

# Data Flow Processing, eventBased Algorithms and Data

## Main philosophy

Garo Garabedian  
Independent researcher,

*Everything around us can be solved by software architecture of data flow and we all react on events about changes on which we have declared our interest. I have tried to understand and find how to optimize our and computers' reactions in order to save time and work.*

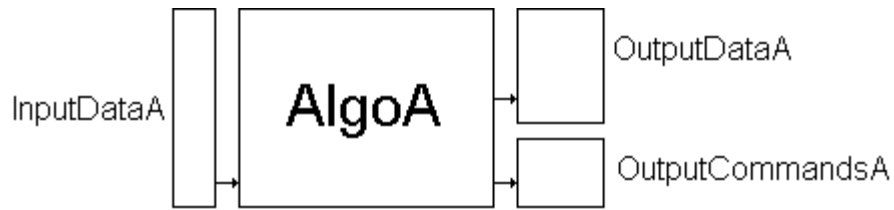
Humans' work can be generalized into three main periodically repeated, but on different objects in general, steps: collecting data, applying policy/ algorithm (taking decision), producing output data/ commands. When an algorithm or data is changed the whole process of work have to start again (reapplication) in order to apply (react on) the change.

If the program is cut to segments in such a way that output data of the one to be input of the other(s) the present research draws a tool building plan to optimize recalculations on a changed already computed data, which change of its own forces re computation.

Re computation is required only if it will produce different final result. The final result on a certain change of input data is only certain after full re computation in general, and even if a solution is build for a concrete product' current version, which solution to calculate if there would be a change of the result or not, so the present described platform is more rational and efficient in doing the same job.

In a comparison with the computer program, man takes many atomic input data, too, applies many times one and the same algorithm and produces output data or commands or both. When the algorithm or input data are changed the man reapplies the update not to the all system, but only where the changed part of algorithm/ data is computed (it is supposed to produce a different final result than when the algorithm/ data wasn't updated). This behavior of rationality of needed computations is the main idea of this paper.

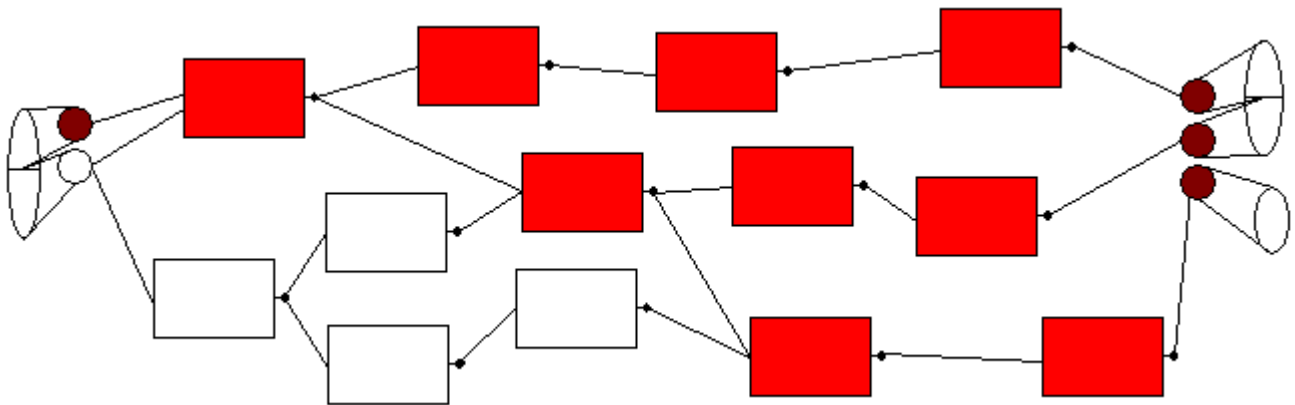
I offer to be declared computational atoms of every algorithm and every data with its structural atoms (smaller in volume). With aim to be check after every step (execution of computational atom) is there a difference between the result of the past computation and the changed (input data, algorithm or both) one. Algorithm atom works only with a group of data atoms or a single data atom.



*Diagram 1: The atom of the work, input data, algorithm, output data.*

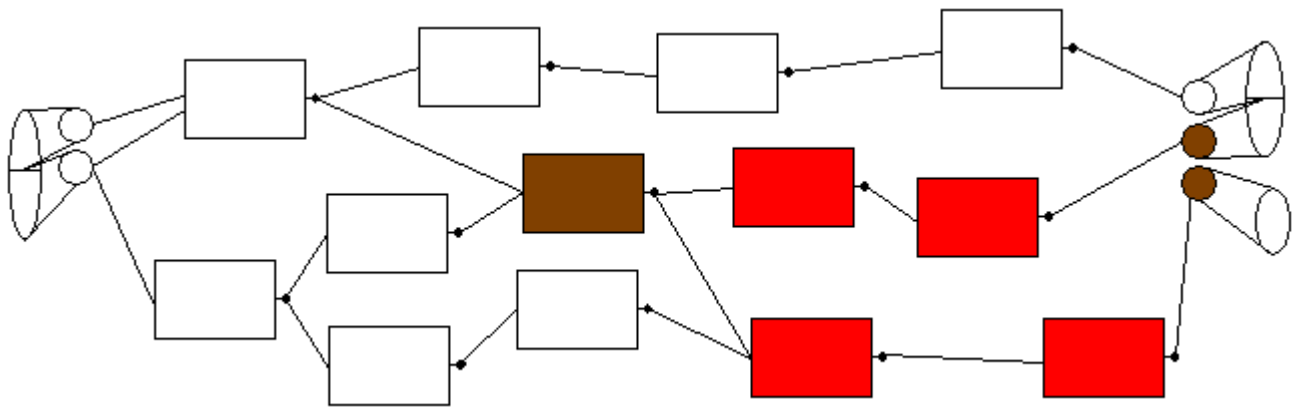
Note: Everything (said here) applicable to data is applicable to commands, too.

Changing of the input data (marked in brown) will produce need of additional rework (marked in red) and possible affection of the output (affected is marked in brown). The steps of additional rework are executed according to the main algorithm:



*Diagram 2: Process of rework when one input data is changed.*

Changing of the group of algorithms (marked in brown) will produce need of additional rework (marked in red) and possible affection of the output (marked in brown). The steps of additional rework are according to the main algorithm. The changed algorithm (in brown) is recomputed first:



*Diagram 3: Process of a work and how it have to be reworked when one algorithm is changed.*

In efficiency reasons, check after every step is there a difference between the past computation and the changed (input data, algorithm or both) one.

There exists input data which we can not check every time. It is not rational to call them every time, to detect possible change. In such case we have to call them in the worst circumstances, when data are directly related to a computation in process, but not an optimization computation.

### **Computer programs, computers applying algorithms**

In computer perspective the machine must be able to save the all input and output data. In this reason, the application must be written on enough abstract platform which enables sucking out (and

future injection) of the all amount of input and output data and commands.

In developers' perspective, find and declare as much atomic parts of the algorithms in order to technologically enable with a big percent of efficiency the searching and recomputing the only necessary input data when an update is applied.

### **Content editors, user works on data through computer applications**

Applying it to a human work, we can't recognize the all variations of the algorithm which people apply. But when somebody changes something we can inform him about the declared piece(s) of data which according to the declaration(s) is relevant in a described or not described path to the changed piece.

Data in the human work is connected by theirs relevance. Enabling technological storing of this relations can secure the user from making mistakes when changing some pieces of a big document. Without reading the whole document and trying to find the relations between the piece which he wants to edit and the rest parts of the document, the user can see the all relevant parts as they are declared in the system. One data atom can contain pieces of text which are parts of other data atoms.

There is no limitation a piece of data to be relevant to many pieces in different meanings

Because people apply different algorithms on content and this algorithms' count is not finite and because the relations between pieces is a function of the applied algorithm. The user will not be 100% secure of not changing content and in the view of the all document making it nonsense.

While the user is editing, every related piece of content to the edited piece have to be presented to him.

It is useful to enable the user to declare abstract algorithm atoms and model the process of work on one content, the algorithm atoms in their connections between themselves and with the data atoms.