The Design of Administrative Algorithms

Mikaël COZIC (LIS & IUF) & Pierre VALARCHER (LACL)

ALGOCIT Team (Université Paris-Est Créteil)

PEPS 2014-2017

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

1. Introduction

Introduction: Administrative Algorithms

- Computer software play an increasing role in administrative decisions.
- A significant amount of large scale public decisions, sometimes with lifelong consequences for citizens, are based on such software.
- Examples:

✓APB Algorithm (2009-2017): matching algorithm for the admission to French higher education institutions (+3000). +850 000 applicants in 2017.

✓Income Tax Calculation Algorithm. +36 million French households in 2016.

- The use of Administrative Algorithms (AAs), as we will call them from now on, has recently been the focus of vigorous public debates in France.

Introduction: the ALGOCIT team

- Lots of people complain about the lack of transparency of these computer-aided processes, asking notably for the publicity of the software source codes.
- These debates motivated the creation of ALGOCIT - a multidisciplinary group of researchers from the Université Paris-Est Créteil.

✓ A dozen people coming from Theoretical Computer Science, Software Engineering, Philosophy, Theoretical Economics, Management and Law.

✓One of our goals: to articulate a code of good practice for the design and implementation of AAs = a compact model describing how the whole process *should* take place.

Introduction

- Warning: most of the ideas put forward have been discussed in seminars and meetings of ALGOCIT. However, not every member of ALGOCIT will agree with them. The talk is thus not supposed to represent the collective judgement of ALGOCIT.
- This is work in progress !

2. Our Basic intuitions

Our basic intuitions: (1) source codes

- By looking at the public debates and by gathering pieces of information about how these software are designed, we were stricken by (what seems to us)

(1) an exaggerate emphasis on the requirement of publicity of the software source codes

✓The first source code of an AA communicated to a French citizen was the Income Tax Calculation Algorithm in 2016. It took place after an injunction by the Administrative Court (confirming the request of a regulatory agency, the CADA).

✓Under mounting public pressure, part of the source code of the APB algorithm was made public in october 2016.

✓The newly promulgated *Law for a Digital Republic* explicitly considers source codes as so-called "administrative documents", creating a right for French citizens to have access to them. Other features of the software are made accessible to citizens by the implementing decree of March 2017.

Our basic intuitions: (1) source codes

- We ultimately agree with the requirement of publicity of source codes. Through retro-engineering, on the basis of the source code,

✓One may infer some properties of the algorithm and of the computed function. In particular, one may check whether

- an *alleged* property of the computed function holds

- an *alleged* property of the algorithm holds

= generic adequacy between the software and higher-level of descriptions.

✓*Provided one has the relevant data*, one may also replicate a specific execution of the software and thus check whether a specific administrative decision has effectively been based on the execution of the official AA.

= specific adequacy between the software and a specific administrative decision (e.g., the University admissions in september 2017).

Our basic intuitions: (1) source codes

- However, knowledge of the source code doesn't guarantee understanding of the function computed by the algorithm.
- Why ? Because source codes operate at a very low level with respect to the one at which basic normative expectations are expressed and assessed.

Our basic intuitions: (2) functional properties

(2) a lack of attention to a level intermediate between

- the level of overall objective (e.g., "assign each student to an University") and general (mostly informal) normative principles (e.g., "everybody should be treated fairly"), and

- the level of software code

- This intermediate level = the functional level = the level of the properties of the function calculated by a given software.
- This level is the one traditionally tackled by Social Choice Theory broadly conceived (including notably matching theory), where these functional properties are typically called "axioms".

Our basic intuitions: (2) functional properties

- More specifically, we were surprised by the fact that
  - ✓The general public didn't ask for a description of the functional properties of AAs.
  - ✓It was unclear (to say the least) that the administration and the designers of the AAs build and choose the software in light of a description of their functional properties.
- This is all the more surprising that there exists a vast scientific literature (Social Choice Theory) which is devoted to
  - ✓ the formulation of precise functional properties (e.g., Neutrality, Unanimity Preservation, Stability, Non-Manipulability, etc.)
  - ✓the mathematical study of the consistency of possible packages of functional properties (with famous results showing that such-and-such package of properties is not consistent).

Our basic intuitions: (2) functional properties

- Basic contention: the design of, and communication about AAs should be centred around these functional properties.
- Functional properties are what really matters - for the administration *and* the public !

Our basic intuitions: (3) behavioral effects

(3) a lack of attention to the behavioral effects induced by the software, and more generally to the user/software interaction.

- Behavioral sciences (broadly conceived) show how sensitive people are to (apparently irrelevant or non-significant) details of their choice environments:

✓which options are selected as default options,

✓how information is displayed,

✓which words are used to frame the options and their consequences, etc.

- Several schools/academic areas or traditions put the emphasis on these features, e.g. the user/human-centered approach in design, the nudge literature in behavioral economics and cognitive psychology.

Our basic intuitions: (3) behavioral effects

- This calls for interventions by behavioral scientists (broadly conceived) both *before* and *after* the design of a prototype of the software

✓before: provide insights based on generic knowledge from behavioral sciences

✓after: devise "wind tunnels" behavioral tests of the software, in order to assess the response of future users and notably detect possible biases in their behaviors.

Our basic intuitions: (4) contestability

(4) an attractive normative principle, notably put forward by the political philosopher Philip Pettit (1997, 2000), is contestability: a democratic institutional arrangement should be such that citizens could effectively challenge public decisions.

- The control of the governed on the government is in general two-dimensional: electoral and contestatory.
- For reasons of practical feasibility, administrative decisions are typically under contestary control.
- Institutions should be organized so as to make contestations possible (requirement of contestability).

Our basic intuitions: (4) contestability

- In the present context, contestability implies several forms of publicity. Notably the publicity of source codes for the reasons mentioned above.
- Contestability implies more: it implies that the AA should be accompanied by a justification of the software choice by the administration.
- What kind of justification ? The core of the justification should mention

(i) the overall goal of the AA (e.g., to match students to universities)

(ii) the legal constraints (e.g., it is illegal to take into account such-and-such type of information)

(iii) the computational constraints (e.g., such-and-such functions cannot be computed efficiently)

(iv) the behavioral constraints (e.g., users are not able to rank consistently more than *x* options)

(v) the key functional properties of the software (inputs + output + key constraints on the inputs/output relation) = functional characterization of the software = set of functional properties satisfied by the function computed by the AA.

Our basic intuitions: (4) contestability

- The standard structure of the justification of the choice of an AA by an administration should be this one:

(1) provide a functional characterization of the software

Here is what we believe the software does....

(2) claim functional adequacy = to show that the alleged functional properties of the software are appropriate in view of the various constraints (legal, behavioral, computational).

Here is why we judge that it is good that the software does what we believe it does...(issue with logical, factual, value and interpretive dimensions)

(3) claim software adequacy = to show that the software satisfies the functional properties mentioned in the functional characterization (~ "specification" in the jargon of computer science)

Here is why we believe that the software does what we believe it does...(factual issue)

Our basic intuitions: (4) contestability

- Comment: depending on the case at hand, the "hot spots" of justification may hugely vary.

✓In some cases, few or no liberty in the specification of the function computed by the software given the constraints. Few things to say as far as functional adequacy is concerned.

➢The justification deals mostly with software adequacy.

- Example: the decrees and rules determine quite unambiguously what the Income Tax Calculation Algorithm should compute.

- On the opposite, the commitments involved in the choice of a function may be debatable, but the software implementation rather trivial. Few things to say as far as software adequacy is concerned.

➢The justification deals mostly with functional adequacy.

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

3. A compact code of good practice

The object, and the inspiration

- We tried to summarize these basic intuitions as a compact model describing an ideal process of design and implementation of an AA.
- We were notably inspired by the literature on software engineering (which rely on such notions as stakeholders, specification, owner products, etc.)
- Administrative Algorithm: an algorithm

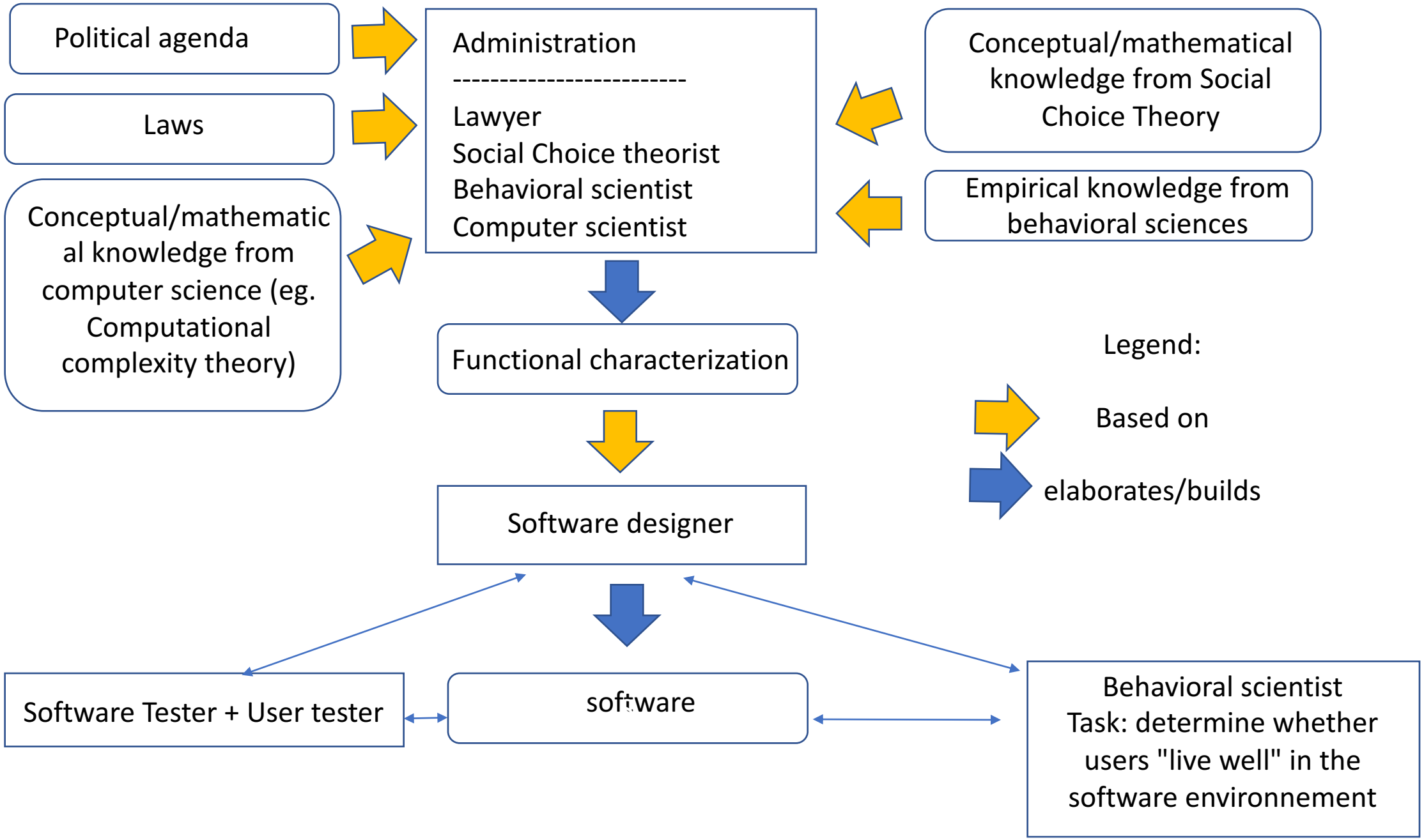
(i) whose *product owner* is a public administration

(ii) which takes as input pieces of information about a group ***I*** of impacted agents.

- the pieces of information may or may not be given by these agents

- the agents may be individuals or moral persons

(iii) whose output is a decision taken in the name of the public institution and which impacts agents in ***I***


Political agenda
Laws
Conceptual/mathematical knowledge from computer science (eg. Computational complexity theory)
Administration

Lawyer
Social Choice theorist
Behavioral scientist
Computer scientist
Conceptual/mathematical knowledge from Social Choice Theory
Empirical knowledge from behavioral sciences
Functional characterization
Software designer
Software Tester + User tester
software
Behavioral scientist
Task: determine whether users "live well" in the software environnement
Legend:
Based on
elaborates/builds

Some comments

- Ideally, at the end of the process, there should be both functional adequacy and software adequacy.
- These two forms of adequacy should be the basis of the public justification of the chosen AA.
- This public justification should be mandatory. Perhaps transmitted to a dedicated agency together with the source code.
- Contestability is not the only motivation for such a public communication. May also be important in order to "reduce duplicative software development and save taxpayer dollars" (Code.gouv)
- The granularity/degree of precision of the justification should be proportional to the stakes at hand (an AA whose function is to give slots to visit the Elysée's garden vs. an AA which deals with kidney transplants).

UPEC
UNIVERSITÉ PARIS-EST CRÉTEIL

4. Conclusions

Open questions ?

(Q1) Should the citizens have the means to check specific adequacy = that the software that the administration claims to use has been used to determine a specific administrative decision ? Troubles raised by the privacy of the individual data used by the algorithm. Delegate the verification to some independent agency ?

(Q2) When the administrative decision deals with people employed by the State (e.g., academics) as such, by contrast with citizens, are the requirements the same ?

(Q3) Should we go beyond contestability and require some involvement of citizens in the design of the AA ?

ALGOCIT TEAM - Université Paris Est Créteil

Matthias BÉGEAN, Management, IRG
Patrick CÉGIELSKI, Computer Science, LACL
Julien CERVELLE, Computer Science, LACL
Maïté GUILLEMAIN, Law, MIL
Ronan LE ROUX, Education Science, LIS
Farida SEMMAK, Computer Science, LACL
Arnaud THAUVRON, Management, IRG
Sylvie THORON, Economics, LIPHA
Julien TESSON, Computer Science, LACL
Marion VALARCHER, Sociology, OSC-Science Po
Noé WAGENER, Law, MIL

PEPS
2014-2017