# D1C : The autonomous systems compiler

- Spend 100% of your developper time on creative tasks
- Design complex multiprocess, multilanguage, multiplatform, multimachine systems by just coding functions and drawing graphs
- Don't bother with implementing complex functional/temporal interactions, communication between modules, networking and GUI data models and controllers
- Let D1C generate all this tedious stuff for you !

# Code modules and draw architectures

MODULES

A.py

```
def printme( x,y,z ):
   print x,y,z
   return
```

B.c

```
void mult(float* x, float* y, int N, float* out) {
      for(int i=0; i<N; i++) {
      out[i] = x[i]*y[i];
}
```

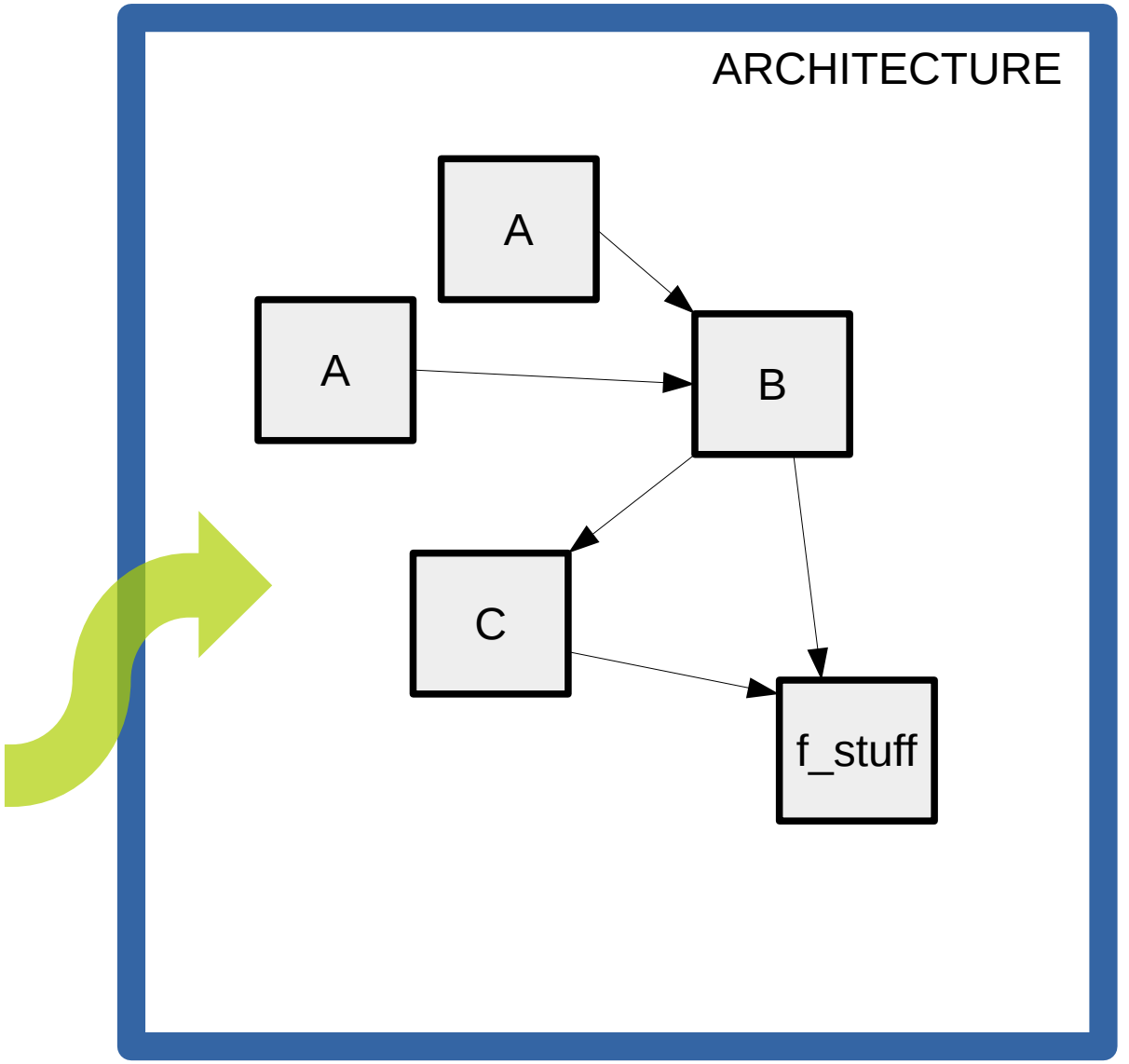
C.m

```
A = rand(N);
for r = 1:100
    for ii = 1:N
        out(ii) = A(ii,:);
    end
end
```

f_stuff.c

```
typedef struct {
    float x;
} MyData;

void function stuff(int gpe) {
...
}
```



# Modules vs Archi

**MODULE**

- Implements an algorithmic function
- Written in Programmer's preferred language
- Reusable
- Expose prototype
- Context-agnostic
- Compiled/Source
- No additionnal code
- Easily packageable
- Version-controlled

**ARCHI**

- Data-flow between modules => graph structure
- Graphically editable
- Live editable
- Multiple machines/processes/sources/observers/languages
- Provides Scheduling/timing
- Predefined relations (typed links)
- Automatic deterministic unambiguous deployment
- Easily exchangeable
- Easily batchable
- Easily reproducible
- Publication value
- Version-controlled

# Observations

- All programing languages already have a well-defined functional dependency and function prototyping syntax

  => Dataflow graphs are language-agnostic

- Calling an existing function from external code is native to all languages : this is the role of libraries !

- Parallelism, IPC, RPC, asynchronisms and timing is NOT native to all languages : requires system primitives and/or a virtual machine

  => IMPLEMENTATION of dataflow graphs is language-specific, and requires additionnal code and OS-specific primitives

# Problematics

- Given multiple functions over multiple languages/machines/OS/timings/processes
- Given a dataflow graph specification
- Can we automatically generate its implementation in a deterministic and systematic way ?
- Is there any benefit to implement it manually instead ?
- Is there any place for creativity here ?

# The lacking layer : From functions relations to processes relations

- Process = scheduling of functions
- Process = functions + time (sequential!)

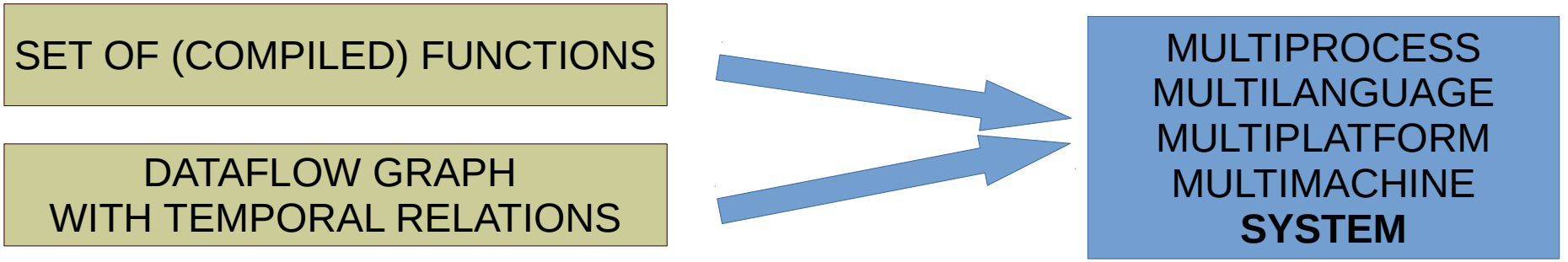
  => Program = process in computer's core
- Multiple processes can have relations

  => Processes are temporally linked

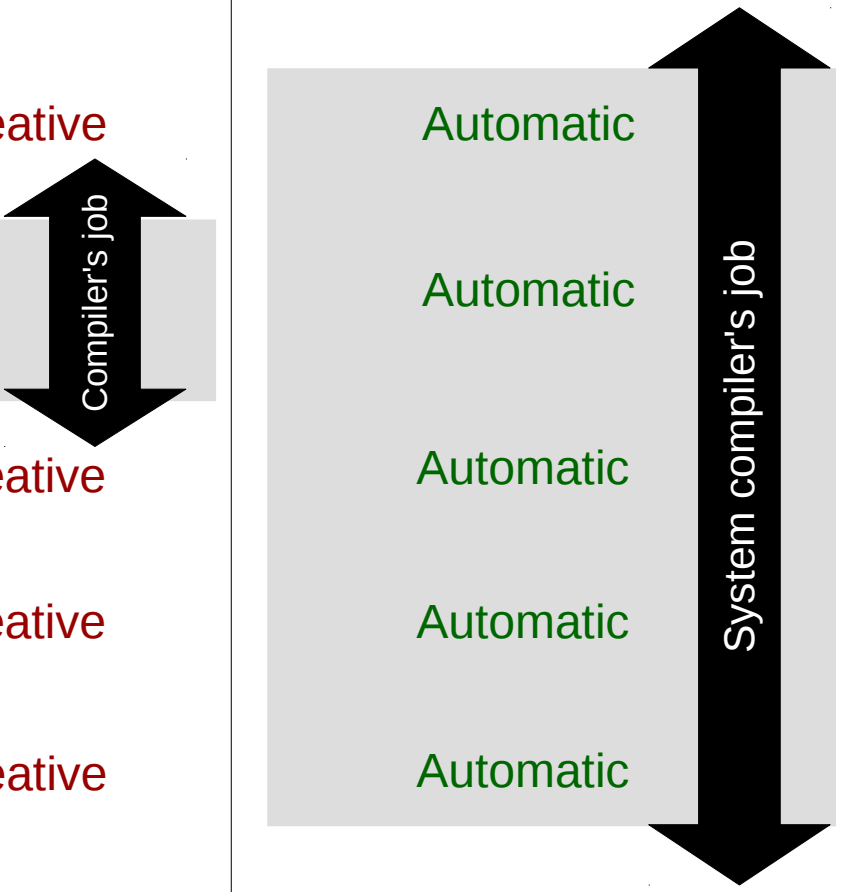
  => Designing a data flow requires specifying temporal relations
- We are lacking a standard way of describing coupled temporal/functional relations in a unified machine-readable fashion.

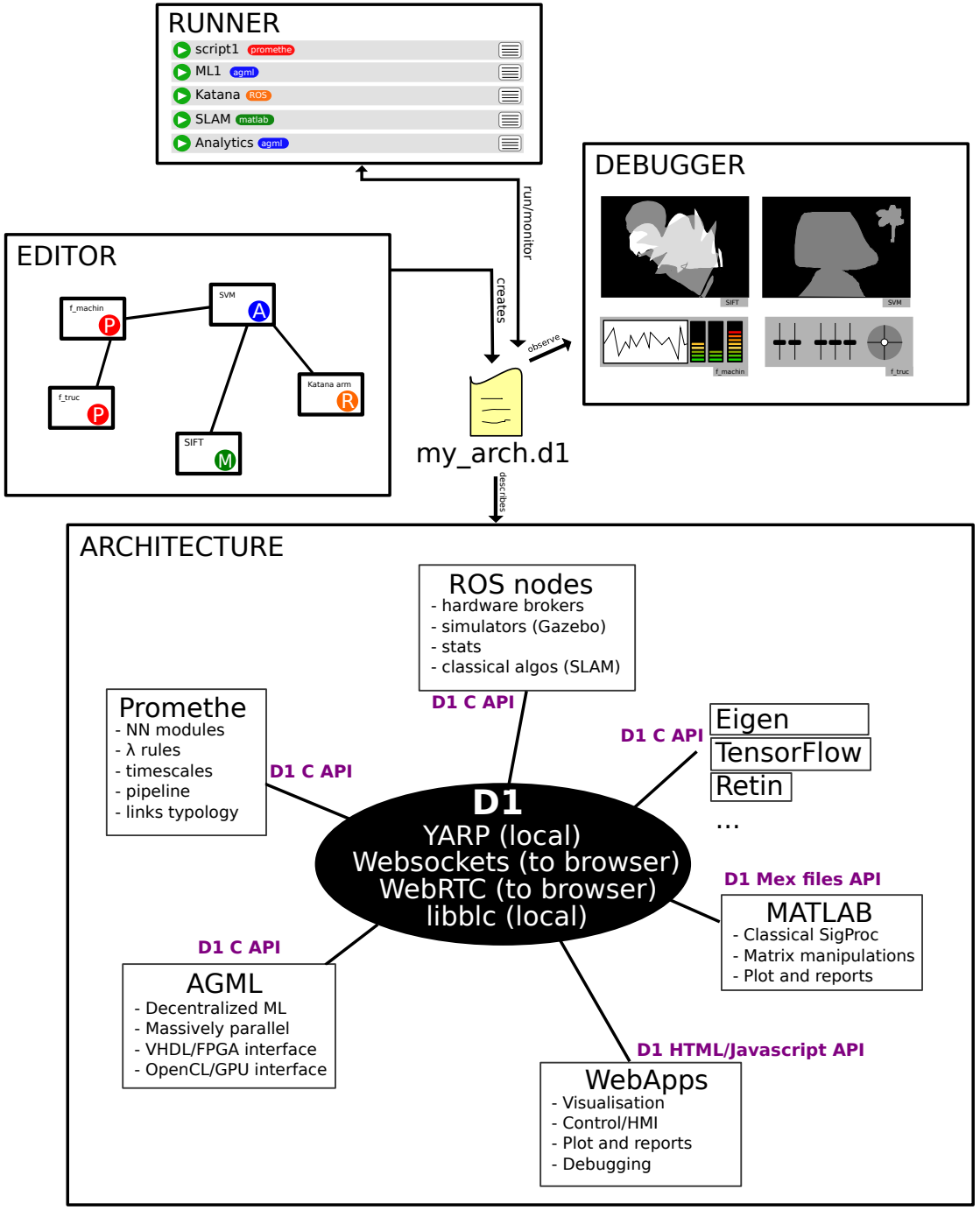
# The future of compilation : Compile Program → Compile Systems

- Compilation takes care of translating user's creative “artwork” into machine-readable instructions
- WHY THE HELL IS IT LIMITED TO SEQUENTIAL PROCEDURES AND STACK-CENTERED FLOWS ?!!
- Let's create a compiler of multi-process, multi-language, multi-platforms, multi-machines systems



| | Classical compilers | System compiler |
|---|---|---|
| Develop modules | Manual, creative | Manual, creative |
| Design architectures | Manual, creative | Manual, creative |
| Implement main procedures And communication routines | Manual, non-creative | Automatic |
| Compile all programs | Automatic | Automatic |
| Deploy on multiple machines | Manual, non-creative | Automatic |
| Launch programs | Manual, non-creative | Automatic |
| Develop communication layer with GUI | Manual, non-creative | Automatic |
| GUI design | Manual, creative | Manual, creative |
| Live system analysis | Manual, interesting | Manual, interesting |






Module (=function in the mathematical sense, y=f(x))

Functional link (can be remote/local sync/async request/stream)
=> Global URL scheme : x://machine/channel/obj
=> transfer low-high-level data (float* vectors, char* images, string messages)