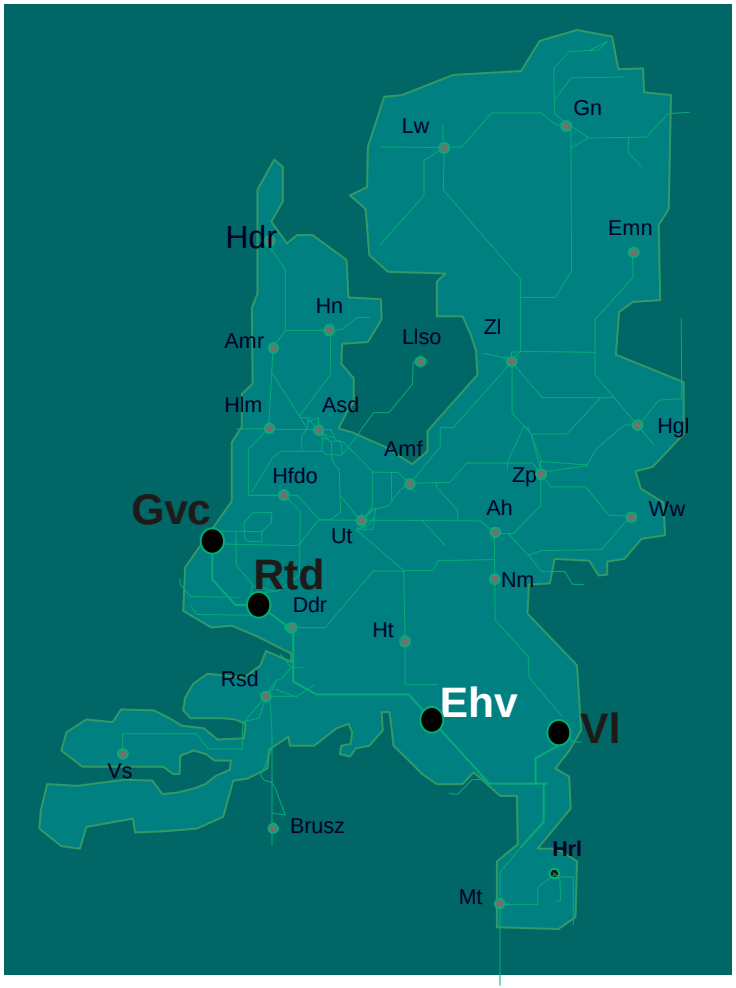
SISCOG
SISTEMAS COGNITIVOS

The job

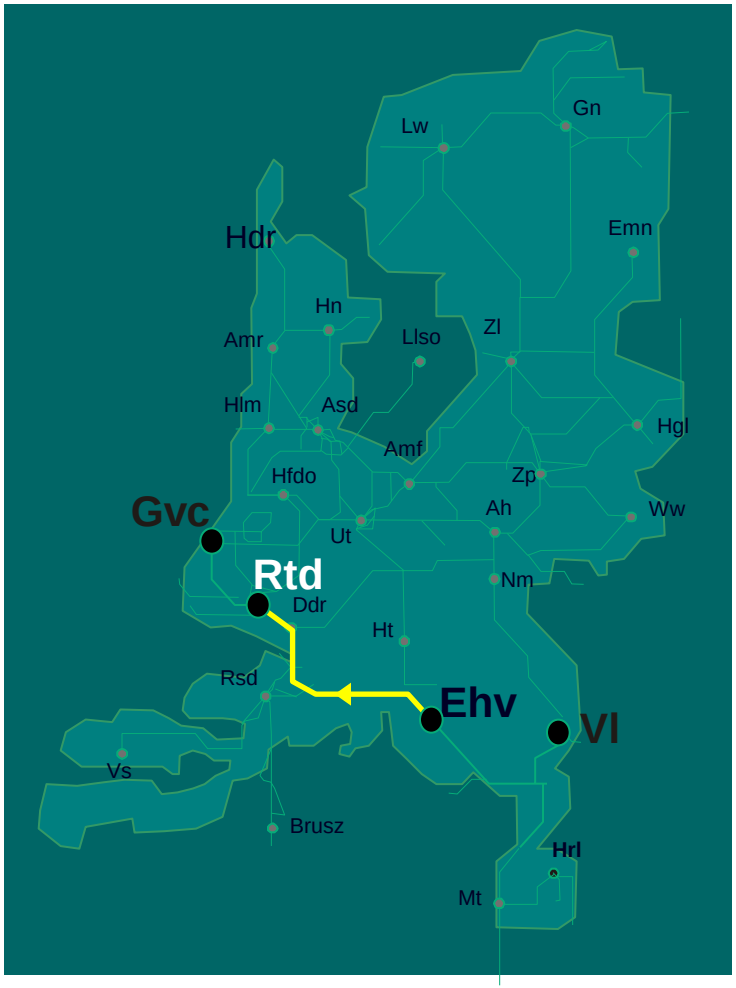
- Customers **supply transportation services**
  - Meet certain transportation demands
- Transportation services need **resources**
  - Track, stations, vehicles, crew, etc.
- Job 1: make an **operational plan**
  - Assign resources to services in an optimized way
- Job 2: **adjust the plan** because some resources become suddenly unavailable for a certain period of time
  - Track, vehicles, crew, etc.
- SISCOG provides **tools** for these complex jobs (performed by experts)



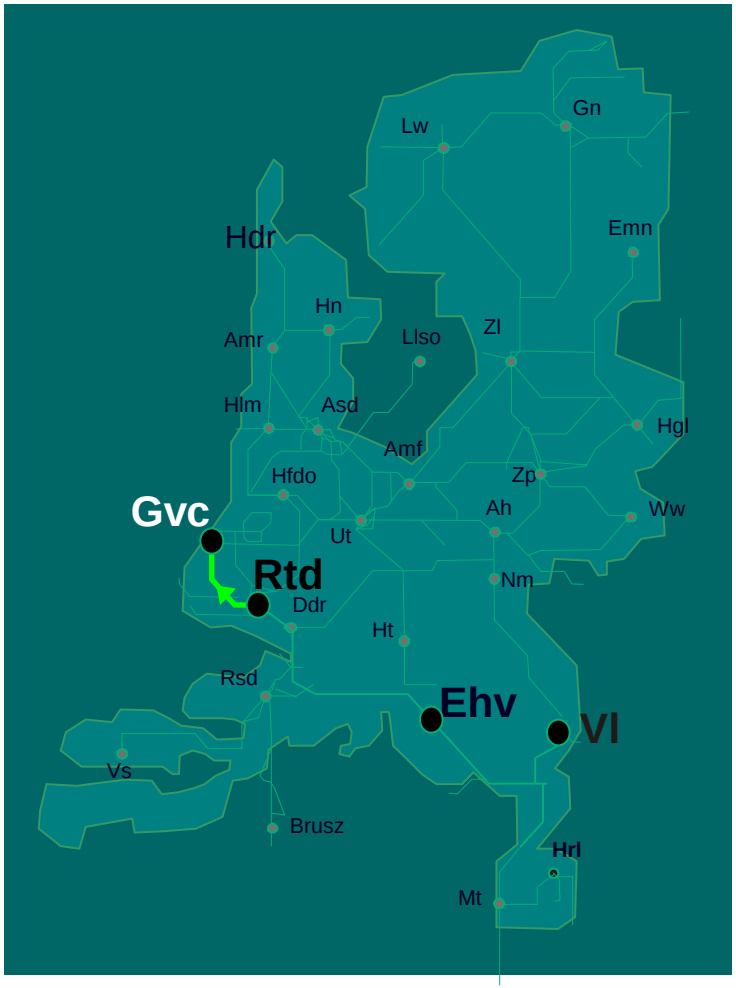
SISCOG
SISTEMAS COGNITIVOS



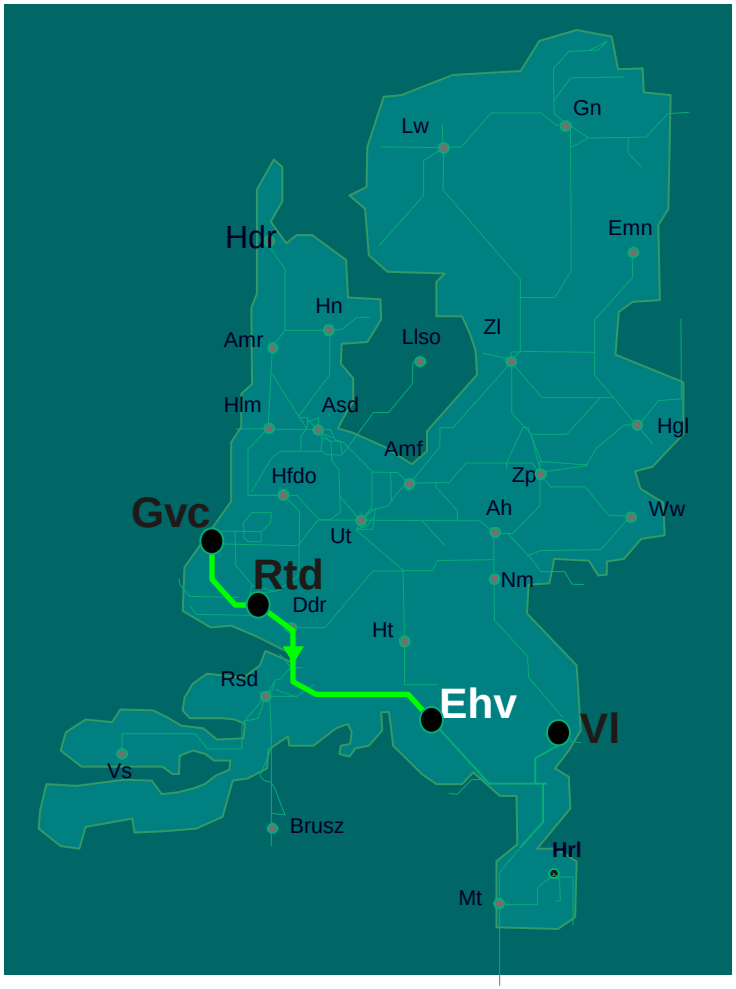
SISCOG
SISTEMAS COGNITIVOS

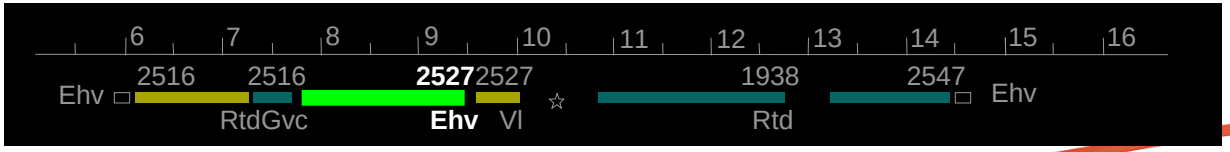


SISCOG
SISTEMAS COGNITIVOS

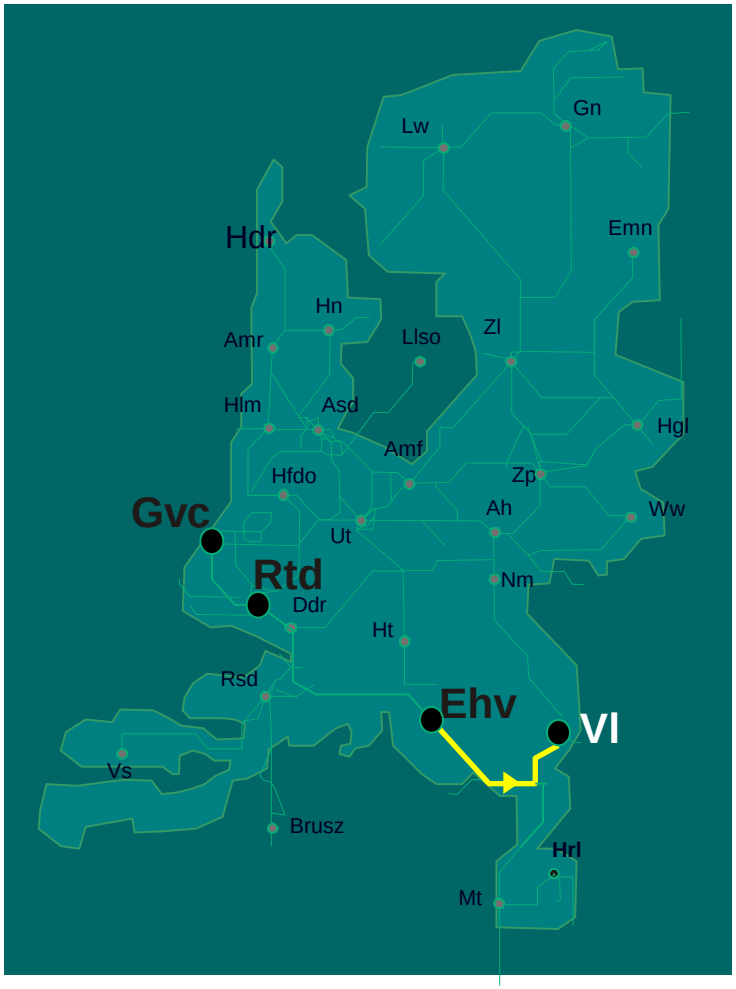


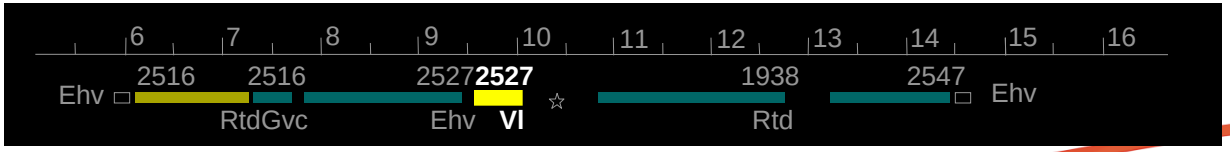
SISCOG
SISTEMAS COGNITIVOS



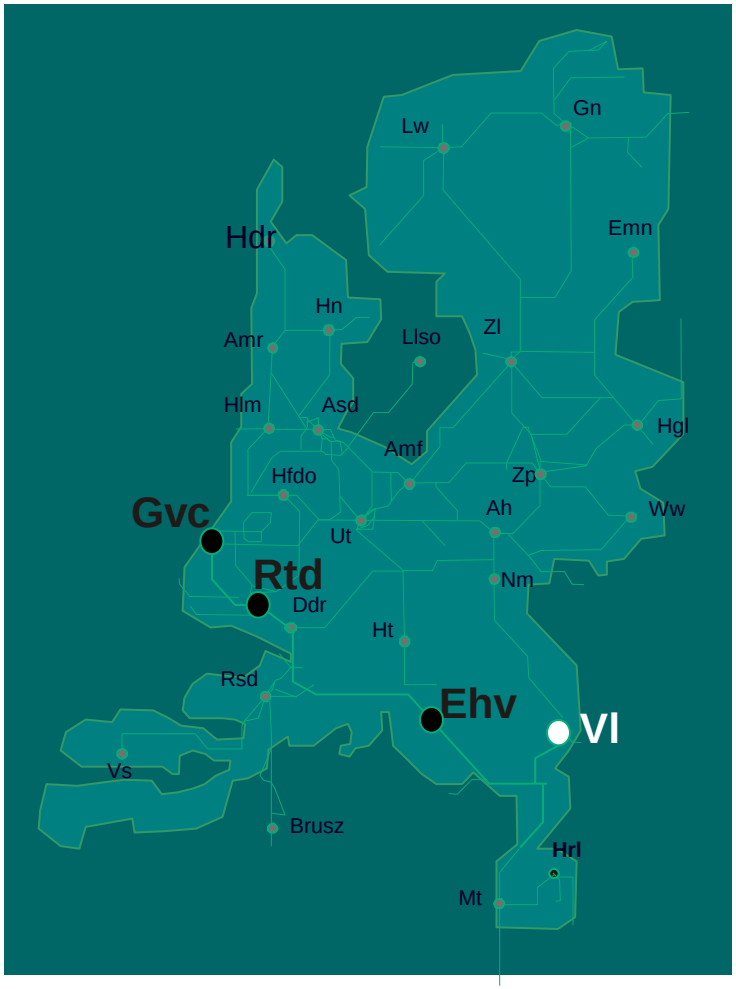


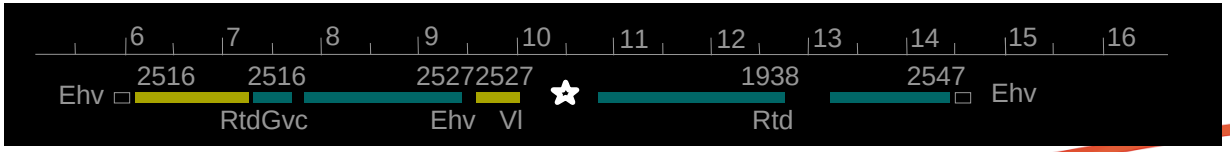
SISCOG
SISTEMAS COGNITIVOS



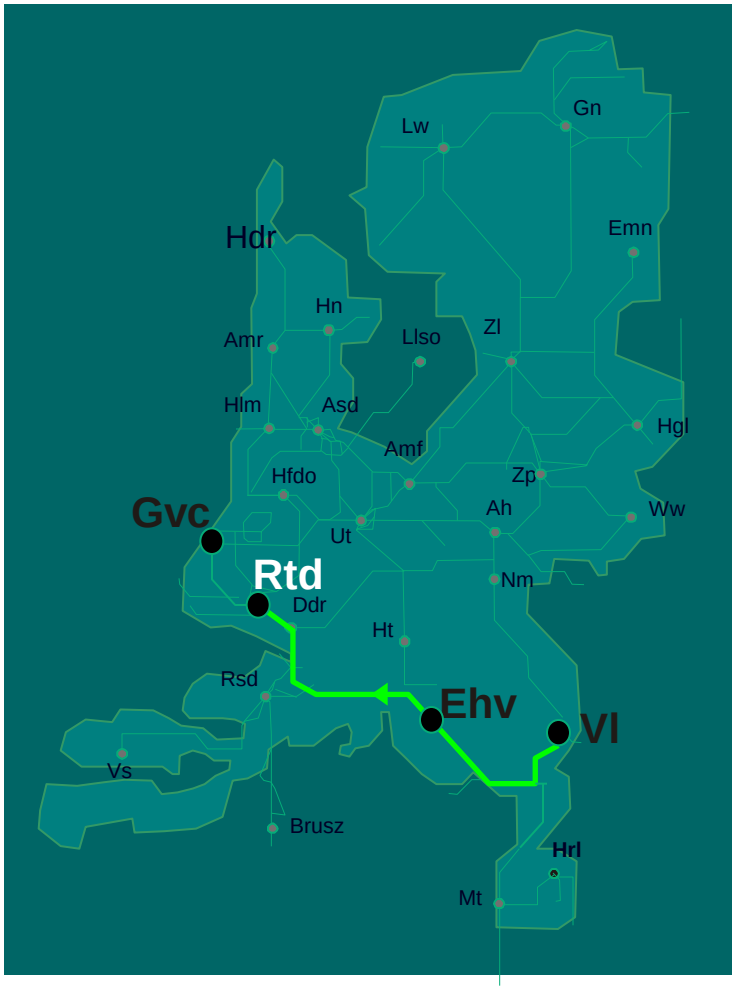


SISCOG
SISTEMAS COGNITIVOS

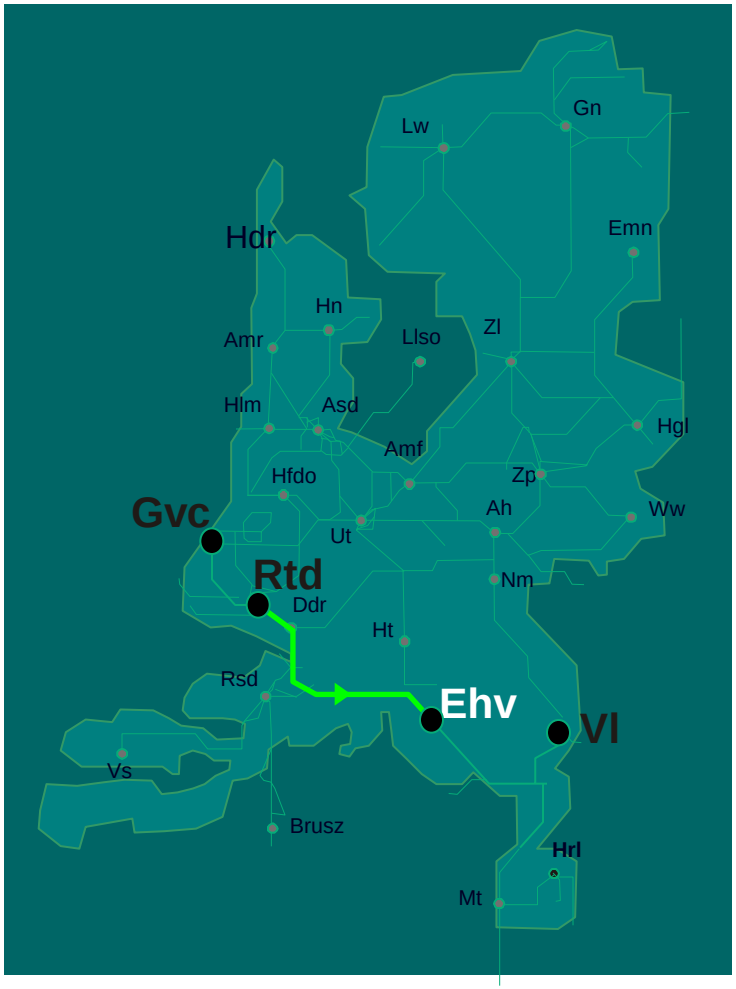


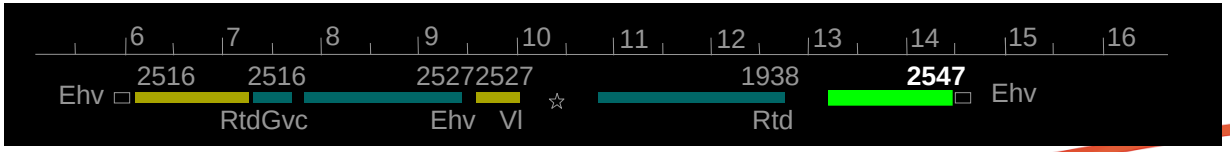


SISCOG
SISTEMAS COGNITIVOS

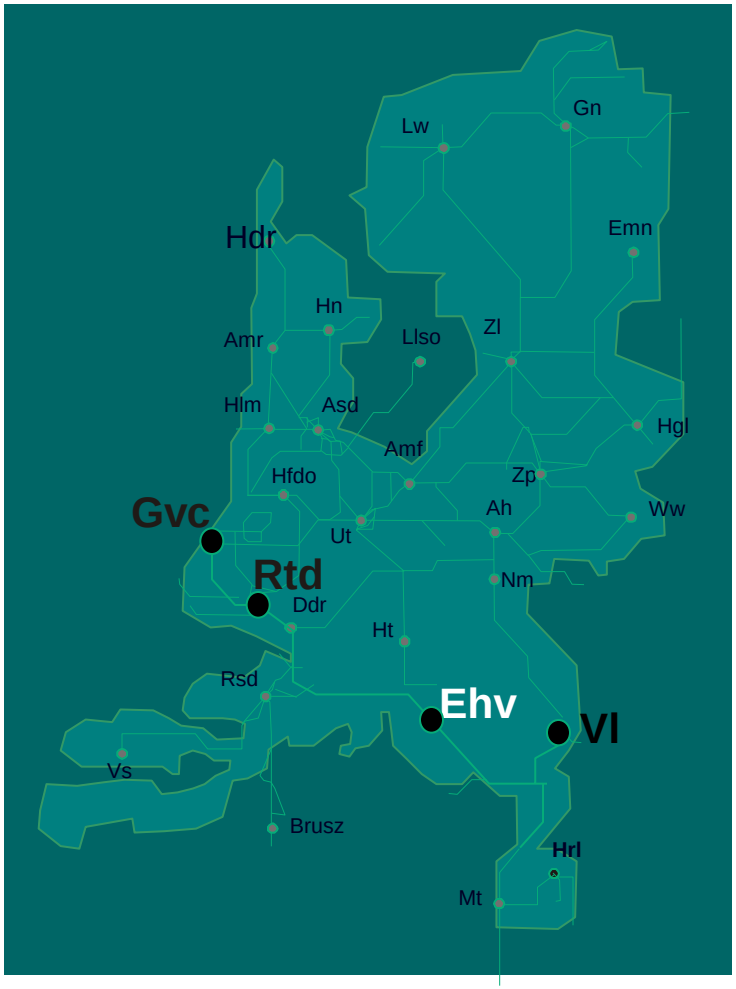


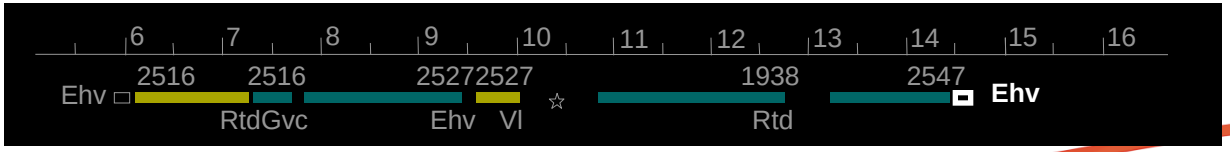
SISCOG
SISTEMAS COGNITIVOS



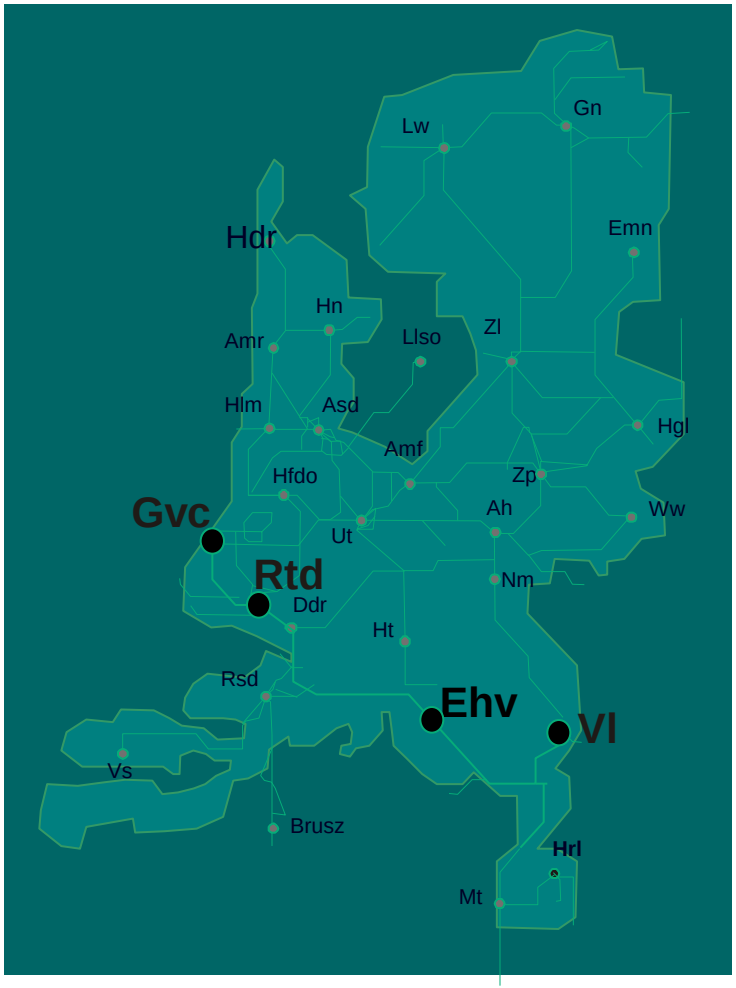


SISCOG
SISTEMAS COGNITIVOS



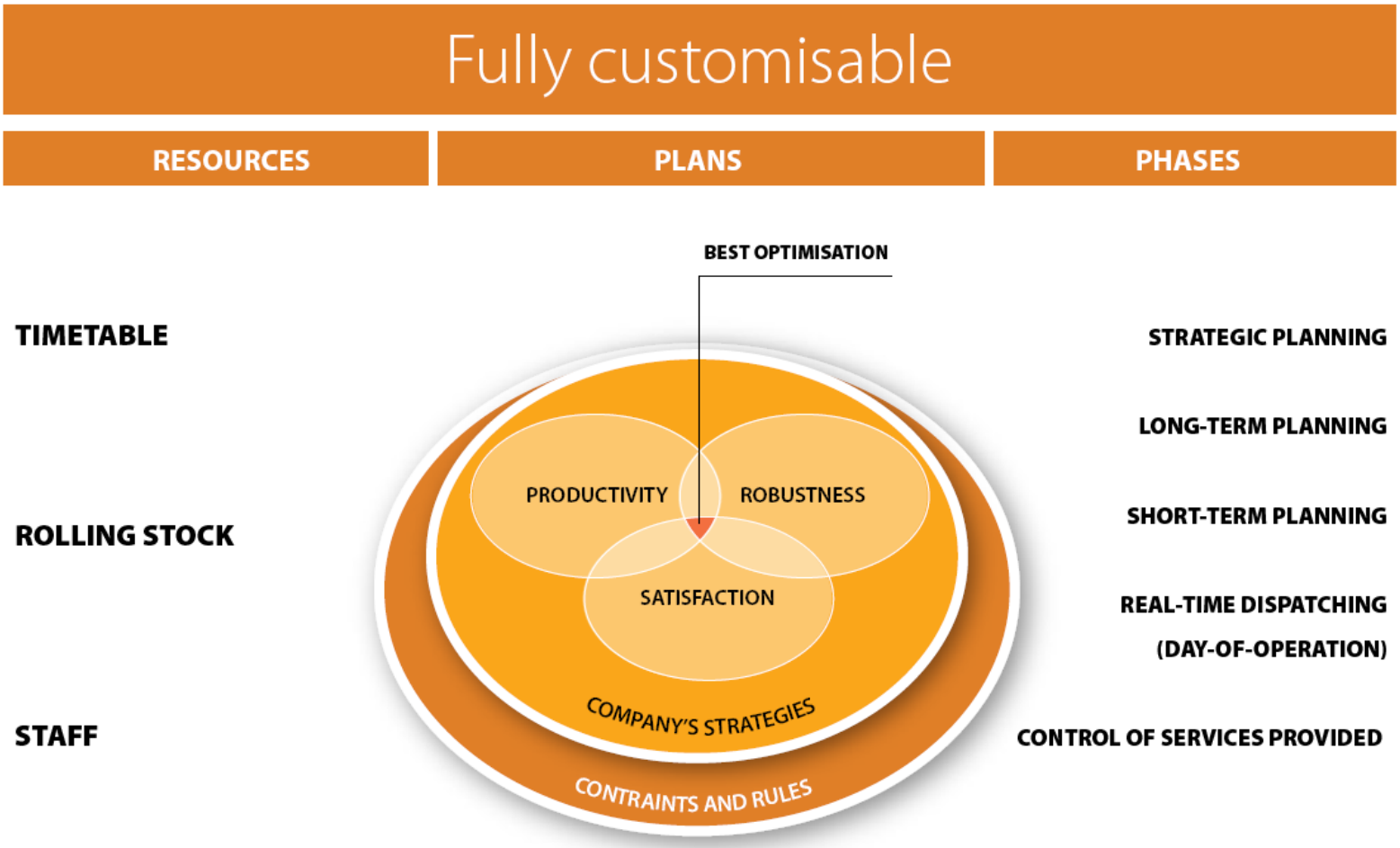


SISCOG
SISTEMAS COGNITIVOS



SISCOG
SISTEMAS COGNITIVOS

Global perspective



SISCOG
SISTEMAS COGNITIVOS

Products

ONTIME

Planning and Management of Timetables (Track & Time)

FLEET

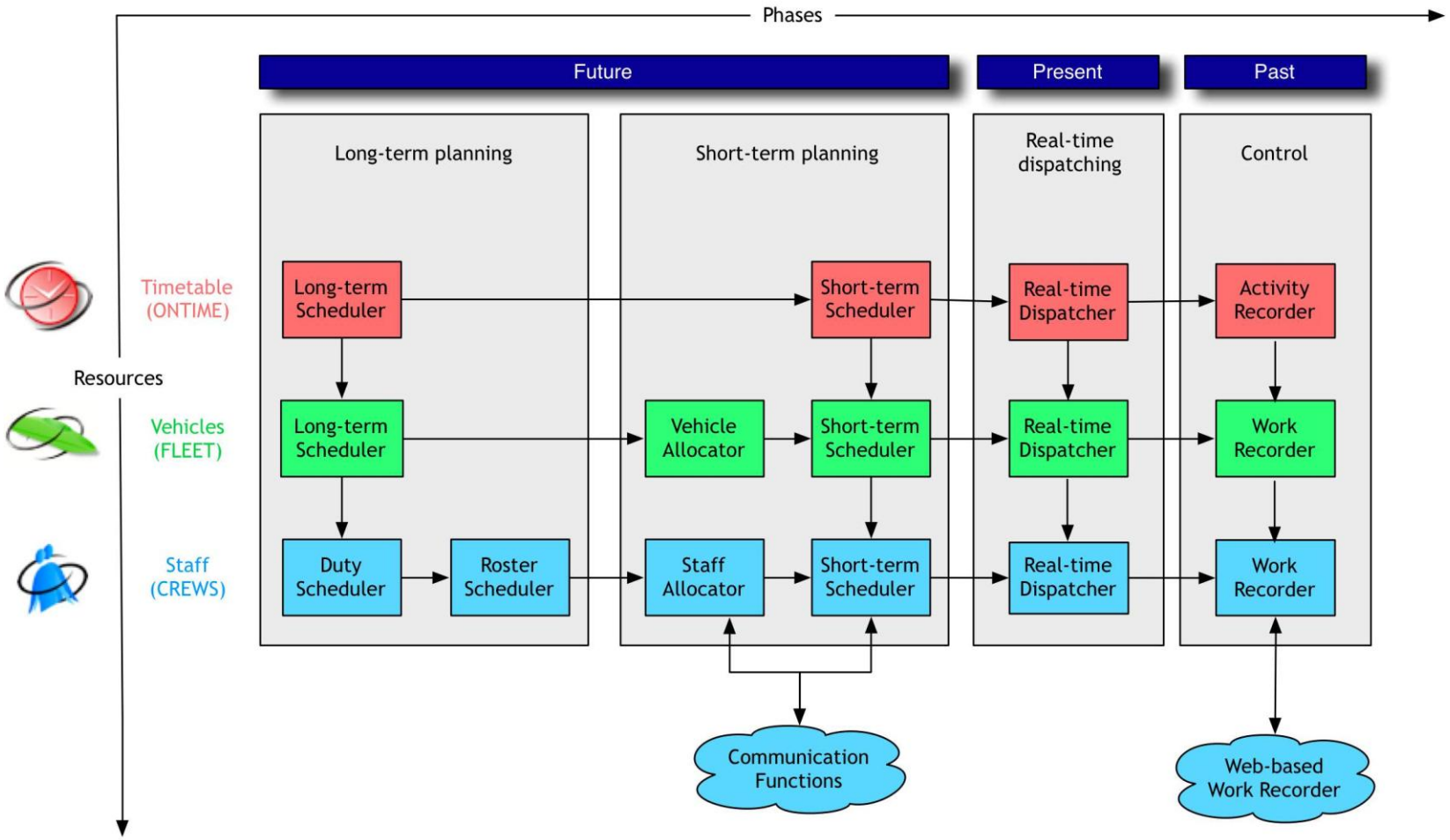
Planning and Management of Rolling Stock

CREWS

Planning and Management of Personnel

SISCOG
SISTEMAS COGNITIVOS

Complete integration, for all phases and resources



SISCOG
SISTEMAS COGNITIVOS

CREWS Demo

SISCOG
SISTEMAS COGNITIVOS

Case study: Dutch Railways (NS)

- Main train operator in Holland
- Ridership > 1M pass/day and > 5M pass/km/year
- 4,700 train trips per day
- Network with 2,350 km
- 5,200 crew members
  - 2,500 drivers
  - 2,700 guards
- Snow problems

Case study: Dutch Railways (NS)

- In production since 1998
- **Intensive use of optimisation** support
- 6% reduction in crew members (drivers and guards)
  - **12M€ annual savings**
- 60% reduction in planners
- Reduction in planning time
  - Few days vs. several months
- Only **15-20 minutes to react to accidents**
- Very **“social duties”**
  - Fair work assignments among workers
  - Non repetitive work

Case study: London Underground (LUL)

- Ridership > 3.3M pass/day
- Network with 400 Km
  - 11 separate lines
- Planned staff: > 3,000 drivers
- Before purchase
  - 100% manual planning
  - 13 weeks to compile a schedule
  - 15 planners needed
  - Training time: 3 to 5 years

SISCOG
SISTEMAS COGNITIVOS

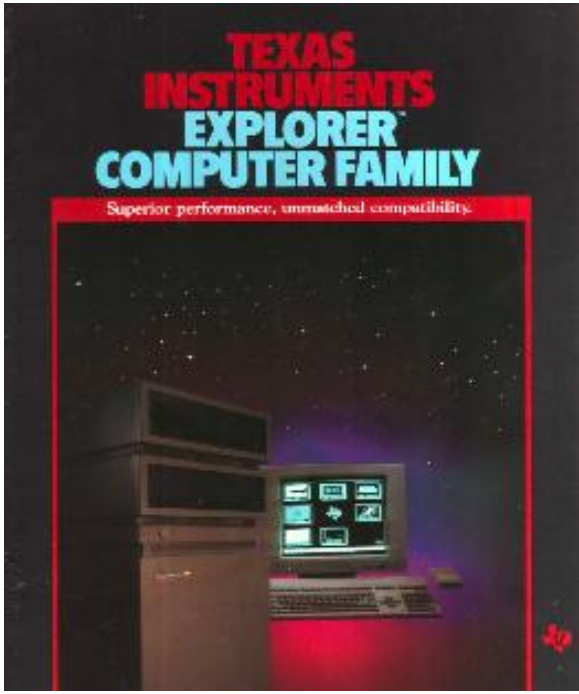
Case study: London Underground (LUL)

- In production since 2008
- Intensive use of optimisation support
- Northern Line
  - **Increase robustness** of plans, while keeping the same number of drivers
  - **Overtime reduced 90%**, from £1.000.000 down to £120.000 per year
  - Fewer delayed trains, happier passengers
- Piccadilly Line
  - **Reduced number of drivers**, while keeping the **robustness** of the plan
  - Increased schedule **efficiency** saves £200.000 per year
- London Olympics

Why Lisp?

- Founders were **PhDs on AI**
  - Learned Lisp (1978)
  - Worked on Knowledge Representation
- AI Boom
  - Expert Systems (1980)
  - AI-based companies
- Wanted to **tackle a difficult problem using AI**
  - Failure of OR to solve the difficult problems of transportation companies
  - **AI looked promising**
- **SIS**temas **COG**nitivos (Cognitive Systems)

Lisp Machines



- **21st century technology**
  - Effective garbage collection
  - Object-oriented environments
  - Windowing systems
  - Computer mice
  - High-resolution bit-mapped graphics
  - Computer graphic rendering
  - Networking innovations and protocols
  - Laser printing
  - Interactive programming environment
- **Partnership with Sperry** in the commercialisation in Portugal
- KEE was Lisp Machine specific
  - Fast development of expert systems

Hardware comes and goes, Lisp stays

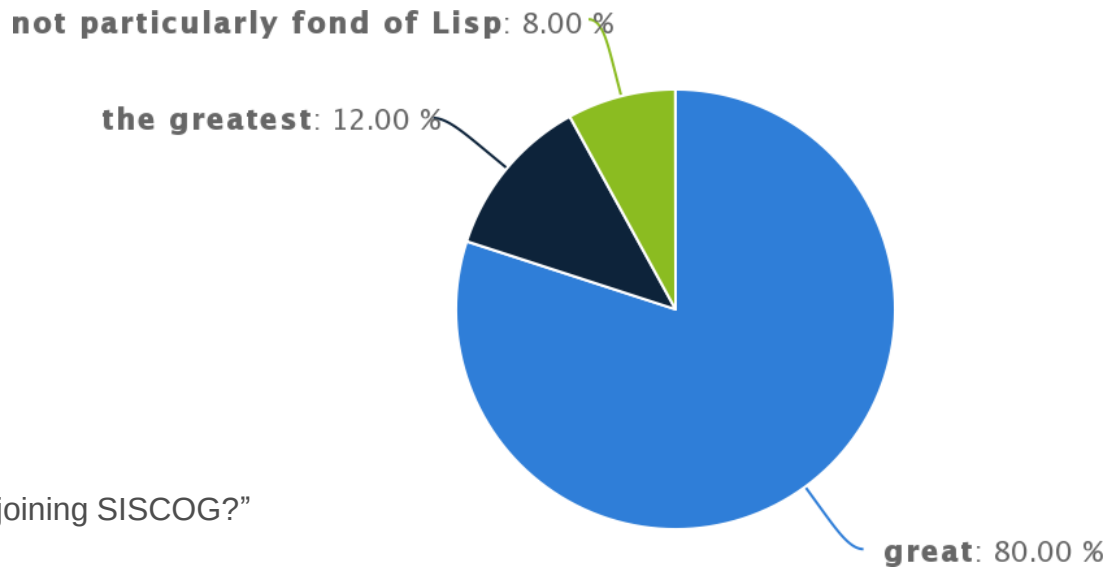
- **ZetaLisp** / TI Explorer, TI Explorer II (1986-1992)
  - Flavors
  - Implemented bits of ZetaLisp and Flavors on top of CL and PCL to ease porting
- **Lucid Common Lisp** / UNIX (1992)
  - For a while development still took place on the Explorers
  - Reused GUI on top of CLX
  - KEE abandoned in favour of SiKE
- **Allegro Common Lisp** / UNIX (1992-1997)
- **Allegro Common Lisp** / Windows (1997 onwards)
  - Common Graphics

Lisp-related anecdotes at SISCOG

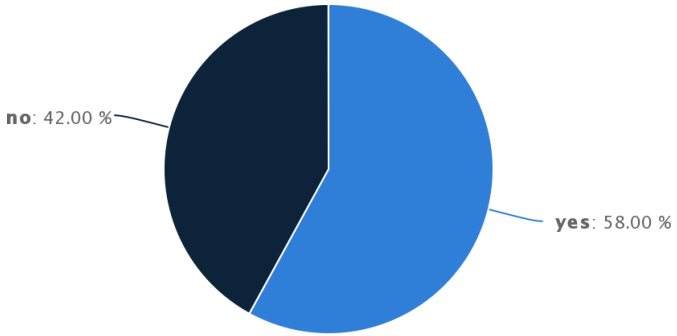
- Examples of how we use Lisp
- Factoids about SISCOG's Lispers

How Lisp is perceived at SISCOG

"Lisp: great language or the greatest language?"



"Did you know how to program in Lisp before joining SISCOG?"

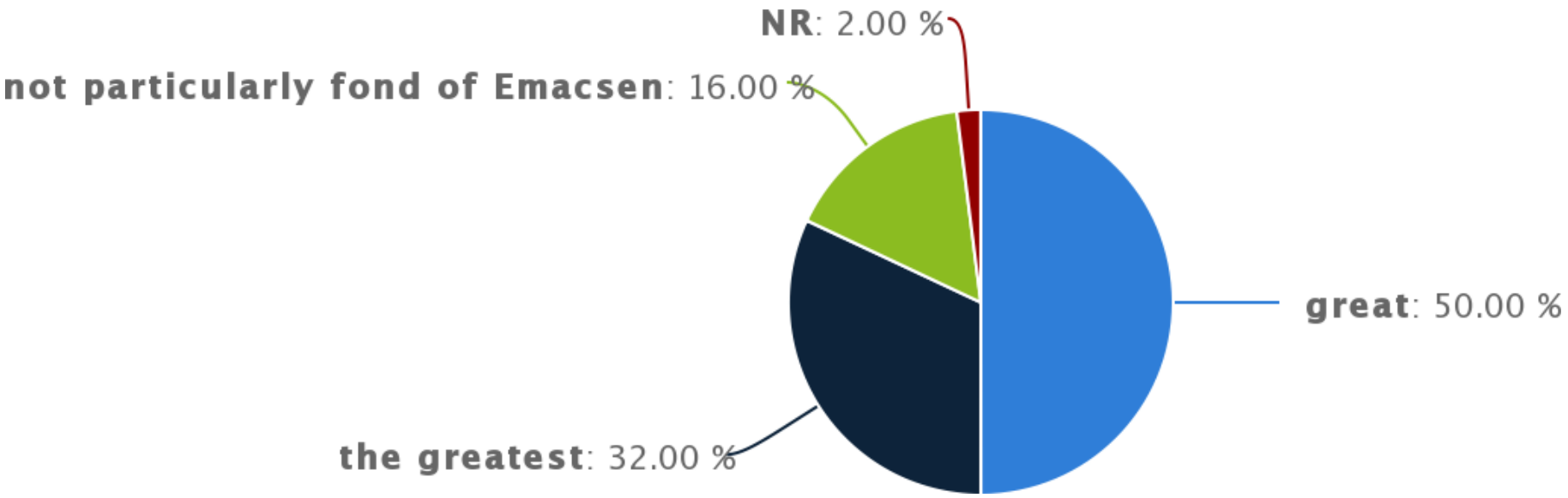


SISCOG
SISTEMAS COGNITIVOS

Interactive development and prototyping

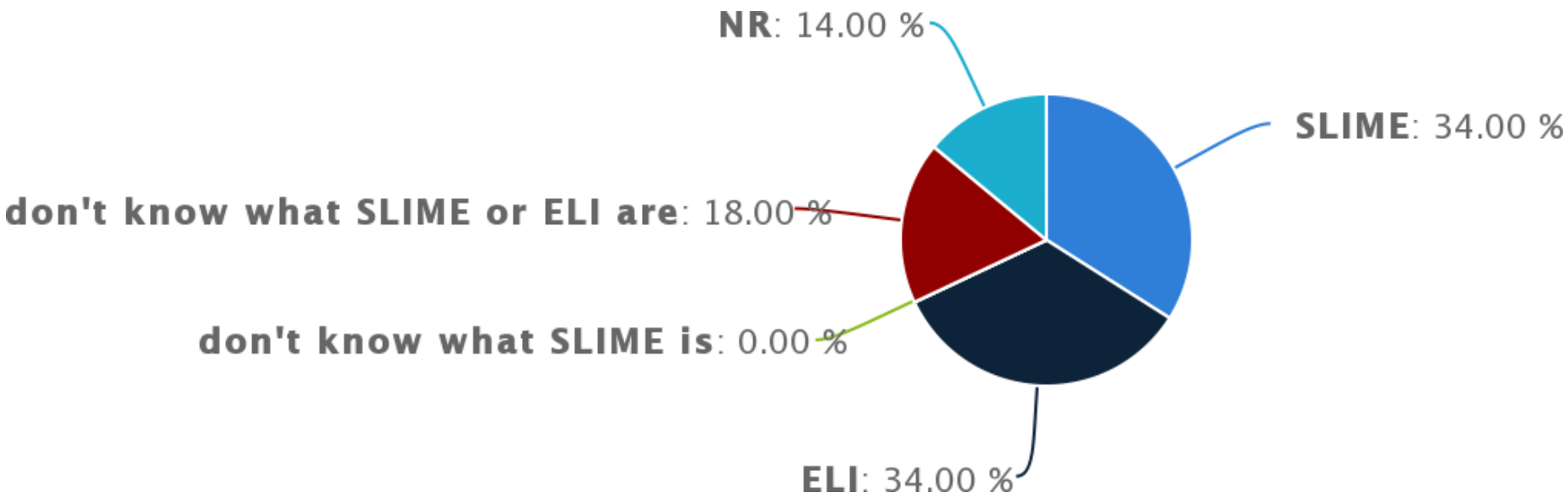
- **[Inspect]** + REPL manipulation
- **Incremental compilation**
  - Complex system, stays live for weeks during development
  - Hard time imagining how to do it any other way
  - Enables real-time bug fixing on systems with uptime demands running on the client
- Developing alongside the client
  - **Fast write/compile/test cycle** even more important if the client is staring at you
  - Optimises the limited time we can spend **1-on-1 with clients**
  - Leads to a **better product** at the end

“Emacs: great editor or the greatest editor?”



SISCOG
SISTEMAS COGNITIVOS

“SLIME or ELI?”



SISCOG
SISTEMAS COGNITIVOS

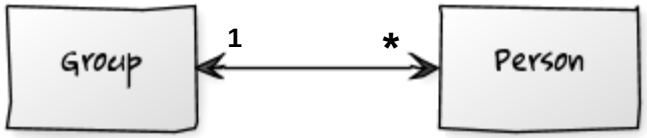
Emacs

- Everybody uses GNU Emacs
- **Patches**
  - Based on file/system dependencies
  - Move definitions to the appropriate patch via context menu
  - Straightforward sending of bug fixes to clients
  - Applications load patch files (fasls) during startup
- **Documentation and Change Log management**
  - Javadoc-style headers
  - Post-processed to HyperSpec-style manual

SiKE

- SISCOG's Knowledge Environment
- Implementation of important KEE features
  - Relations
  - Worlds
  - Interned objects
  - Other OO features mostly superseded by CLOS

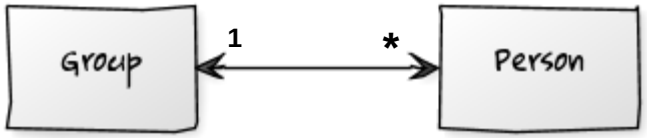
SiKE (example)




```
(sike:defclass group ()
  ((people :accessor people-of)))

(sike:defclass person ()
  ((name :accessor name-of)
   (group :accessor group-of)))
```

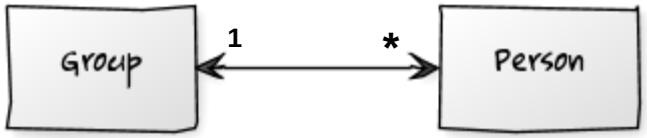
SiKE (example)




```
(sike:defclass group ()
  ((people :accessor people-of
           :relation (person group) :type :list)))

(sike:defclass person ()
  ((name :accessor name-of)
   (group :accessor group-of)))
```

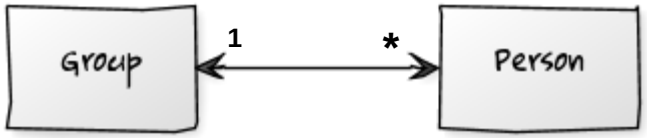
SiKE (example)




```
(sike:defclass group ()
  ((people :accessor people-of
           :relation (person group) :type :list)))

(sike:defclass person ()
  ((name :accessor name-of)
   (group :accessor group-of
          :relation (group people) :type :atom)))
```

SiKE (example)




```
(sike:defclass group ()
  ((people :accessor people-of
           :relation (person group) :type :list
           :worlds t)))

(sike:defclass person ()
  ((name :accessor name-of
         :worlds t)
   (group :accessor group-of
          :relation (group people) :type :atom
          :worlds t)))
```

SiKE (example)

```
(defparameter *the-beatles* (make-instance ‘group))

(defparameter *the-60s* (make-instance ‘sike:world))

(within-world *the-60s*
  (dolist (name ‘(“Paul” “George” “John” “Ringo”))
    (make-instance ‘person :name name :group *the-beatles*))
  (people-of *the-beatles*))
⇒(#<P “Paul”> #<P “George”> #<P “John”> #<P “Ringo”>)
```

SiKE (example)

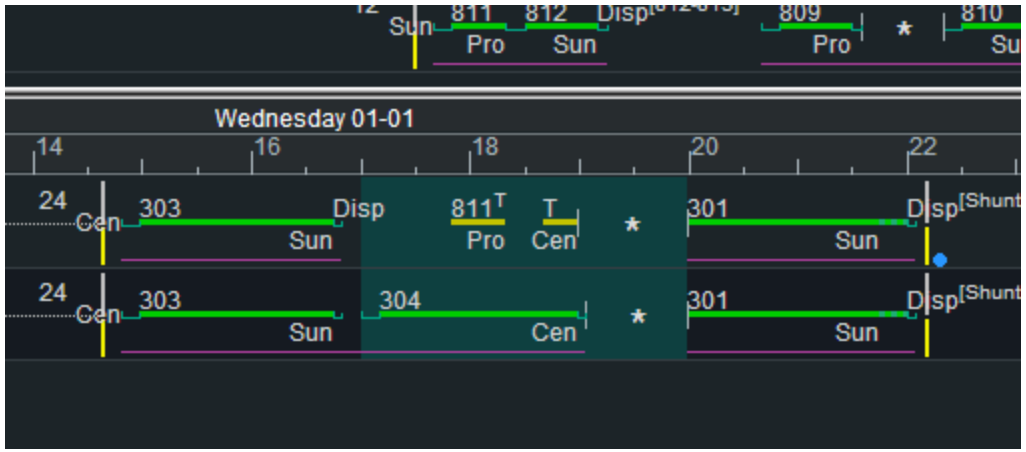
```
(defparameter *today*
  (make-instance ‘sike:world :parent *the-60s*)))

(within-world *today*
  (setf (group-of *john*) nil)
  (setf (group-of *george*) nil)
  (people-of *the-beatles*))
=> (#<P “Paul”> #<P “Ringo”>)

(within-world *the-60s*
  (people-of *the-beatles*))
=> (#<P “John”> #<P “George”> #<P “Paul”> #<P “Ringo”>)
```

SiKE (application)

Same object (EQ)



← World A

← World B

SISCOG
SISTEMAS COGNITIVOS

JavaScript Bridge

- Motivation: extend GUI with HTML and JavaScript
  - Seamless for the end-user
  - Nice widgets like jQuery UI accordions (collapsible areas)
- Bidirectional communication
- Attached to IE, Windows and Common Graphics

JavaScript Bridge (Lisp side)

- Motivation: update content without full refresh
- Evaluate JS from Lisp

```
(js-eval widget (ps (alert "Hello JS World!")))
```

- Poking JS objects from Lisp

```
var x = { foo: 0,
          getFoo: function () { return this.foo; },
          setFoo: function (v) { this.foo = v; } };

(let ((x (js-eval widget "x")))
  (ole:auto-method x "getFoo")      ; => 0
  (ole:auto-method x "setFoo" 1)
  (ole:auto-getf x "foo")           ; => 1
  (setf (ole:auto-getf "foo") 42)) ; x.foo === 42
```

- Maybe in the future: more sophisticated proxy objects via the MOP

JavaScript Bridge (JS side)

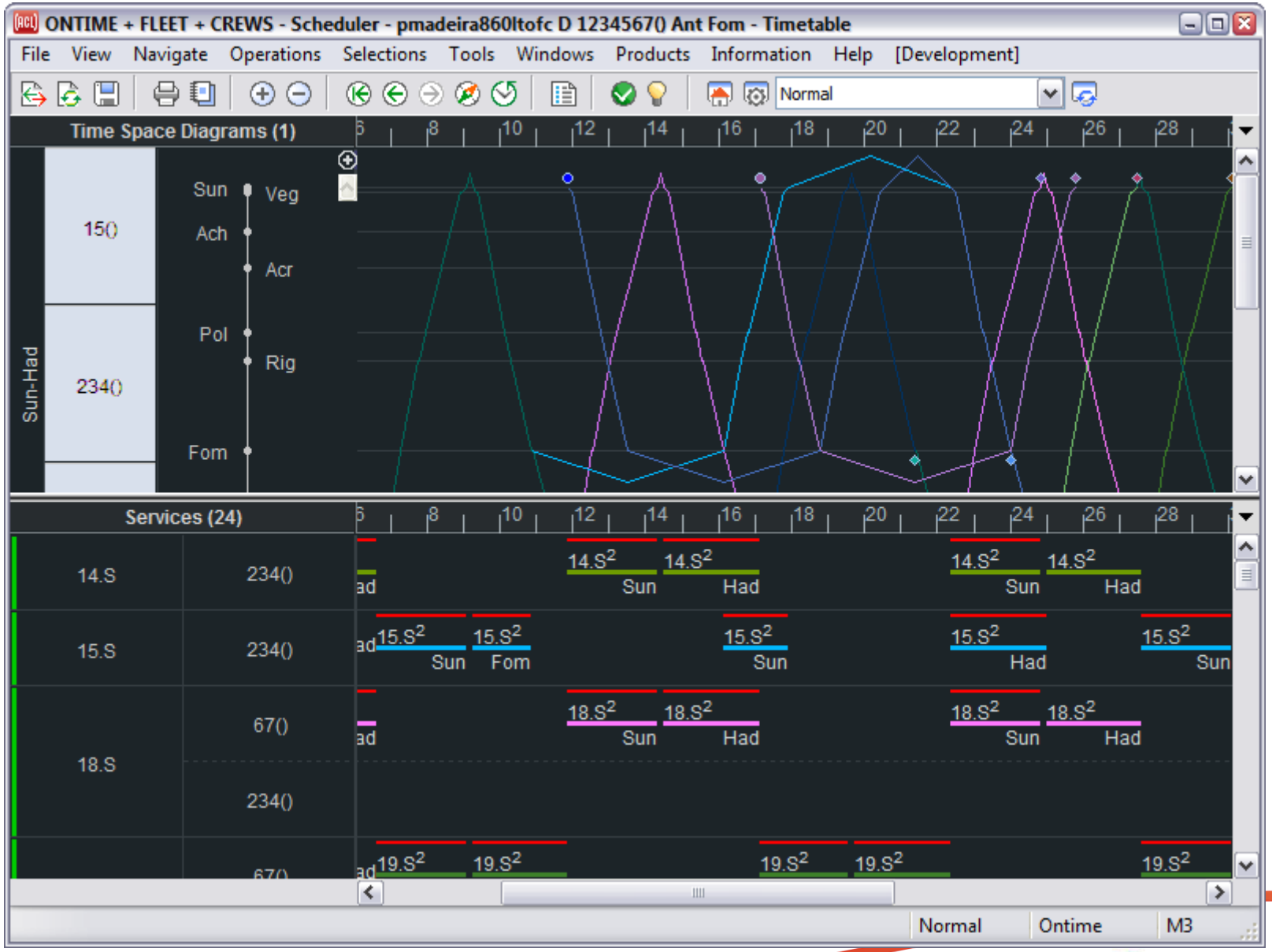
- Motivation: get/set printing options, apply data filters, jump to objects
- Evaluate Lisp from JS

```
window.external.readEval('(cg:message "Hello World!")');
```

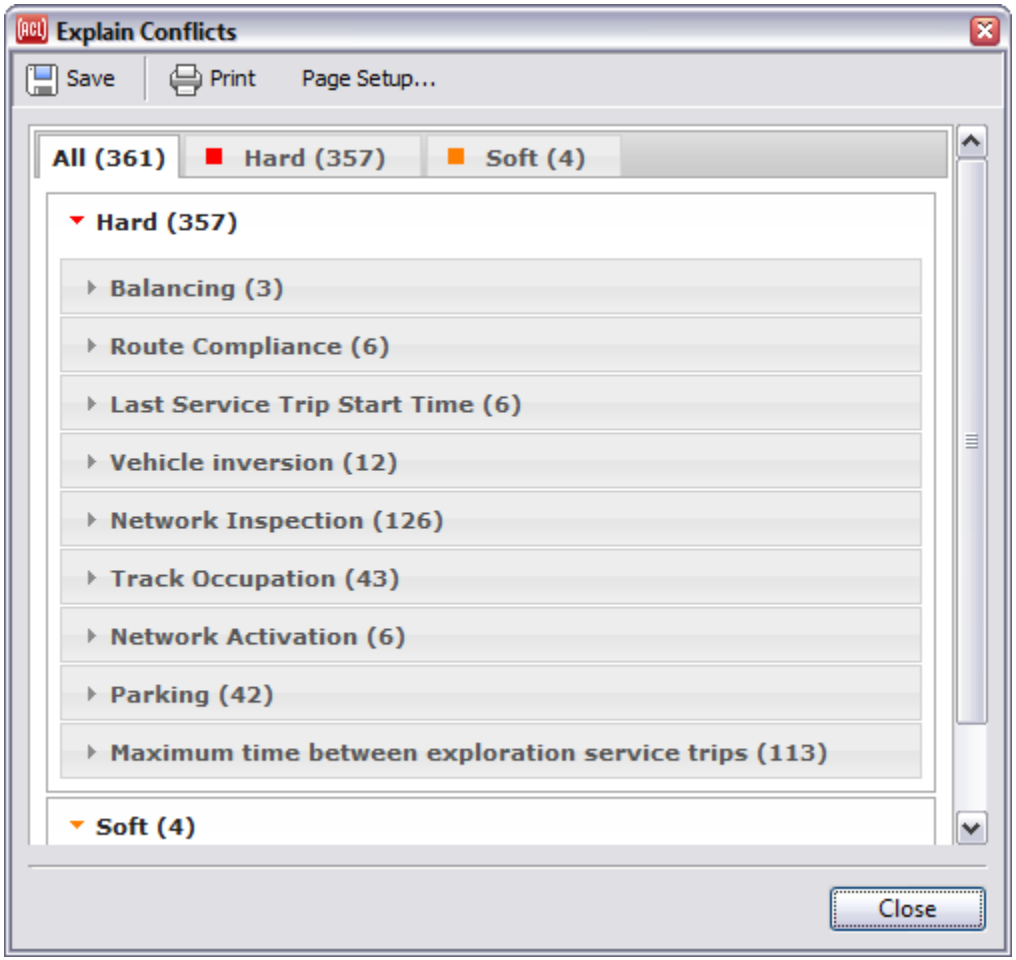
- HTML widget holds an easily referenceable object

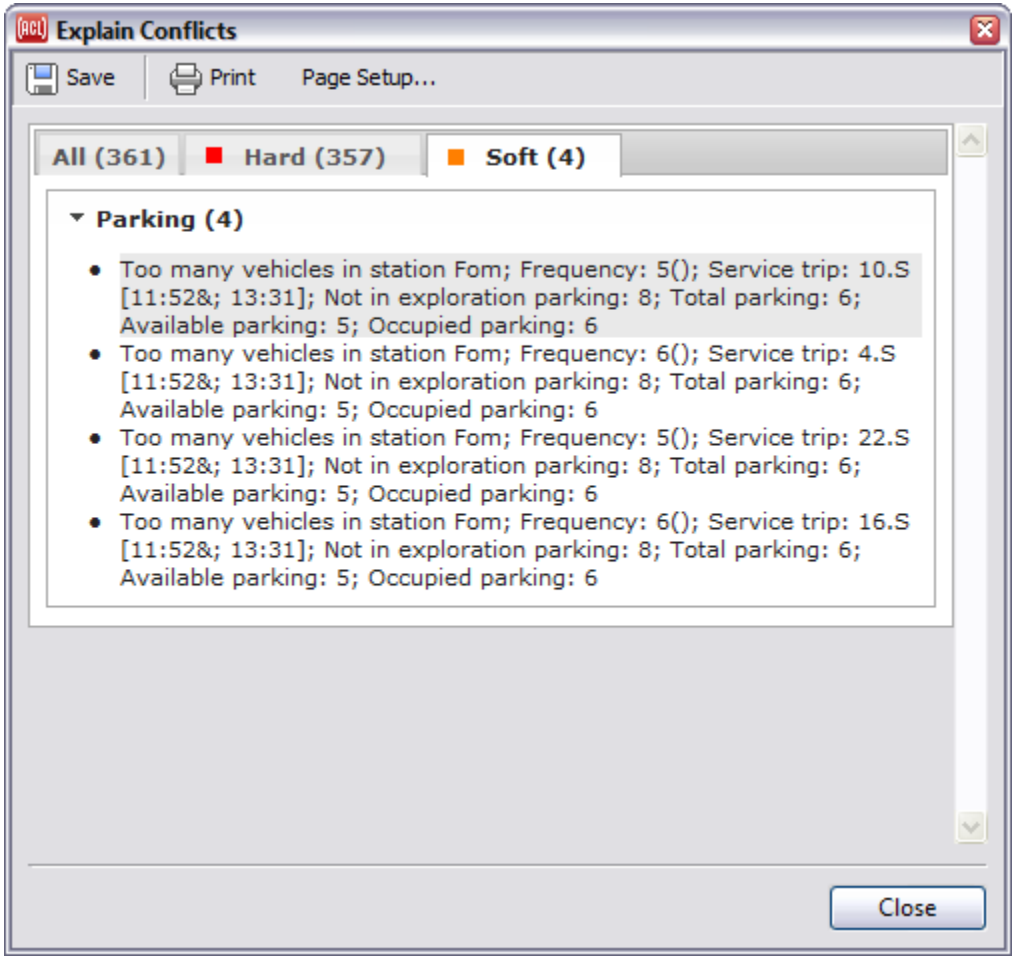
```
var result = window.external.readEval('(frob *object*)');
```

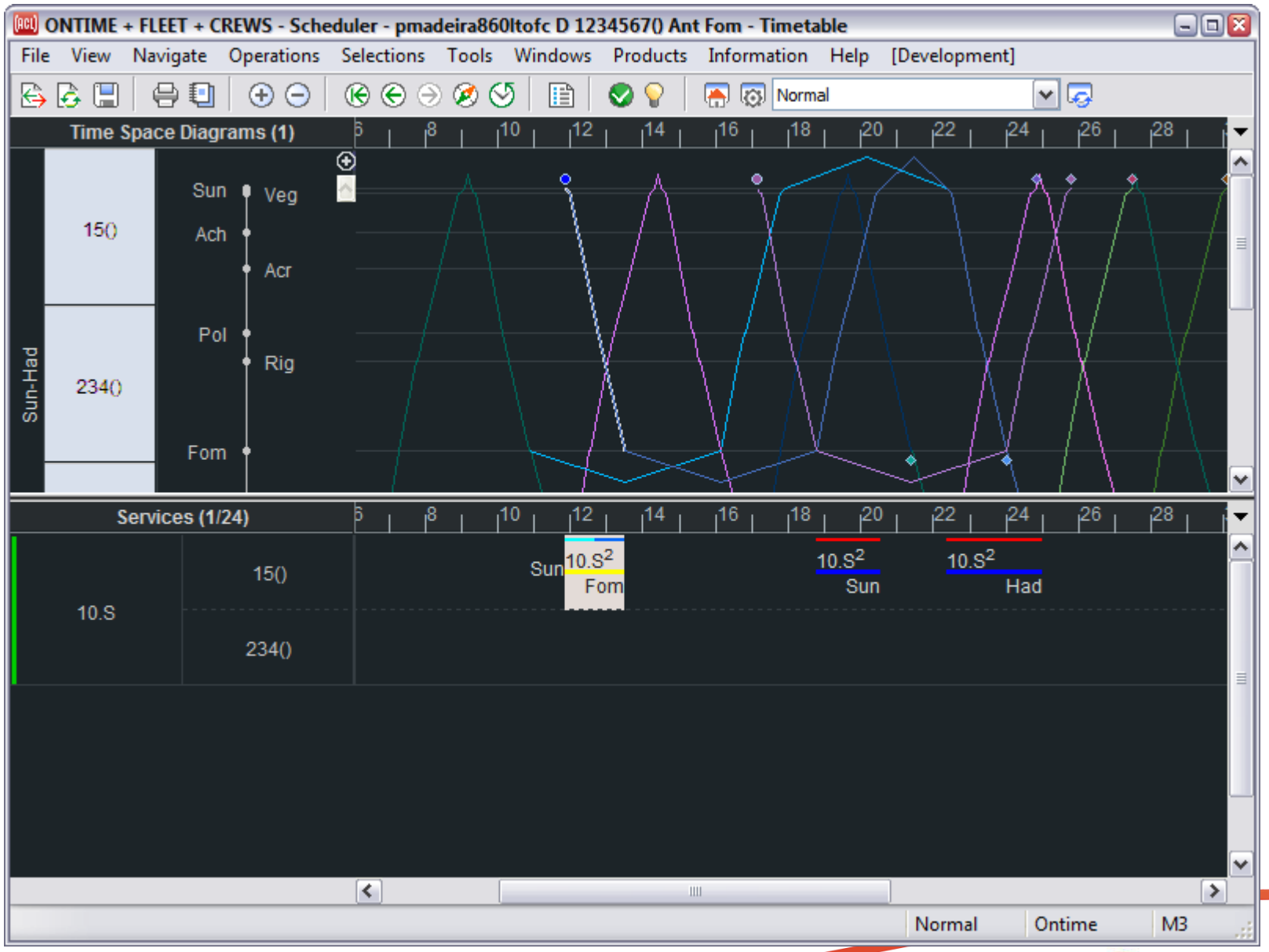
- Maybe in the future, make Lisp objects scriptable
  - Wrap them in an IDispatch object
  - Convert `<obj>.getFoo(...)` to `(get-foo <obj> ...)`



SISCOG
SISTEMAS COGNITIVOS


Explain Conflicts
Save
Print
Page Setup...
All (361)
Hard (357)
Soft (4)
Hard (357)
Balancing (3)
Route Compliance (6)
Last Service Trip Start Time (6)
Vehicle inversion (12)
Network Inspection (126)
Track Occupation (43)
Network Activation (6)
Parking (42)
Maximum time between exploration service trips (113)
Soft (4)
Close

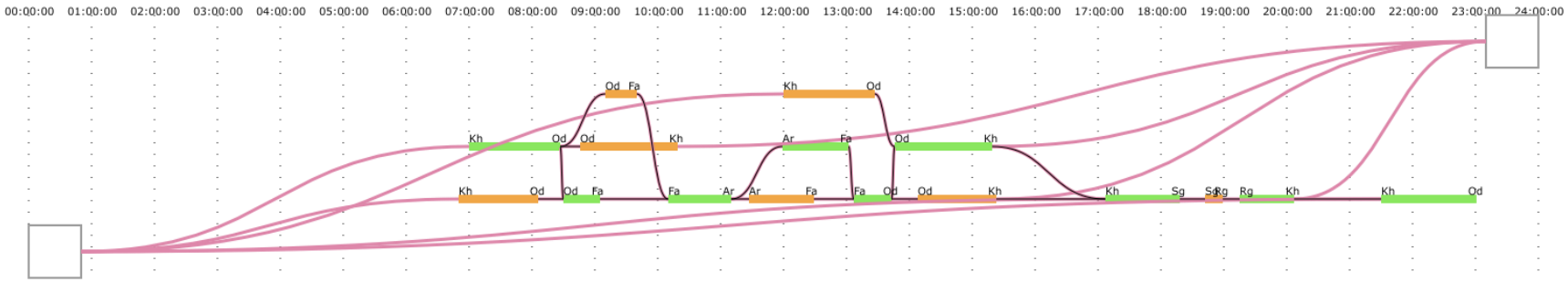

Explain Conflicts
Save
Print
Page Setup...
All (361)
Hard (357)
Soft (4)
Parking (4)
Too many vehicles in station Fom; Frequency: 5(); Service trip: 10.S [11:52&; 13:31]; Not in exploration parking: 8; Total parking: 6; Available parking: 5; Occupied parking: 6
Too many vehicles in station Fom; Frequency: 6(); Service trip: 4.S [11:52&; 13:31]; Not in exploration parking: 8; Total parking: 6; Available parking: 5; Occupied parking: 6
Too many vehicles in station Fom; Frequency: 5(); Service trip: 22.S [11:52&; 13:31]; Not in exploration parking: 8; Total parking: 6; Available parking: 5; Occupied parking: 6
Too many vehicles in station Fom; Frequency: 6(); Service trip: 16.S [11:52&; 13:31]; Not in exploration parking: 8; Total parking: 6; Available parking: 5; Occupied parking: 6
Close



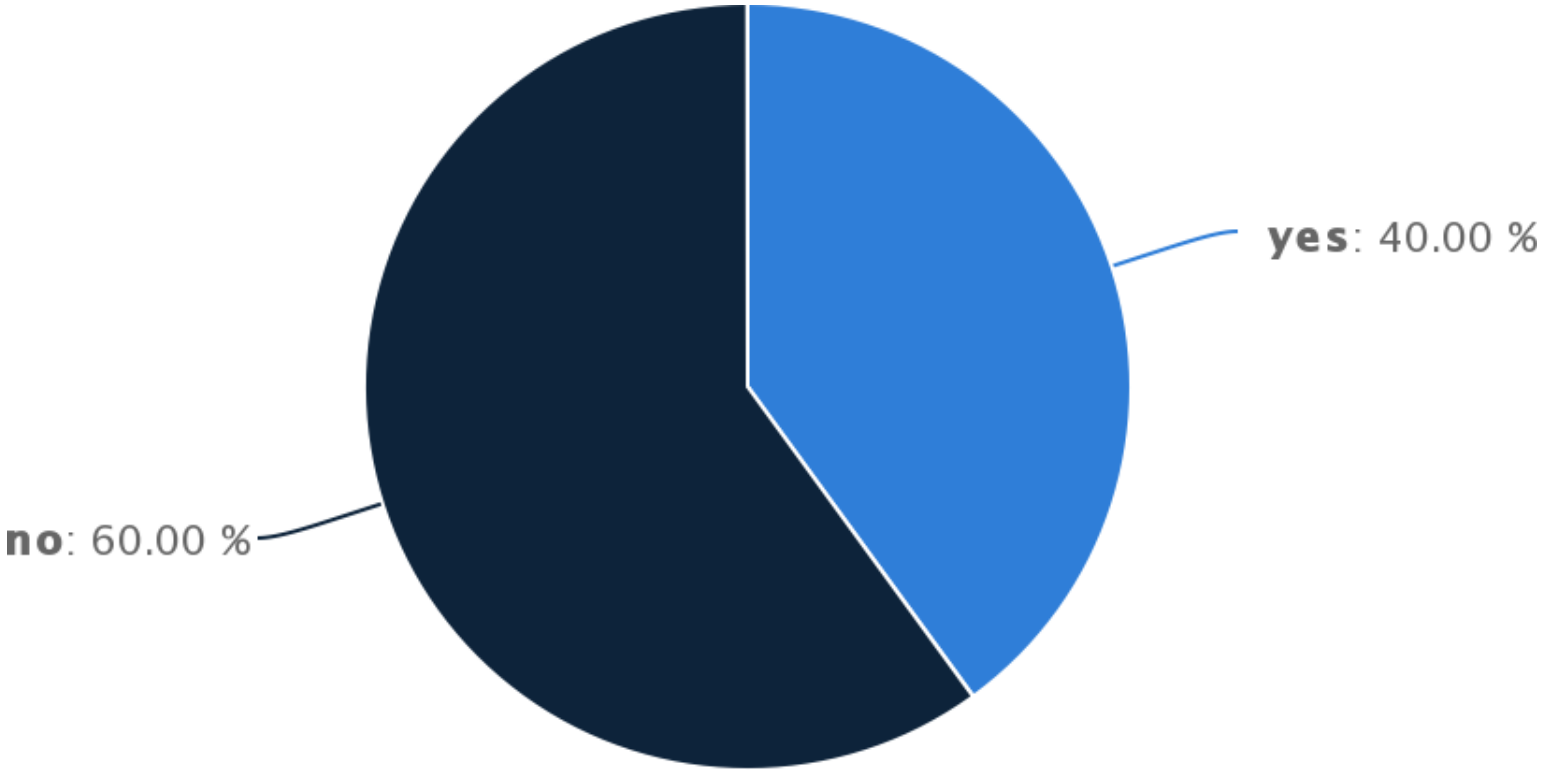
SISCOG
SISTEMAS COGNITIVOS

Open-Source Involvement

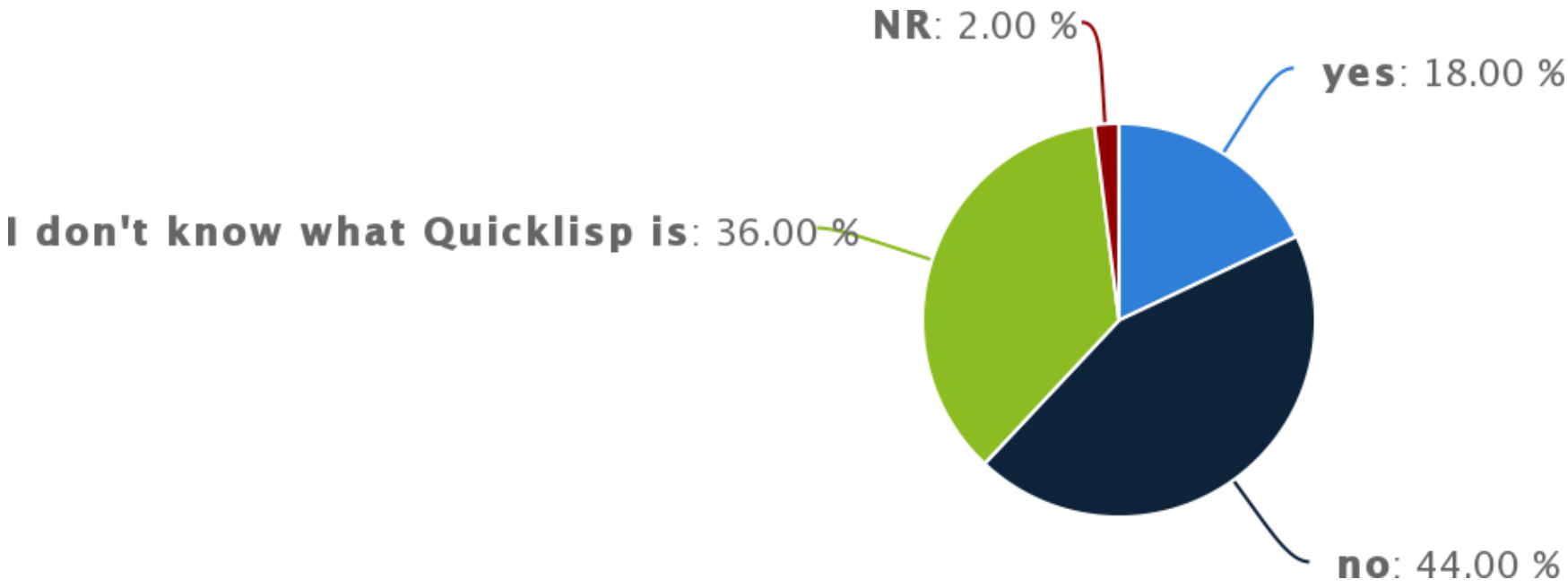
- SISCOG has traditionally used **commercial Lisps** along with their **commercial libraries**
  - Notable exceptions: Emacs and SLIME
- **Increasing use of OSS** in in-house projects and prototypes
  - ASDF, Alexandria, Vecto, Stefil, CL-Graph
  - YASON, CL-WHO, CL-HTML-DIFF



“Have you ever used an open-source Lisp library?”



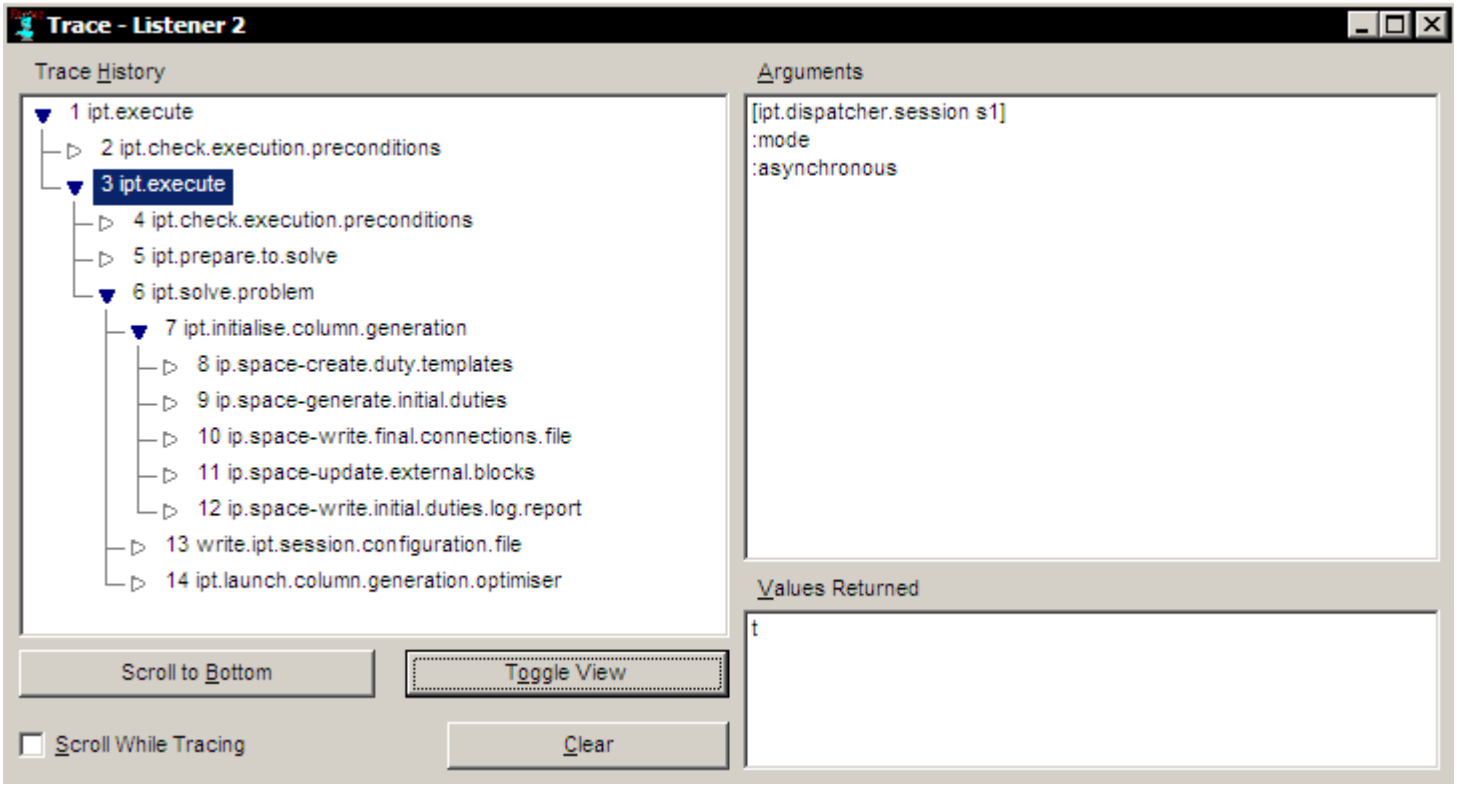
“Have you ever used Quicklisp?”



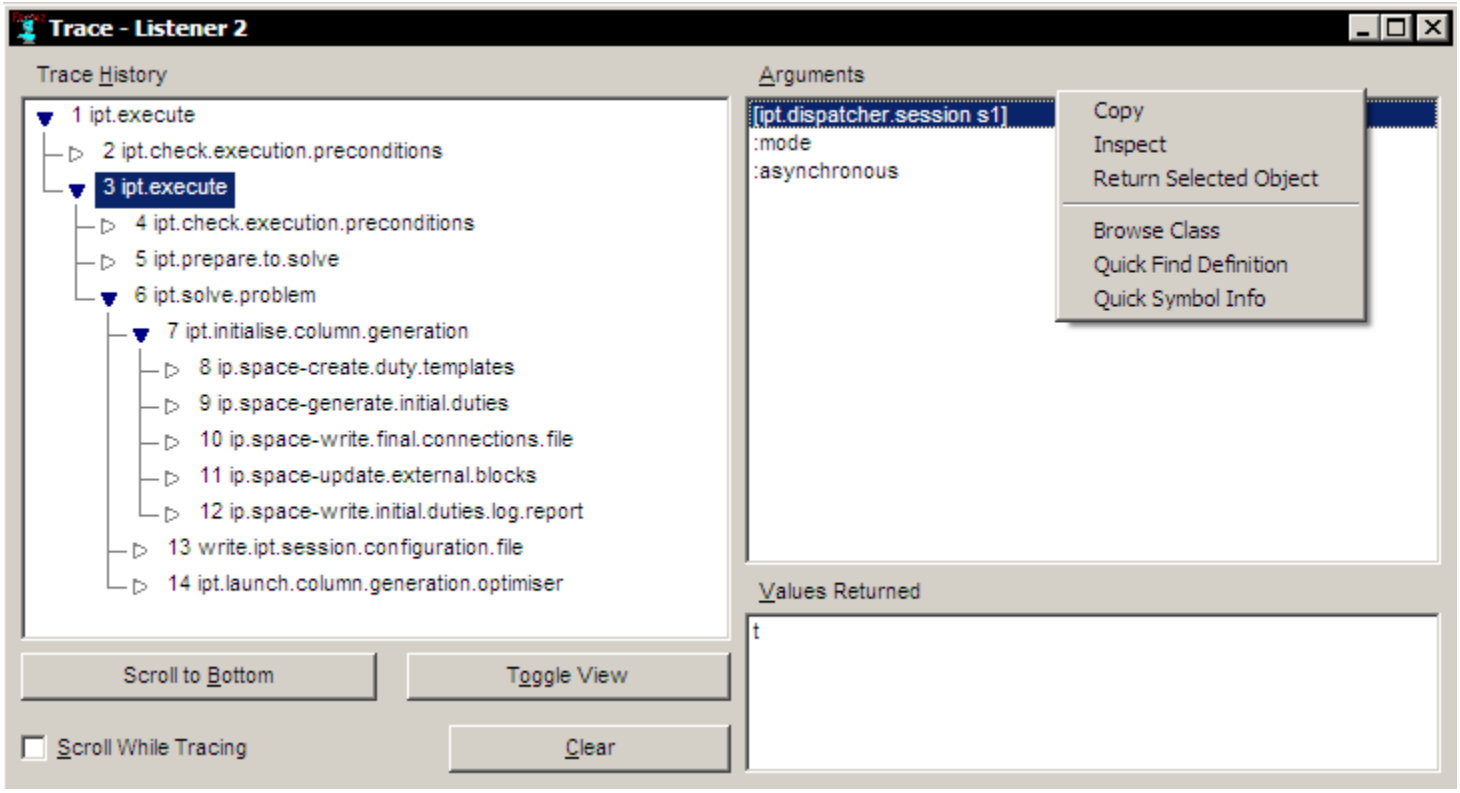
SISCOG
SISTEMAS COGNITIVOS

Things we like about ACL

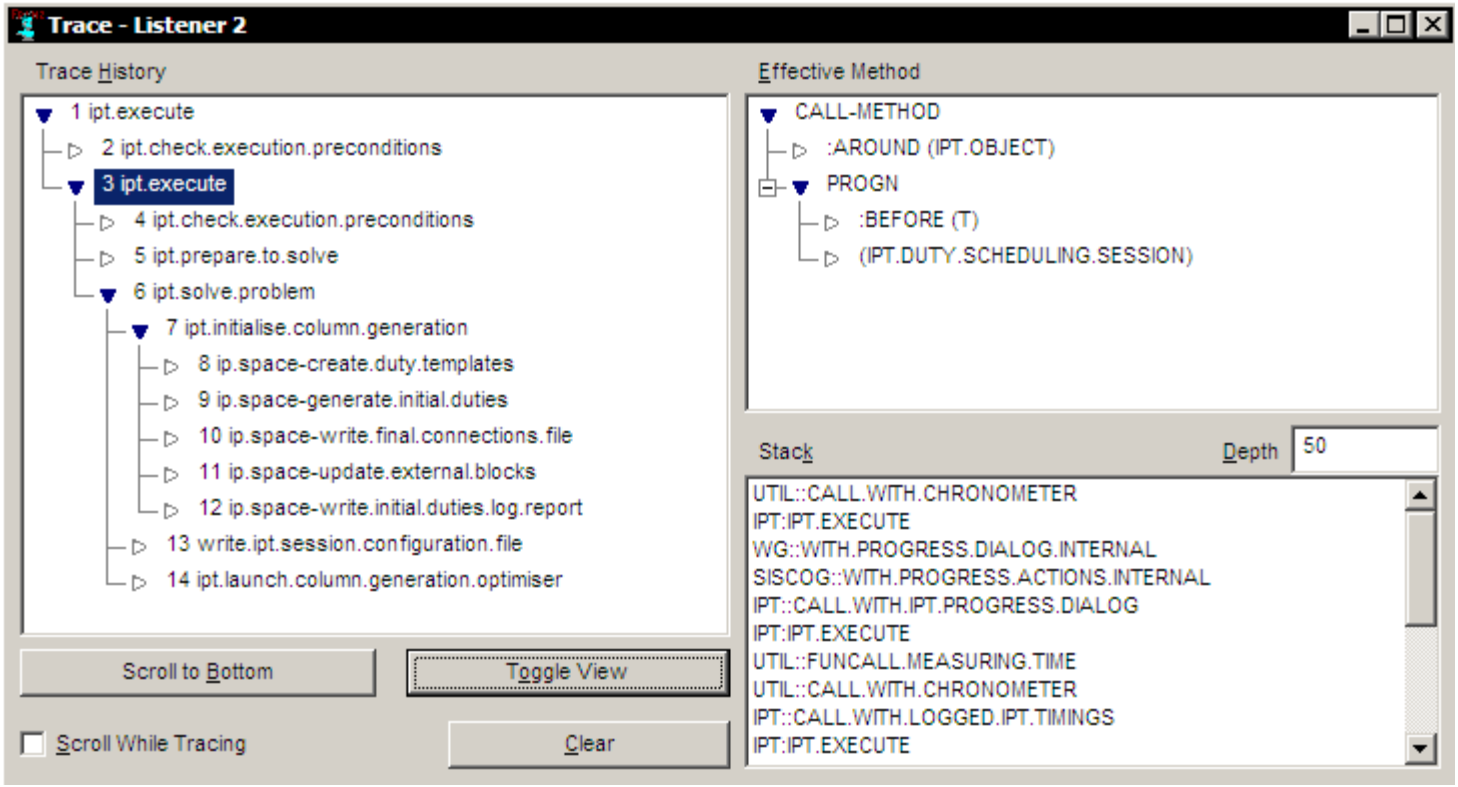
- IDE: trace dialog, profiler, class browser
- Fast compiler
  - Fast code too, given sufficient hand-holding
  - Downside: very few warnings
- GC fast enough for GUI
- Multiprocessing (SMP)
  - Took a while, but it’s here!
  - GC might become an issue due to Amdahl’s law



SISCOG
SISTEMAS COGNITIVOS

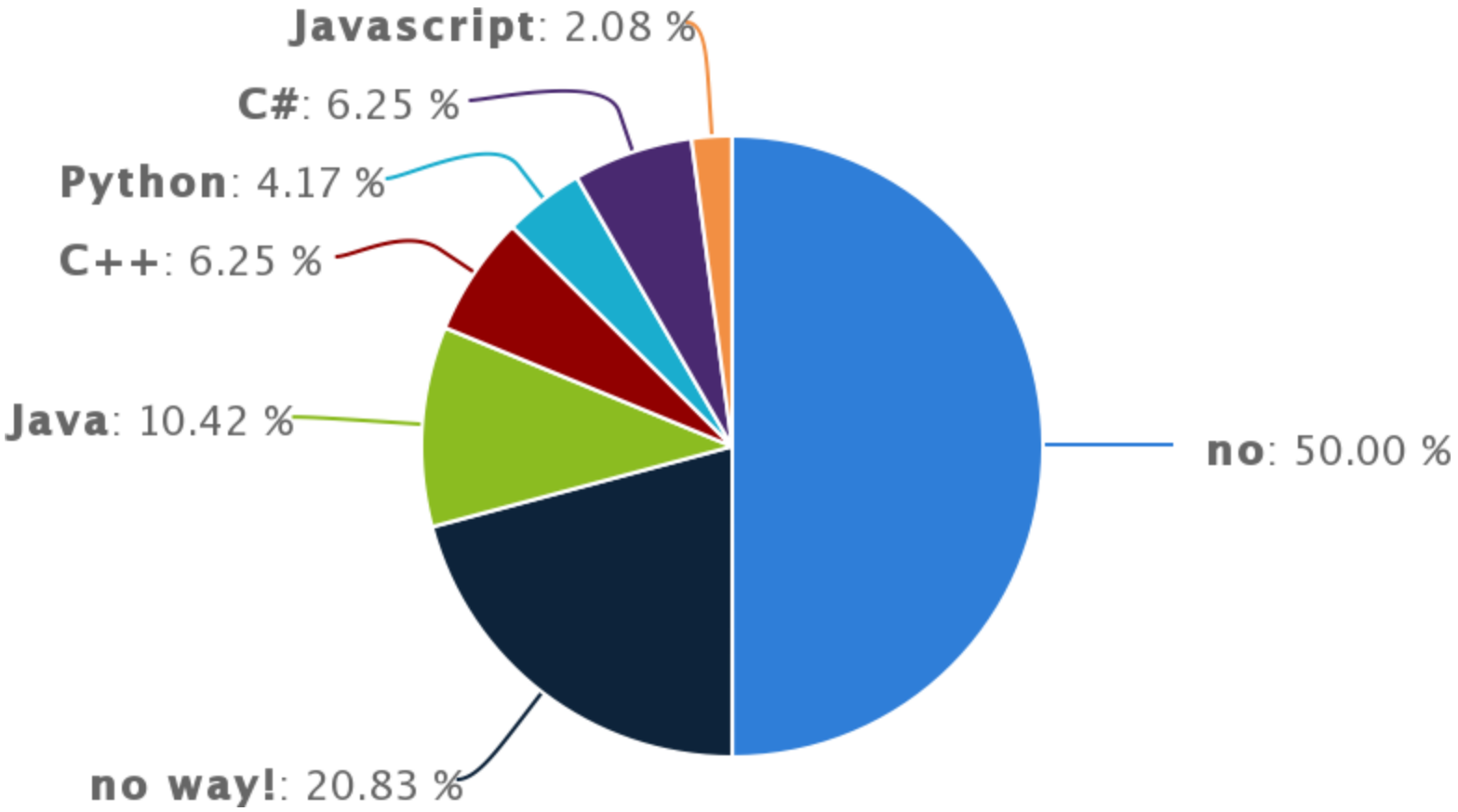


SISCOG
SISTEMAS COGNITIVOS



SISCOG
SISTEMAS COGNITIVOS

“Would you rather use something other than Lisp?”



SISCOG
SISTEMAS COGNITIVOS

Competition with C++

- **OR optimiser** written in C++ for historical reasons
  - “Floating-point operations in Lisp are too slow!”
- Recent experiments demonstrated **boxing** can be avoided
  - Simple arrays of floats
  - Floating-point intensive bits of code contained in one function
- Critical parts of the process **migrating to Lisp**
- Some modules with **C-like performance** and others with **CLOS flexibility**

Conclusions

- Lisp has been giving SISCOG a 21st century development environment for 27 years
- Even today, how many mature languages provide Lisp's flexibility, performance and interactivity?
- Developers are happy
- Clients are happy
- More chapters to come… Written in Lisp

Thank you

Obrigado

SISCOG
SISTEMAS COGNITIVOS