LISP DEPLOYMENT

Kyle Kerslake
RavenPack

Our Problems

- Large applications
- Lots of required data
- Subtle variations of these applications
- Long startup times (data loading)

Build + Run from Source

- Syncing repositories is easy
- Git pull and reload to hotpatch (easy)

- Full process takes over 20 mins

Dump an image!

- Startup down to 2 minutes
- One file to rule them all

- File is 7gb large (hard to move)
- Requires deployment targets to be set up
- Care needed for stored pathnames

Lisp images in Docker images

- Infrastructure can handle big images
- Deployment is almost as simple as source
- Base Lisp container means no Lisp setup

- Hotpatching is more awkward
- Additional tool to learn

Lisp in the middle of web browsing

Michael Raskin, raskin@mccme.ru

Dept. of CS, TU Munich

April 2, 2019

In the previous installments

Controlling everything with Lisp is great

Lisp machines were tools of saner times

Web and hardware are horrible
Previous years: how to control hardware drivers

This time: web in practice

Thoughtful theridion

Works on web
(as every theridion — cobweb spider — is expected to)

Wraps tools like CL-HTML5-Parser and Drakma and CSS-Selectors

Thinks before acting:
Redefinable policies for everything via CLOS

State of web

They took a web of documents and killed it!
Web of applications — but snapshots still work
Save everything **before** reading!

Javascript and styles are not really needed most of the time
Designers ask for semantic markup?
Facebook requires metadata for good preview?
Let's use the results

Stop looking for updates by eyes — process article history automatically using only human-oriented pages
(and download archives if desired)

Crawling DSL

```
(with-page (x "https://planet.lisp.org/")
   "div#content > p > a"
   ("href" x)
   (with-page (x x :recurse recurse)
      (let prev "tr:first-child > td > a"
        ("href" x))
      (page-walker-brancher-progn
        (with-page (x x :fetch nil :keep-brancher t)
           "li > a"
           ("href" x)
           (if (cl-ppcre:scan "/201[0-8]/" x)
             *page-walker-terminator* x))
        (recurse prev))))
```

Crawling DSL

```
(with-page (x "https://planet.lisp.org/")
   "div#content > p > a"
   ("href" x)
   (with-page (x x :recurse recurse)
      (let prev "tr:first-child > td > a"
        ("href" x))
      (page-walker-brancher-progn
        (with-page (x x :fetch nil :keep-brancher t)
           "li > a"
           ("href" x)
           (if (cl-ppcre:scan "/201[0-8]/" x)
             *page-walker-terminator* x))
        (recurse prev))))
```

https://gitlab.common-lisp.net/mraskin/thoughtful-theridion

What's new with Clasp & Cando?

Martin Cracauer <cracauer@thirdlaw.tech>

European Lisp Symposium 2019
Genoa, Italy

Clasp/Cando support organization

Christian Schafmeister and Martin Cracauer founded a company:

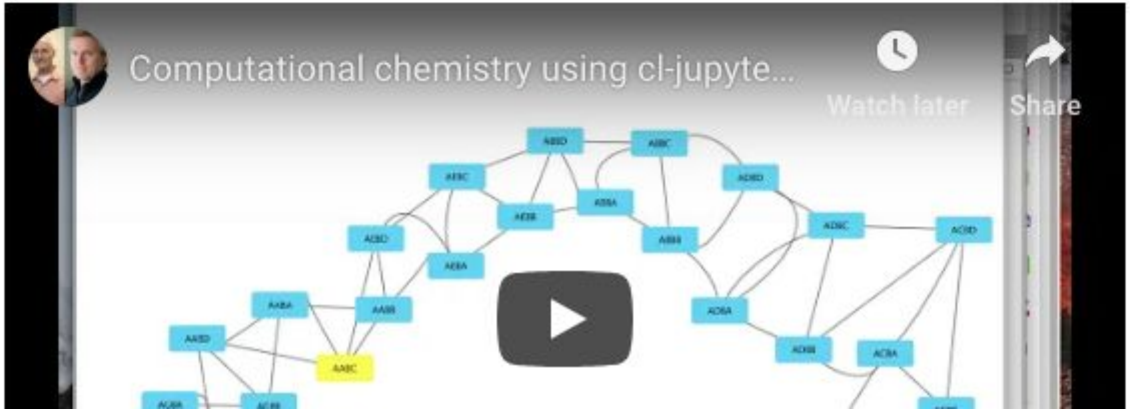
ThirdLaw LLC
technologies

- ThirdLaw Technologies www.thirdlaw.tech
- With Temple University and DOE
- Paid support for open source Clasp and Cando
- Commercial modules (e.g. Free Energy Perturbation)
- Distributed computing framework to execute Cando output directories for molecular dynamics package (e.g. Amber) on distributed computing fleets
- Emphasis on security
- Research to optimize cost and/or latency for those computing jobs

New youtube demos!

www.thirdlaw.tech

Demo: free energy perturbation in Cando using cl-jupyter based notebooks to blend graphical user interface and language based control and reusability.



Clasp infrastructure and accessibility progress

- Progress in Clasp startup time
- Lots of general Lisp correctness patches (thanks Karsten Poeck)
- Progress in compile time (Clasp 30 min, Cando+quicklisp 40 min)
- Progress in debug info
- Buildbot ensures branches stay buildable. -ish
- Officially supported: Linux w/LLVM6 (e.g. Debian Stretch w/backports, Ubuntu 18.04), OSX/brew, OSX/macports, FreeBSD port
- Contiguous update of Docker image: `docker run … cracauer/cando`

Linux users can also use the Docker image to extract the binary /opt/clasp or use it as a chroot instead of Docker.

Clasp, Cando and cl-Jupyter

- Jupyter notebooks are the official GUI for Cando
- Our hacked up cl-jupyter is on github/clasp-developers
- Our buildbot builds the whole stack of Clasp, Cando, cl-jupyter and all quicklisp modules required by that, and puts it into Docker
- Can be used by anybody who just wants to use cl-jupyter, not specific to Cando
- Anyone wants to backport it to general Cl/quicklisp? Right now a bit Clasp-specific

Docker image can be used to unpack /opt/clasp, which has it all.

Open source used in Clasp/Cando

Clasp doesn’t go on crusades to reinvent wheels. Partial list:

- SICL
- LLVM (all code generation goes through LLVM optimization)
- Boehm GC and MPS (open-source-ish)
- CL-jupyter and various graphics packages for web from quicklisp
- Alexandria
- Closer-mop
- Concrete syntax tree

Distributed computing for molecular dynamics

- Not specific to molecular dynamic, can run anything that is data-in-disk-files oriented
- Focus on security (single user keys, proper keychain, proper isolation of individuals, no network based security part of scheme)
- Optimizations will be in Lisp
- Many subjobs are Lisp jobs (most are GPU computing)

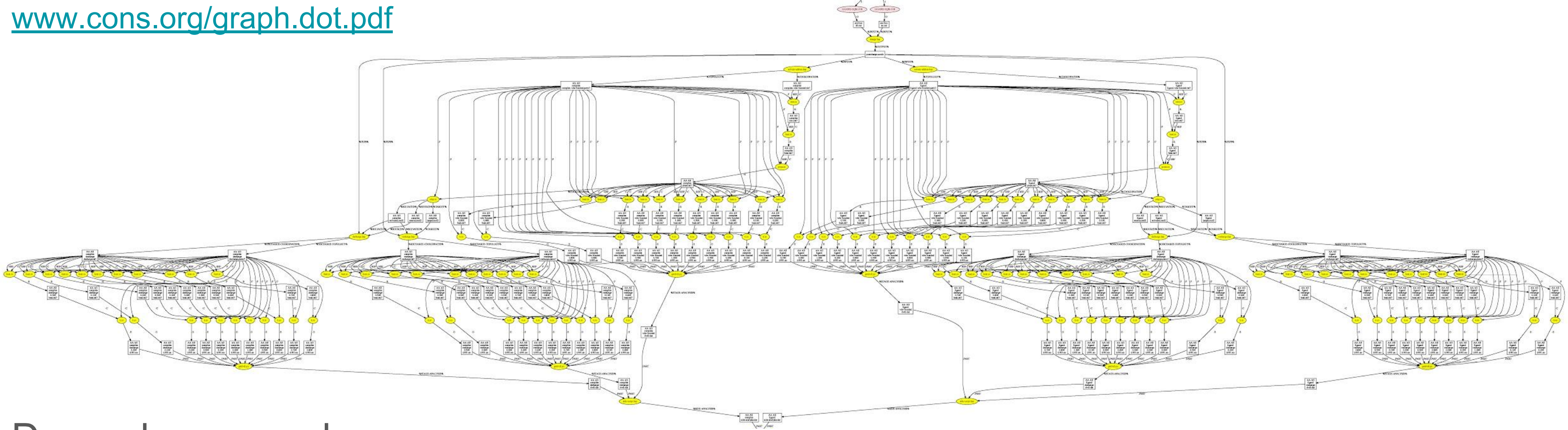
Distributed jobdir execution

- Hundreds of subjobs
- Subjobs can depend on previous jobs output
- Incremental jobdirs, update while it is executing
- No fixed machine assigning, new jobs picked up from graph as available

Heterogeneous fleet:

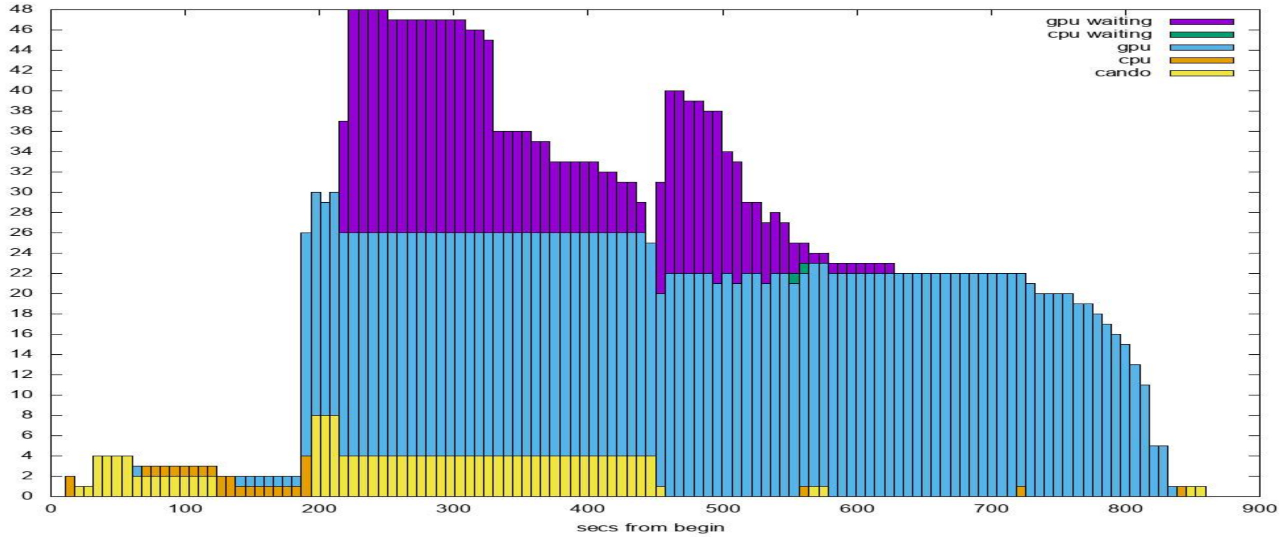
- Fast CPU (dumb jobs), many cores (multithreaded jobs), different GPUs
- Jobdir has ~150 GPU subjobs in 4-8 “classes”
- Different classes have different relative speed on different GPU types
- Matching between subjob class and GPU type known-ish

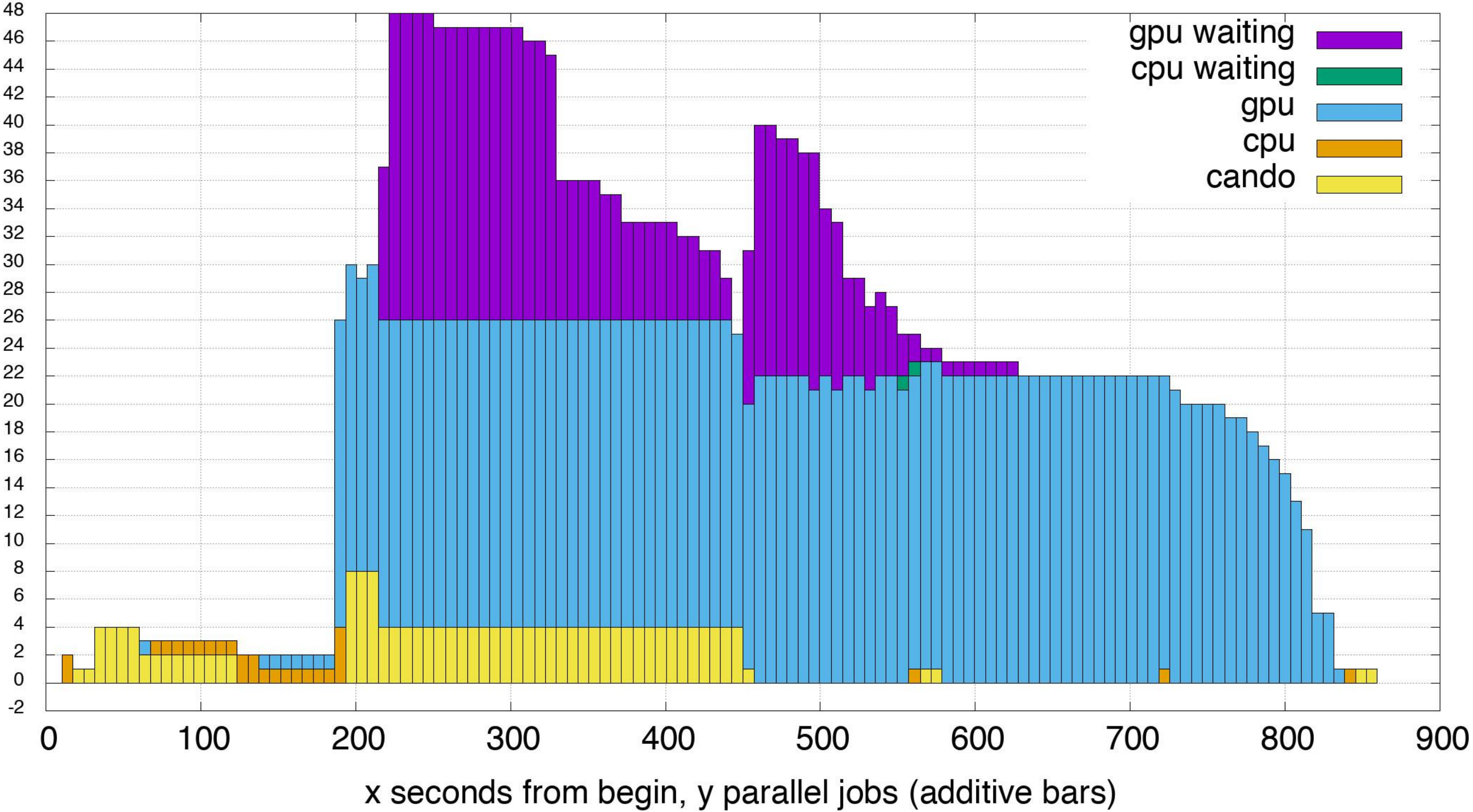
www.cons.org/graph.dot.pdf



Dependency graph above turns into job execution like this.

20 AWS ec2 GPU instances, Chris’ daughters gaming computer, GPU and CPU from Martin’s datacenter




gpu waiting
cpu waiting
gpu
cpu
cando
48
46
44
42
40
38
36
34
32
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
-2
0
100
200
300
400
500
600
700
800
900
x seconds from begin, y parallel jobs (additive bars)

Lisp advocacy

Drive home the message “we don’t just need readable software, we need *changeable* software”.

That requires compile-time computing. Keep every assumption you make during implementation in a single place. Generate all the other places where that assumption is needed from that one location, with compile-time code. You cannot hunt down human-driven spread of that assumption.

https://medium.com/@MartinCracauer

A gentle introduction to Compile-Time Computing — Part 1

bip - business in progress

E. Klockmann
H. Paulsen
T. Pohl
D. Junker

STK Consulting & Development GmbH

02. Apr 2019

bip

Enterprise Resource Planning (ERP) for

- Finance & Accounting
- Human resources
- Order Processing
- Supply chain management
- Project management
- Customer relationship management
- ...

bip

bip kernel + modules

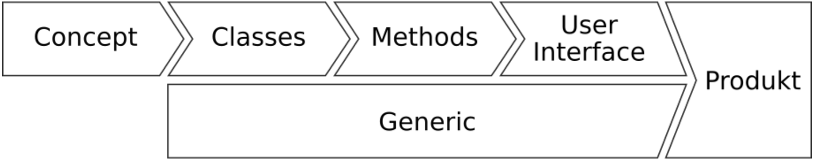
- bip industry
- bip logistics
- bip insurance
- bip ...

individual methods/add-ons/customizations

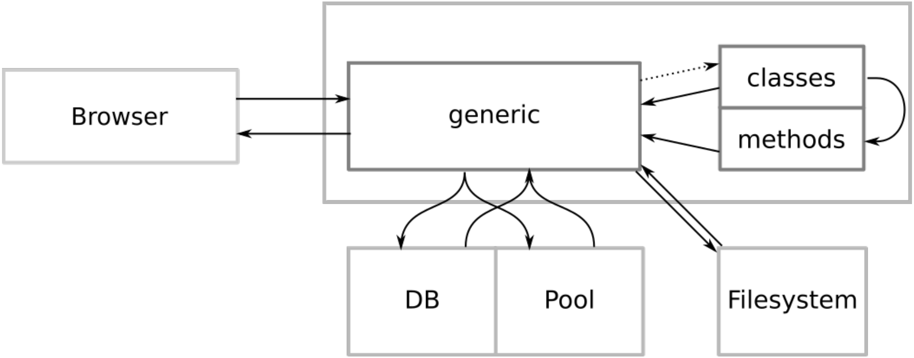
generic

- DB-access
- HTTP(S)-Server
- E-Mailing
- CSV import/export
- XBRL
- odt/pdf printfiles
- ...

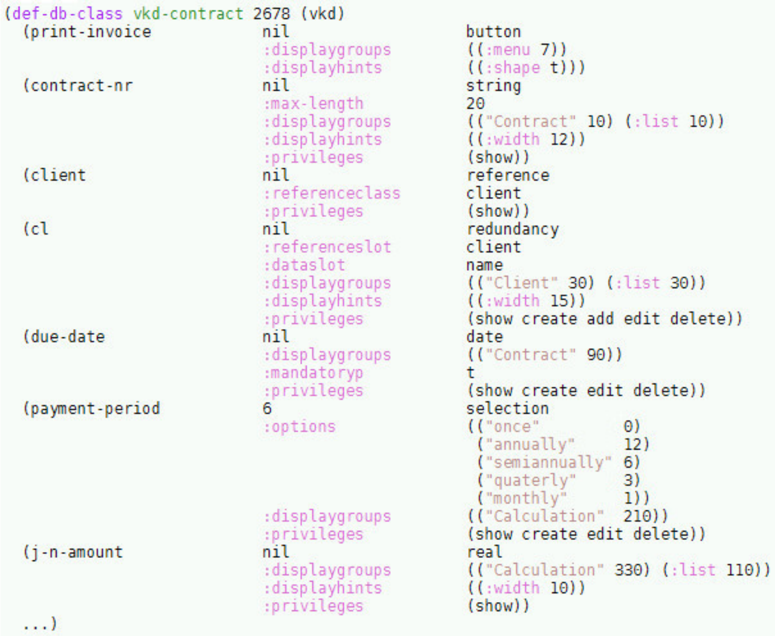
project development



overview

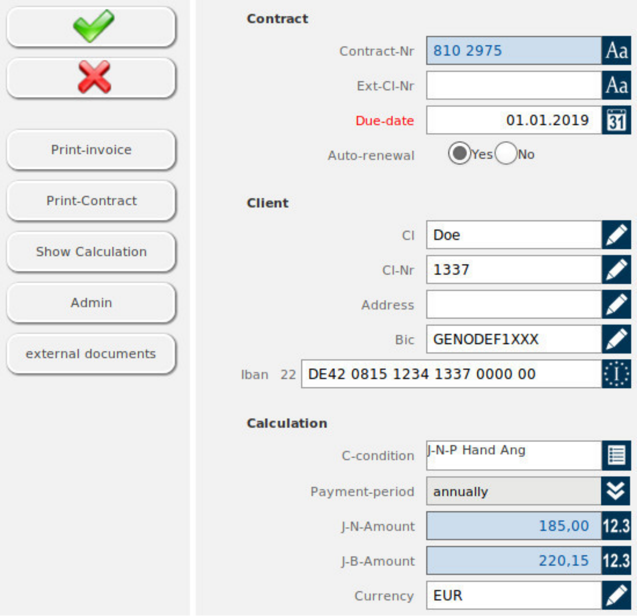


classes



```
(def-db-class vkd-contract 2678 (vkd)
  (print-invoice            nil                   button
                            :displaygroups        ((:menu 7))
                            :displayhints         ((:shape t)))
  (contract-nr              nil                   string
                            :max-length           20
                            :displaygroups        (("Contract" 10) (:list 10))
                            :displayhints         ((:width 12))
                            :privileges           (show))
  (client                   nil                   reference
                            :referenceclass       client
                            :privileges           (show))
  (cl                       nil                   redundancy
                            :referenceslot        client
                            :dataslot             name
                            :displaygroups        (("Client" 30) (:list 30))
                            :displayhints         ((:width 15))
                            :privileges           (show create add edit delete))
  (due-date                 nil                   date
                            :displaygroups        (("Contract" 90))
                            :mandatoryp           t
                            :privileges           (show create edit delete))
  (payment-period           6                     selection
                            :options              (("once"         0)
                                                   ("annually"     12)
                                                   ("semiannually" 6)
                                                   ("quaterly"     3)
                                                   ("monthly"      1))
                            :displaygroups        (("Calculation"  210))
                            :privileges           (show create edit delete))
  (j-n-amount               nil                   real
                            :displaygroups        (("Calculation" 330) (:list 110))
                            :displayhints         ((:width 10))
                            :privileges           (show))
  ...)
```

classes



Contract

Contract-Nr: 810 2975

Ext-Cl-Nr:

Due-date: 01.01.2019

Auto-renewal: ● Yes ○ No

Client

Cl: Doe

Cl-Nr: 1337

Address:

Bic: GENODEF1XXX

Iban 22: DE42 0815 1234 1337 0000 00

Calculation

C-condition: J-N-P Hand Ang

Payment-period: annually

J-N-Amount: 185,00

J-B-Amount: 220,15

Currency: EUR

why Common LISP?

- Macros
  - class description
  - method definition
  - print settings
- error handling
  - writing log files
  - message to browser
  - roll-back on error
- process clean up
  - restart task while debugging
- recompiling
- ...

bip - business in progress

E. Klockmann
H. Paulsen
T. Pohl
D. Junker

STK Consulting & Development GmbH

02. Apr 2019

Experimental Scheme

Christian Jaeger
ch@christianjaeger.ch

Exploring syntax

```
(require easy)

(def (f x)
  (square (inc x)))

(def g (comp square inc))

'(TEST
  > (for-all* (=> (produce-stream random-integer*)
                  (.take 200))
              f g)
 ())
```

Modules and predicates

```
(defmodule (<list-of> T?)
  (export null null? pair? cons first rest map fold)

  (def null (vector '<null-of> T?))
  (def (null? v)
    (eq? v null))

  (def pair-tag (vector '<list-of> T?))
  (def (pair? v)
    (and (vector? v)
         (= (vector-length v) 3)
         (eq? (vector-ref v 0) pair-tag)))

  (def list? (either null? pair?))

  (def (cons [T? v] [list? l])
    (vector pair-tag v l))
```

(Module cont.)

```
(def (first [pair? l]) (vector-ref l 1))
(def (rest [pair? l]) (vector-ref l 2))

(def (map fn [list? l])
  (if (null? l) l
      (cons (fn (first l))
            (map fn (rest l)))))

(def (fold fn start [list? l])
  (if (null? l) start
      (fold fn (fn start (first l)) (rest l)))))
```

(Module cont.)

```
(TEST
 > (modimport/prefix integer: (<list-of> integer?))
 > (%try (integer:cons "a" integer:null))
 (exception text: "v does not match T?: \"a\"\n")
 > (integer:fold
    + 0
    (integer:map square
                 (integer:cons 2
                               (integer:cons 3 integer:null))))
 13)
```

Object oriented

```
(defclass ((oolist _oolist) [predicate? T?])

  (defmethod (cons s [(.T? s) v])
    (oolist-pair T? v s))

  (def ((oolist-of T?) v)
    (and (oolist? v)
         (eq? (oolist.T? v) T?)))

  (defclass (oolist-null)

    (defmethod (map s fn)
      s))

  (defclass (oolist-pair [T? first]
                         [oolist? rest])

    (defmethod (map s fn)
      (oolist-pair T? (fn first) (.map rest fn)))))
```

(Object oriented cont.)

```
(def. (oolist-null.fold s fn start)
  start)

(def.* (oolist-pair.fold s [function? fn] [(.T? s) start])
  (.fold rest fn (fn start first)))

(TEST
 > (%try (=> (oolist-null integer?)
             (.cons "a")))
 (exception text: "v does not match (.T? s): \"a\"\n")

 > (=> (oolist-null integer?)
       (.cons 3)
       (.cons 2)
       (.map square)
       (.fold + 0))
 13)
```

ch@christianjaeger.ch
github.com/pflanze/chj-schemelib
www.meetup.com/London-Metaprogrammers/