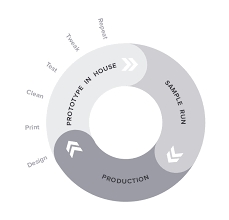
**CronBatch © in 45 minutes**

**Analysis, design and implementation**

CronBatch is a programmable web application written in php, javascript, html, sql to explore the world wide web areas: surface, deep and dark. The CronBatch main purpose is to provide up-to-date info on the diversity of the two web areas through sampling and scenarios built out of the analytics. It serves well the purpose of online sampling, web page identification, input for statistics, to assist researchers in carrying out netnographic studies and online surveys. The design has been based on a cycle of extreme rapid prototyping following requirements gathered as needs arisen from an actual netnographic research project. The project involved the identification of an internet sample of digital online storefronts and the selection of narratives from private forums or other means of online communication. The requirements were identified after a first phase of adaptation and research of the environment under investigation with objectives for identifying an internet population of web stores.

**Interface**

The interface design has been directly prototyped and implemented with languages html and Javascript containing visual objects which can be used for the implementation of interfaces prototype with ease and practicality giving an idea of functionalities and visual aspect of the final interface in a very short time. This has been possible considering the poor of input and dialogue interface under consideration. A simple paper prototype reporting just the visuals and the dialogues has been implemented using languages html with basic dialogue functionalities in Javascript.

The interface design has been compressed within the actual development/implementation of the interface saving the time into one stage of design/implementation.

**Code HTML, Javascript, PHP, C, C++, SQL**

The same route has followed the analysis, design and implementation of code and database. Conceptually the implementation of the code and the database has absorbed most of the design stage. Code routines, classes, database tables have thus been designed and implemented without a formal analysis and design but implemented after a first abstraction of the code classes and entities. The code main routines have thus firstly been implemented with functional objectives and successively later classes abstracted maintaining the core of the code already written as routine methods. Main entities normalization followed the code development and implementation.

Entities identification

Entities normalization

Relations, messaging and data records

Classes for data management

Classes for searching/scraping

Interfacing classes

Synchro and schedule for Cron offline background processes

**Database: MySQL**

Entities Normalization

Entities relationships Interfacing classes

Software tests revealed timing issues, loss of speed and the quantity of data to store or transmit over the internet. Subsequently a stage of test and maintenance followed for finely tuned routines and optimization and abstraction of entities and classes.