RWTH AACHEN UNIVERSITY

Clojure

Seminar on Languages for Scientific Computing

Aachen, 6 Feb 2014

Navid Abbaszadeh

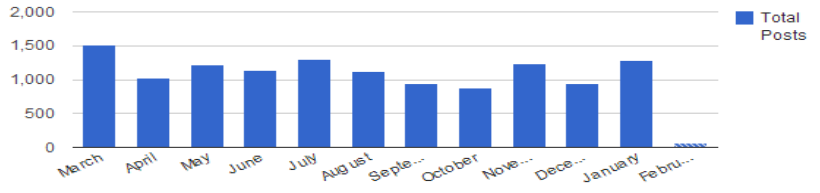
navid.abbaszadeh@rwth-aachen.de

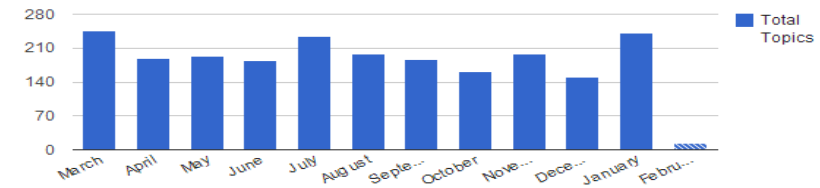
Overview

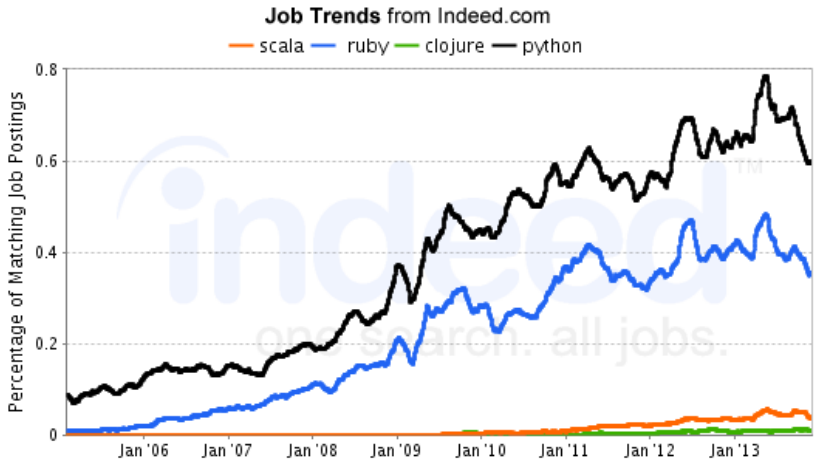
- Trends
- Introduction
  - Paradigms, Data Structures, Syntax
- Compilation & Execution
- Concurrency Model
  - Reference Types
- Performance
- Clojure & Scientific Computing
- Pros & Cons
- Conclusion
- References
- Demo

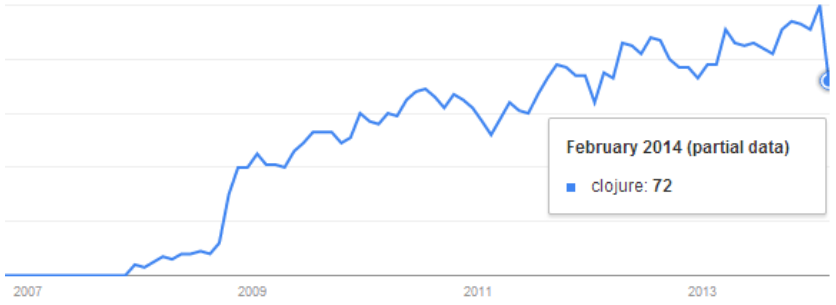
How Hot is the Topic?

- Clojure Google Group
  - ~8800 members
  - Active Community



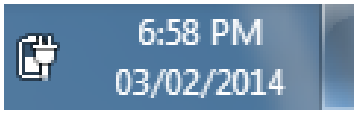






- Matching Job Positions
  - Still Educational

- How often it is Googled



Introduction

- Appeared in 2007 under Eclipse Public License
  - Rich Hickey created the language because "… I needed a Lisp, for functional programming, symbiotic with an established platform and designed for concurrency, and I couldn't find one."
- General Purpose: can be used wherever Java is used
- Hosted on JVM
  - Core in Java & Clojure
  - Current Version: 1.5.1, bin + source + docs < 5MB
  - No Installation
    - java -cp clojure.jar clojure.main /path/to/myscript.clj arg1 arg2 …
- Other implementations:
  - ClojureCLR (.Net)
  - ClojureScript (JS)
  - Clojure-py (Python)
  - Rouge (Ruby)

Paradigms

- Lisp
  - S-Expressions
    - a notation for nested list composed of tree-structured data
    - $(x_1\ x_2\ x_3)$ stands for $(x_1.\ (x_2.\ (x_3.\ NIL)))$ where x_is are S-Expression
  - Homoiconic
    - No distinction between code & data
    - Program Represented Via Core Data Structures of Language
  - Small Core, Extensive use of macros
- Functional (Clojure is Impure)
  - Functions: 1st-Class Citizens (HOF)
  - Why Pure Functional?
    - Easier to Understand, Test, and Implement Concurrency
  - Immutable Data Structures : Avoid mutating State
    - Persistent : Old Version + Changes → Efficiency

Paradigms

- Object Oriented?
  - Inheritance
    - Java Inheritance, Clojure Protocols
    - Clojure Hierarchies
      - Only to Support “isa?”!
  - Polymorphism
    - Multimethods
  - Encapsulation
    - Datatypes & Protocols : Define Methods for Interfaces
    - “Clojure eschews the traditional object-oriented approach of creating a new data type for each new situation, instead preferring to build a large library of functions on a small set of types.”

Dynamic

- Dynamically typed
- REPL
- Modify functions while running (Demo)

Data Structures

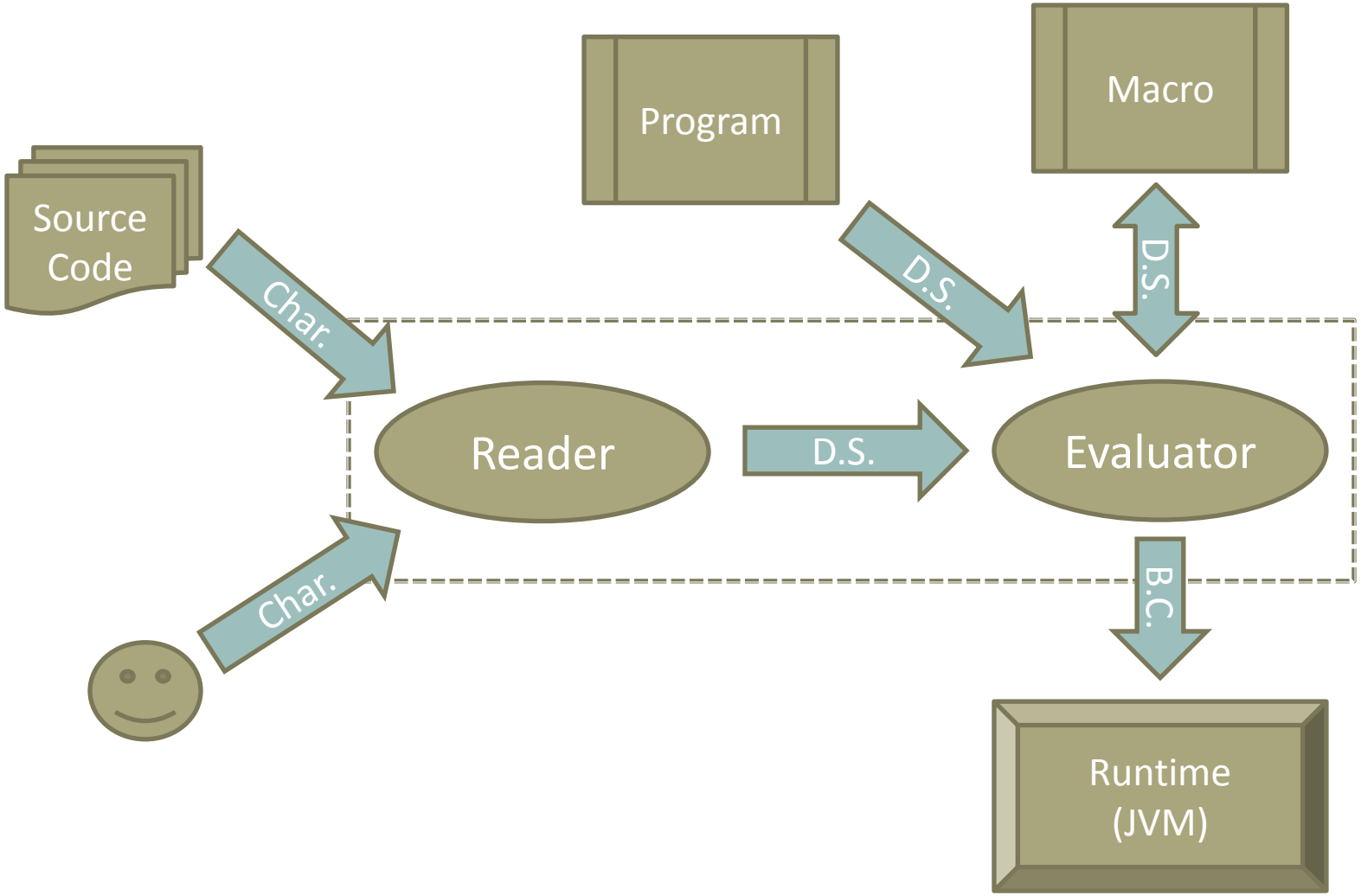
- Atomic
  - Arbitrary Precision Integers, Doubles, Strings, Ratios, etc.
- Sequences
  - (list <expr>*)
  - (hash-map 1 “one” 2 “two”)
  - sorted-map, hash-set, sorted-set, vector, etc.
    - Sugar Syntax e.g. (vector 5 8) = [5 8]
- ISeq Interface
  - (first (list 3 4 5)) → 3 (rest (list 3 4 5)) → (4 5)
    - Similar to Prolog

Get a Taste of Syntax

		
int i = 10;	Declaration & Assignment	(def i 10)
if(x > 0) return a; else return b;	Control Structure	(if (> x 0) a b)
a * b * c	Operator	(* a b c)
f(x , h(x))	Function Call	(f x (h x))
obj.f(x)	Java Method Call	(. obj f x)

int foo(int x, int y) {
System.out.println(“adding”);
return a+b;
}
	Function Definition	(defn foo [x y]
		(println “adding”)
		(+ x y)
		)

Compilation & Execution



Concurrency Model

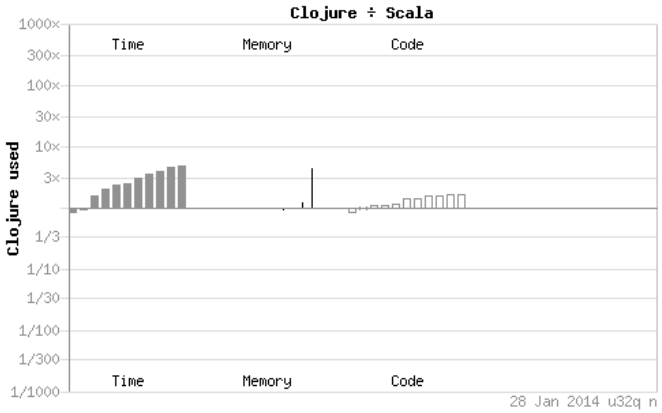
- Shared Memory Concurrency Options
  - Pessimistic Approach
    - Locks (popular in Java, C, etc.)
      - Low level issues
  - Optimistic Approach
    - Transactional Memory (Popular in Clojure, Haskell)
      - Same Idea as In Databases: ACID
      - Generally Easier to Implement than Locks
      - Identify Sections of Code requiring a Consistent View of Data
      - No Deadlocks, No Race Conditions
- Clojure Software Transactional Memory
  - Multi-version Concurrency Control
    - Maintain Multiple Versions of Objects with Commit Timestamps
  - Snapshot Isolation
    - Transactions operate on a Snapshot of Memory taken when they started

Reference Types

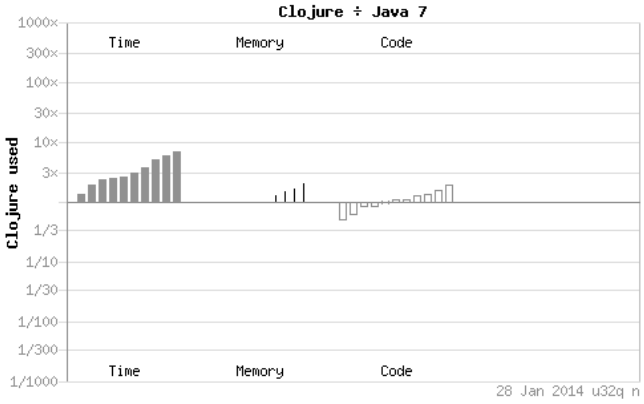
- Accessing Object Indirectly Through Boxes/Wrappers
- Boxes hold Mutable References to Immutable Objects
  - Var
    - Can have Shared/Thread-Specific Value
  - Atom
    - Independent, Synchronous change of Individual Locations
  - Agent
    - Independent, Asynchronous change of Individual Locations
  - Ref
    - Coordinated, Synchronous change of Multiple Locations
    - Only Modifiable inside Transactions
    - (dosync …)

Performance

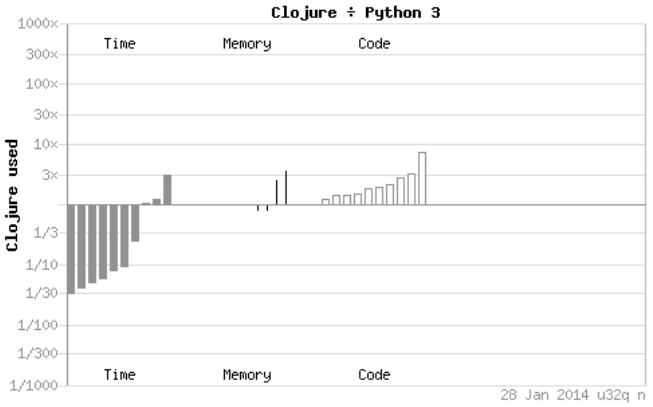
- Computer Language Benchmarks Game
  - X86 quad-core, Ubuntu
  - Based on 11 classic algorithms for benchmarking



Clojure vs. Scala



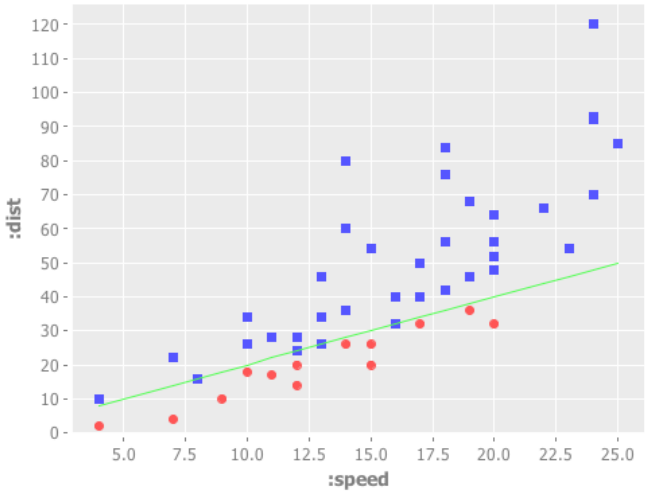
Clojure vs. Java 7

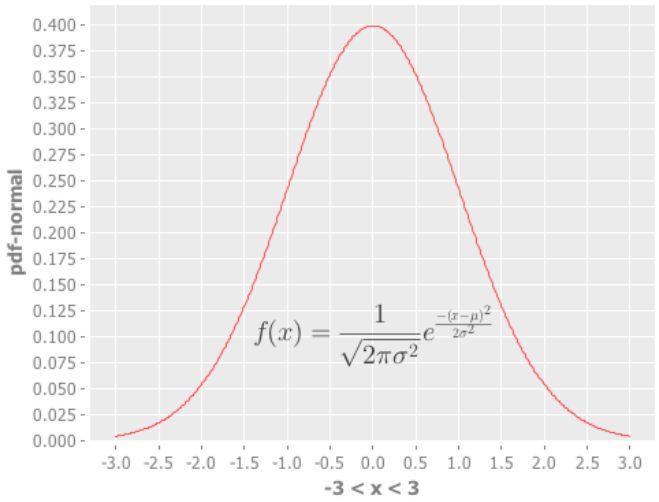


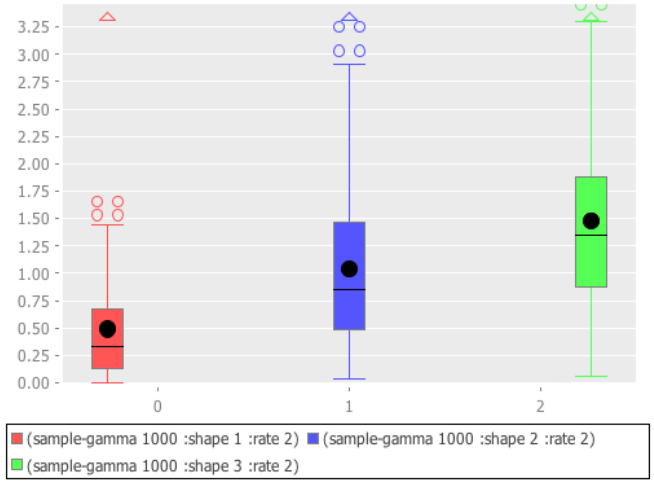
Clojure vs. Python 3

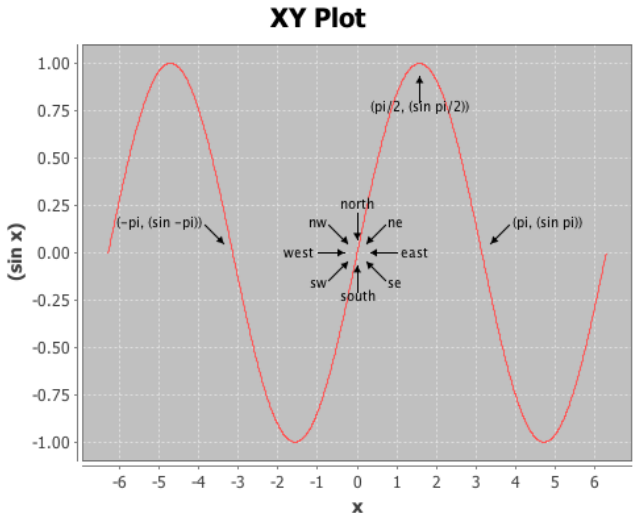
Clojure & Scientific Computing

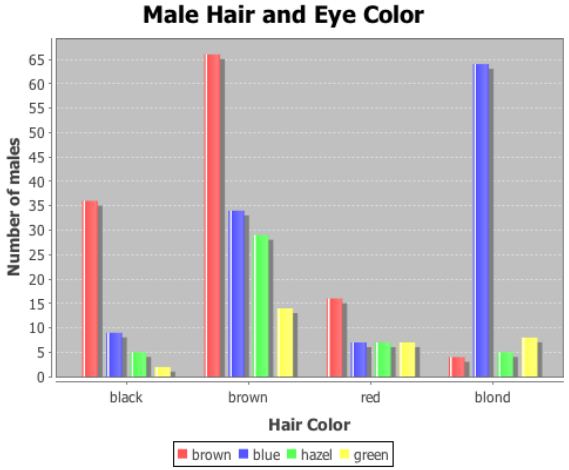
- Incanter
  - R-like platform
  - Mathematical & Statistical functions
  - Matrix & Linear Algebra functions
  - Charting and Visualization
- Direct access to java libraries.

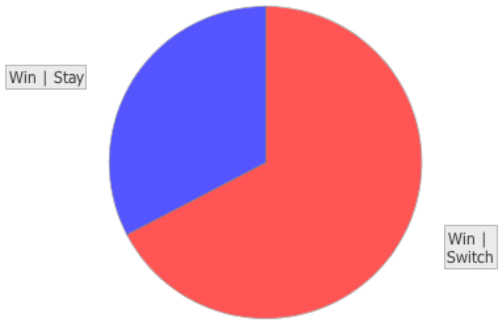





3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
1
2
(sample-gamma 1000 :shape 1 :rate 2)
(sample-gamma 1000 :shape 2 :rate 2)
(sample-gamma 1000 :shape 3 :rate 2)


XY Plot
(sin x)
x
(pi/2, (sin pi/2))
(-pi, (sin -pi))
(pi, (sin pi))
north
nw
ne
west
east
sw
se
south


Male Hair and Eye Color
Number of males
Hair Color
black
brown
red
blond
brown
blue
hazel
green


Win | Stay
Win | Switch

Pros

- Concurrency Using STM
  - Deadlock/race-free code, not even thinking about locks
- Ease of Code Generation
  - Manipulate Sequences to form AST
  - Print AST recursively
- Ease of Automatic Code Analysis and Optimization
- Benefit the Maturity of JVM
- Inherit the Abundance of Tools and Libraries from Java

Cons

- Performance
- Still Not Mature
  - Lacking Standards, Specifications, Documentation, Tools, etc.
- Misuse → as Readable as Assembly
  - Conventions can help
- Steep learning curve
  - Especially coming from an Imperative Programming background
  - Documentation presumes Lisp Background
- Lack of Trained Developers
- Flat Variable Structure Can Get Out of Control

Developer Tools

- Leiningen [1], Maven, Gradle (Build)
- YourKit [2] (Profiling for Java & .Net)
- Jswat [3] (Debugging for Java)
- The Clojure Toolbox [4] (a categorized directory of libs & tools)
- Clooj [5] (stand-alone jar file, IDE+compiler, written in Clojure)
- Plugins for Eclipse, Netbeans, IntelliJ IDEA, Emacs, Vim, etc

[1] http://leiningen.org/
[2] http://www.yourkit.com/
[3] http://sourceforge.net/projects/jswat/
[4] http://www.clojure-toolbox.com/
[5] http://www.clojure-toolbox.com/

Conclusion

- Clojure in my own words:
  - An amazing language helping you avoid repeating yourself. Coming from an imperative programming background, I love its flexibility, although it takes time to get used to programming model.
- Will I use Clojure?
  - Definitely, also interested in contributing in its development
- What I like most about it?
  - Concurrency Model
  - Flexibility as a result of being Homoiconic & using Macros
- Clojure is on the rise! [again, IMHO]

Try Clojure Online

- 4Clojure [1] (Problem-set & tutorial, easy to hard)
  - More than 500,000 Problems, sorted by difficulty
  - Explanations how to solve problems
- Try Clojure [2] (Online REPL)

[1] http://www.4clojure.com/
[2] http://tryclj.com/

References

- Clojure Official Website [1]
- Clojure Docs [2]
- ClojureTV YouTube channel [3]
- Incanter Platform [4]
- Computer Language Benchmarks Game [5]
- Clojure - Functional Programming for the JVM [6]
- Software Transactional Memory, Mark Volkmann[7]

[1] http://clojure.org/
[2] http://clojuredocs.org/
[3] http://www.youtube.com/user/ClojureTV
[4] http://incanter.org/
[5] http://benchmarksgame.alioth.debian.org/
[6] http://java.ociweb.com/mark/clojure/article.html
[7] http://java.ociweb.com/mark/stm/article.html

RWTH AACHEN UNIVERSITY

Demo

- Any Questions up to now?