Lessons learned and the application of mathematics during the development and implementation of an innovative software at the state health sector. Theoretical motivation and practical examples in Ruby (Rails).

English language version of the presentation given during the Municipal Science, Technology and Innovation week.

São Paulo, Brazil, 25 October 2017

Own translation and adaptation prepared: February 2018

Marcelo Eduardo Redoschi

Goals of this presentation

* To share experiences and lessons learned during design and development of an innovative web based information system at the state public sector

* In a broader sense, it manages the transfer of financial resources; the current implementation encompasses graduate scholarship program recipients.

System overview

* Development was based on technology previously used during my stay at Isitec (where I worked in 2013).
* Operational since April 2016.
* Modernized student registration and processing, provided improved bank payment (monthly scholarship) management and controls.
* Mathematics was used during software development and quality control.
    * Process optimisation (e.g. student's monthly attendance).

Basic assumptions

* Eliminate, as much as possible, paperwork and manual verification (inappropriate use of spreadsheets as 'databases').
    * (Greater) efficiency (principle stated in the Brazilian Federal Constitution).
    * Transparency and control (compartmentalized, with 'read-only' user access option).
* Intuitive web navigation interface, responsive, with links.
    * User friendly screens on both computers, smartphones and tablets, from the various manufacturers.
    * Designed, from the beginning, with internationalisation (i18n) in mind. Developed in English, localisation files written in Portuguese ("Brazilian").

Assumptions (continued)

* Sustainable (software) development. Harks back to the physical architecture/engineering.
    * Employs "cutting edge" free software, robust, without license costs to the public treasury.
    * Training and updates are more feasible (information is made available on the Internet by the community, developers are able to fully replicate the system on their laptops or place it on production without cost).
    * "Light", leaves small footprint. Able to run on a shared server.
    * Spread knowledge within the organisation ("local capacity building").

Technologies used (very succinctly)

- ✓ Modern, responsive screens: Zurb Foundation
- ✓ Framework: Ruby on Rails (free software)

devise, i18n, cancan, yard, railroady, rspec, ransack

- ✓ Database: Postgres
- ✓ Version control: git
- ✓ External double-checking: R(Studio)

Intended "audience" for the system (user base)

* Context is not-for-profit, for the public (government) sector.
* Over one hundred institutional users registered, from both private and public institutions. Some 40 colleges, universities, and research institutes which operate across São Paulo state (with nearly 1200 new scholarship recipients each year).
* The institutional communication is managed by a small unit (three people). Inspired by agile methodologies, I meet regularly with the team.
    * A "sprint" may normally take from a week to a month, depending on my current workload (since I am the sole developer). I have multiple duties within the organisation. Guiding principle: "local capacity building", to spread knowledge locally within the permanent civil servant staff (i.e. state employees which where approved on public selection exams, rather than outsourced IT technicians or political appointees).
    * It is my expectation that in the future, the students themselves will be able to access their own data directly, such as monthly attendance as well as to print scholarship payment confirmation slips from the system.

Lessons learned

* By 2017, users have incorporated smartphone and tablet viewing habits, therefore it is assumed that navigation, searches and reports will be intuitive, always.
* Important information, such as the ones which directly impact scholarship payment (or eventually social benefits, in another possible implementation) will require special treatment. At first, when the data is entered at the participating institution, it will be flagged as "pending" (unconfirmed) by the system. Only upon confirmation by the central unit (program management team) that the required documentation was duly provided would it then by marked as "confirmed".
    * A corollary of the above is that the system will allow users to attach documents (such as scans of sick leave medical notices), within reasonable size limits (e.g. 2 megabytes).
* There shall be a "definitions" menu, or however you wish to call it, with information such as:
    * Geography (countries, states/provinces, municipalities; perhaps with state administrative subdivisions or districts, depending on your requirements)
    * Bank branch list (which should be kept updated fairly often). For Brazil, I've used the official Central Bank spreadsheets, which are published monthly. The R language will help you with this.
    * Ministry of Labor Professions List ("Classificação brasileira das ocupações - CBO").
    * Etc...

Lessons learned 2

* Inspired by the Australian methodology, I've included gender information (optional, provided in case of diversity) and the options F, M, X.

* Project documentation is essential! If it is in English, even better.

* Within the source code (i.e developer commits)

* Tests (rake tasks, Rspec, watir)

* (Consistency) validations with 'self-explanatory' error messages.

* Help "balloons" on important fields (Zurb Foundation 'tooltips').

* Tutorials within the system (lightweight "html" pages). If content is tailored to the user's permission, better still.

Lessons learned 3

* The system or application will provide a specific solution (in my case the management of a scholarship program), but should be adaptable to handle other, more general situations (e.g. to manage a social assistance program).
    * The functionality to create users and enter contact data consistently is already provided. In addition, the system (in its production version) generates text files (for batch processing) in accordance with the format adopted by the local Brazilian bank which handles student payments.
        * Mathematics may (and should!) be used during system development and improvements.
* Sound logic is what the computer expects (irrespective of programming language or platform used).
* (Potentially) improves communication. It is advisable to use High School level mathematical concepts whenever possible, in order to make the content amenable to a lay internal audience. There is an enormous shortage (due to historical reasons) of (university trained rather than clerical workers) mathematicians, engineers, physicists and computer scientists active in the (Brazilian state public administration). Such structural reasons are (well) beyond the scope of this talk, but should be kept in mind.

Math theory

Definition (syntax of propositional logic)

An *atomic formula* has the form A_i where $i = 1, 2, 3, \ldots$. *Formulas* are defined by the following inductive process:

1. All atomic formulas are formulas.
2. For every formula F, $\neg F$ is a formula.
3. For all formulas F and G, also $(F \vee G)$ and $(F \wedge G)$ are formulas.

A formula of the form $\neg F$ is called *negation* of F. A formula of the form $(F \vee G)$ is called *disjunction* of F and G, and $(F \wedge G)$ is the *conjunction* of F and G. Any formula F which occurs in another formula G is called a *subformula* of G.

Example: $F = \neg((A_5 \wedge A_6) \vee \neg A_3)$ is a formula, and all subformulas of F are:

$$F,\ ((A_5 \wedge A_6) \vee A_3),\ (A_5 \wedge A_6),\ A_5,\ A_6,\ \neg A_3,\ A_3$$

We introduce the following abbreviations which allow a more succinct representation of formulas:

$A, B, C, \ldots$	instead of	$A_1, A_2, A_3, \ldots$
$(F_1 \rightarrow F_2)$	instead of	$(\neg F_1 \vee F_2)$
$(F_1 \leftrightarrow F_2)$	instead of	$((F_1 \wedge F_2) \vee (\neg F_1 \wedge \neg F_2))$

Math theory 2

$$(\bigvee_{i=1}^{n} F_i) \quad \text{instead of} \quad (\ldots((F_1 \vee F_2) \vee F_3) \vee \cdots \vee F_n)$$

$$(\bigwedge_{i=1}^{n} F_i) \quad \text{instead of} \quad (\ldots((F_1 \wedge F_2) \wedge F_3) \wedge \cdots \wedge F_n)$$

Here, $F_1, F_2, \ldots$ can be arbitrary formulas. In particular, that means that $(A \leftrightarrow E)$ is an abbreviation for the formula

$$((A \wedge E) \vee (\neg A \wedge \neg E))$$

which, again, is an abbreviation for

$$((A_1 \wedge A_5) \vee (\neg A_1 \wedge \neg A_5)).$$

Notice that formulas are nothing else but strings of symbols (i.e. syntactical objects). They do not have a "content" or "meaning" at the moment. Therefore, it would be incorrect (or premature) to read $\wedge$ as "and", and $\vee$ as "or". Better would be, say, "wedge" and "vee".

Formulas – and the components occuring in formulas – obtain an associated "meaning" by the following definition.

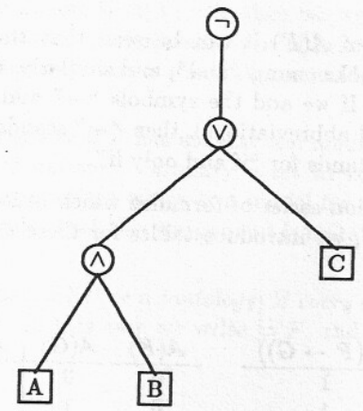
Definition (semantics of propositional logic)

The elements of the set $\{0,1\}$ are called *truth values*. An *assignment* is a function $\mathcal{A} : \mathbf{D} \to \{0,1\}$, where $\mathbf{D}$ is any subset of the atomic formulas. Given an assignment $\mathcal{A}$, we extend it to a function $\mathcal{A}' : \mathbf{E} \to \{0,1\}$, where $\mathbf{E} \supseteq \mathbf{D}$ is the set of formulas that can be built up using only the atomic formulas from $\mathbf{D}$.

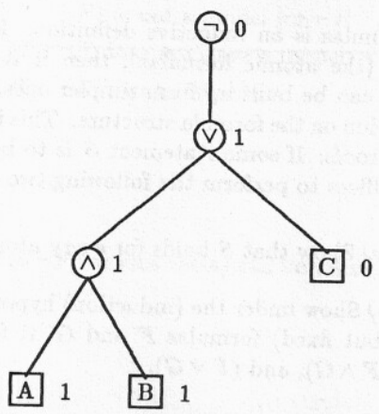
1. For every atomic formula $A_i \in \mathbf{D}$, $\mathcal{A}'(A_i) = \mathcal{A}(A_i)$.
2. $\mathcal{A}'((F \wedge G)) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 1 \text{ and } \mathcal{A}'(G) = 1 \\ 0, & \text{otherwise} \end{cases}$
3. $\mathcal{A}'((F \vee G)) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 1 \text{ or } \mathcal{A}'(G) = 1 \\ 0, & \text{otherwise} \end{cases}$
4. $\mathcal{A}'(\neg F) = \begin{cases} 1, & \text{if } \mathcal{A}'(F) = 0 \\ 0, & \text{otherwise} \end{cases}$

Math theory 3

Using these tables, it is easy to determine the truth value of a formula F, once an assignment of the variables occuring in F is given. As an example, we consider again the formula $F = \neg((A \wedge B) \vee C)$, and we represent the way F is built up by its subformulas as a tree:



The truth value of F is obtainable by marking all leaves of this tree with the truth values given by the assignment $\mathcal{A}$, and then determining the values of the inner nodes according to the above tables. The mark at the root gives the truth value of F under the given assignment $\mathcal{A}$.

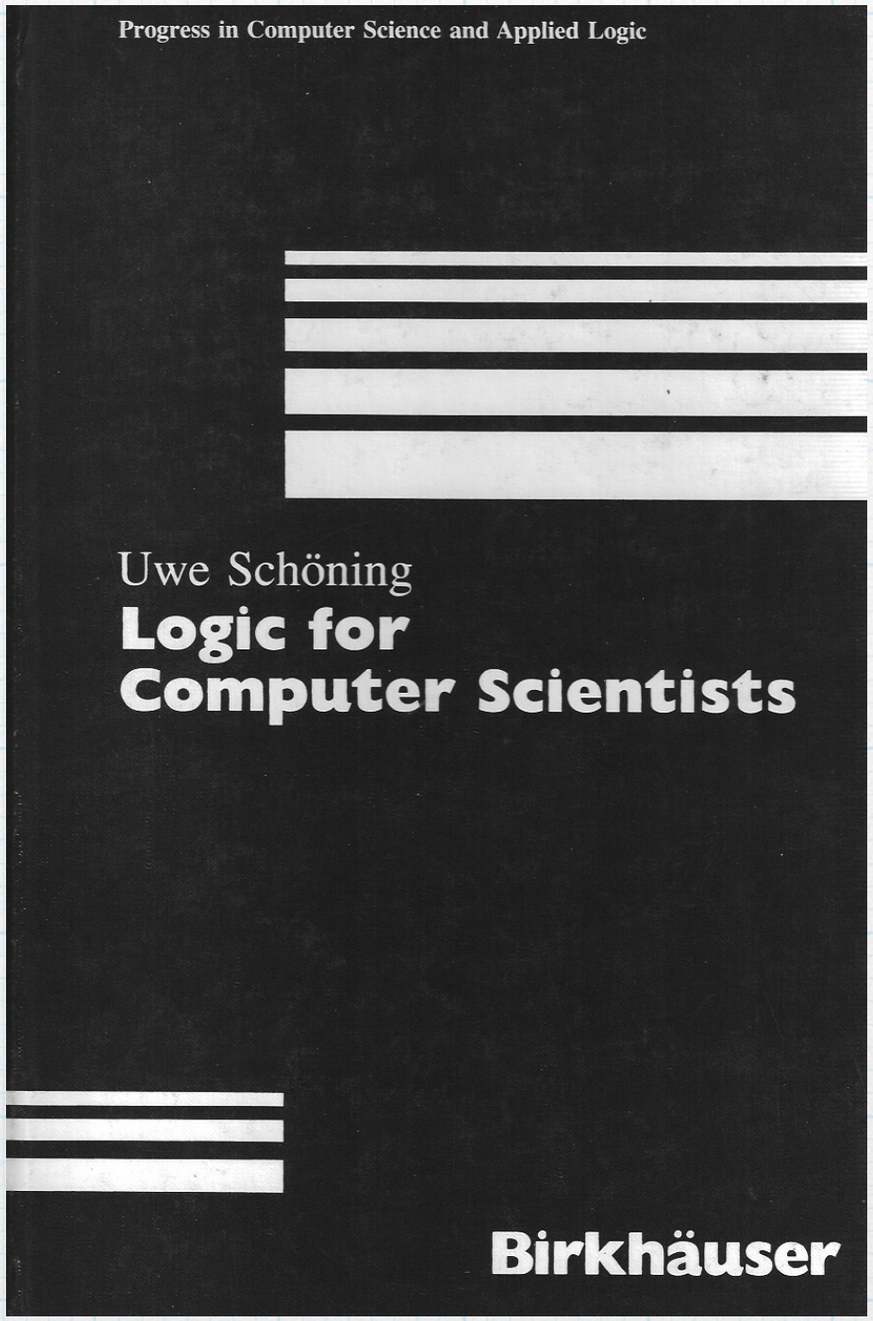


Objective: obtain valid formulas (logical consistency)

all formulas in propositional logic

valid formulas	satisfiable, but non-valid formulas		unsatisfiable formulas
$\neg G$	F	$\neg F$	G

Theoretical basis (reference textbook)



Marcelo Eduardo Redoschi - English version of the presentation given during the Municipal Science, Technology and Innovation week - São Paulo, 25/10/2017
This translation prepared: February 2018

Practical examples of math usage

Mathematical concepts such as binary trees and properties of sets were taken into account during the system's design. This approach allowed me to more accurately define relatively complex concepts - first using mathematical notation, and later expressed in the Ruby programming language - precisely and efficiently (seeking to write) ("elegant code").

Marcelo Eduardo Redoschi - English version of the presentation given during the Municipal Science, Technology and Innovation week - São Paulo, 25/10/2017
This translation prepared: February 2018

Improved student attendance tracking (practical application)

* As an example, I could mention an important rule (definition, concept) which I've started using, pertaining to the scholarship recipients' attendance tracking:

* "All students are assumed to present (every day), unless there is something recorded on their file for that month".

* That is to say, I've inverted the logic commonly used (which assumed nobody was present unless so recorded), whilst preserving the final result.

* This (seemingly) simple initiative immediately produced a 90% to 95% reduction in redundant clerical work; since the vast majority of students will be present every (school) day during the month.

Student attendance tracking (refinements)

* By pursuing the analysis further, it was possible to make a few refinements. When considering those students with some recorded event during the month, for instance, there would be two possible scenarios.

* Every (attendance) event which isn't specifically marked as an absence will be (some kind of) leave. This (concept) inside the in Ruby (on Rails) programming became like so:

```ruby
def self.leave

  where.not(:id => self.absence)

end
```

Details

* Reimbursed (paid)

```
def self.paid_leave

  leave.joins(:leavetype).merge(Leavetype.paid)

end
```

* Pending

```
# Not confirmed - management authorization pending (e.g. paid leave)
def self.pending
     where.not(id: self.confirmed)
end
```

Second example of mathematics application (external verification)

* (For the sake of completeness) student attendance, registration and financial payments are double checked externally (i.e. outside the system, whenever required). This is performed using the R language (RStudio program, community version).

* Innovation example: the so called "special" registrations (make-up, repeat) are already recorded by the system. This feature is being perfected. Here I've used set theory. We know with 100% certainty that all registrations are either normal or "special" (i.e. mutually exclusive subsets)

Special thanks

* Prefeitura de São Paulo, Secretaria de Desenvolvimento, Trabalho e Empreendedorismo.

* Sindicato dos Engenheiros no Estado de São Paulo.

About the speaker

* Mathematician, systems architect/developer, translator.
* Bachelor of Science (Mathematics), University of Maryland at College Park, 1993
* Master of Business Administration (Information Systems), The American University, 1997
* Master of Letters (Italian), Universidade de São Paulo, 2008

E-mail contact

mredoschi@uol.com.br

Marcelo Eduardo Redoschi - English version of the presentation given at the São Paulo Municipal Science, Technology and Innovation week - 25/10/2017
Translation prepared: February 2018