Event and Object Relational Mapping

March 29, 2016

Michael Braverman
Designer of Technology
New York, US
bravm321@newschool.edu

ABSTRACT
This project is an attempt to explore certain fields in technology that remain uninfluenced in non-technical applications. One of these technologies that would defend the argument is Object Relational Mapping (ORM) which is a field in the sphere of technology that lacks an initiative for innovation. It's effectivity has proven to not be significant due to the non-substantial ways of its application. The project will explore wether the reason for ORM' ineffective applications are the principles based on which its data relations are drawn. The relation between data will be created with the help of a physical computing device that will gather data from the physical environment and visualize it using a technology based medium.

Author Keywords
Object Relational Mapping (ORM), Artificial Intelligence (AI), machine learning, necessary connection, object, event, event significance, trigger event.

INTRODUCTION

Object Relational Mapping (ORM) is known as a form of database manipulation in the network technology field. It is designed to create basic relations in data without the use of the more advanced Machine Learning computation techniques. Similar to its adjacent fields that explore data manipulation and relation among different data, the application of ORM in most technological applications, remains unexplored and undeveloped. As a concept however, ORM has the potential to become a an innovative concept in the sphere in technology but in the applications that it has been applied thus far, it might be proving the contrary. http://blogs.tedneward.com/post/the-vietnam-of-computer-science/

It is therefore important to outline the benefits of this technology more clearly and try to confront its poorly documented drawbacks and come up with new effective application outside of its typical.

SOLVING ORMS COMPLEXITY

Solving the complexity problem may be easier than it might appear when taking into account certain philosophic principles developed long before the advent of digital technology. Although applying philosophic theories into design concepts may seem as an addition to problem's complexity, they sometimes allow to view the problem from a different angle and come up with more a penetrable solution.

David Hume's Necessary Connexion

An English philosopher David Hume (1711 -1776) outlined in his *Enquiries concerning Human Understanding (1748)* how compositions, events, and actions are all connected together by some bond, tie, or *principle* (see Figure 1). This theory is also know as the *idea of necessary connexion* and although David Hume's remarks mostly concerned the relationships between ideas within the human mind, this principle served as a great influence for the application of Hume's theory into the physical and metaphysical realm of philosophic theories. David Hume outlines three important characteristics of an event that can be subject of mapping the *necessary connexion* among ideas, and are the following:

1. Contiguity
2. Resemblance
3. Cause & Effect

By assigning data to events according these three principles and comparing events to each other should lead to effective mapping of events.

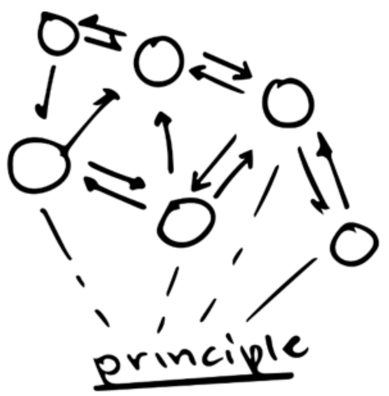

Figure 1. A representation of a principle and its common relation with events

Objects vs. Events

In this paper, it is necessary to define the relation between an *object* and an *event* since there is a merge of their definitions when *Object* Relational Mapping (ORM) is defined together with an emphasis on mapping *events* based on data sources derived from a physical computing medium. When defining the terms literally, as the *American College Dictionary* defines an *object,* "something that may be perceived by the senses, especially by sight or touch". An *event* on the other hand, is an occurrence of an action at a certain point in the dimension of time. Both of these notions however, have twofold meanings which intersect with each other once defined. Moreover, certain thinkers can draw no metaphysically significant distinction between *objects* and *events*, treating both as entities of the same kind. http://plato.stanford.edu/entries/events/#EveVsObj

It is within these assumptions that object-oriented programming has proven to effectively treat any entity the same as any other, wether it is an *event* or an *object.* Since this project mostly concerns the application of technology based on technologic principles, it it possible to proceed with defining these two terms under one common definition. An example that illustrates a comparison of events as if they are objects is illustrated in Figure 2.

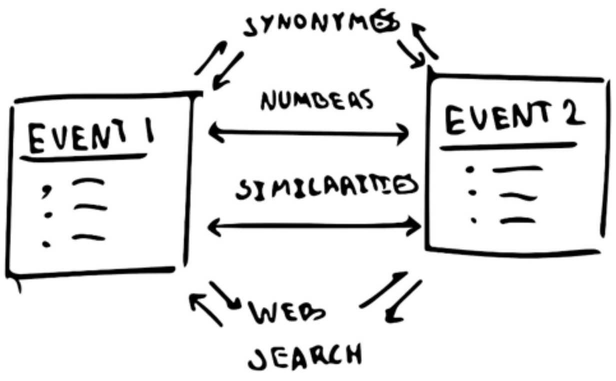

Figure 2. Comparing two events as if they where objects

OUTLINING THE PROBLEM

Before coming up with solutions to a problem, it is important to articulate correctly the problem that must be solved. The problem with ORM not only in the opinion of technological experts but also witnessed in its applications, is that it introduces an unnecessary level of complexity when applied in an incorrect manner. Having a great potential to outline relations between objects ORM lacks however the ability of being easily applied in its most effective manner. Currently, there are other data relation techniques that are in active research and are used to establish relations between sets of data such as Machine Learning and Artificial Intelligence (AI). Although these techniques can be proven to be useful in many applications, they don't necessarily solve the complexity problem due to their even more complex structure. The very definition of ORM suggest a simpler way of establishing connections in already available data. While it might not be capable of generating new data connections as sophisticated as Machine Learning or Artificial Intelligence (AI), it suggest to be useful for making connections with already available data. By applying Hume's three principles of event contingency to the way events are related to each other, it could be possible to apply ORM in less technical applications.

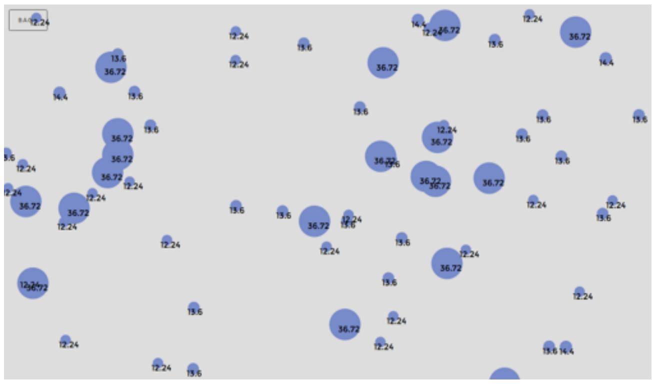

Figure 3. alpha version of in.fORM

IN.FORM: APPLYING THE CONCEPT

The project that showcase the basic essence of the concept in a real world application is called in.fORM (see Figure 3). It consist of various technologies and code libraries such Arduino, node.js, p5,js, d3.js, JSON, an accelerometer, a temperature, and a light sensor. The event processing happens on the Arduino and due to this solution, a faster variant of an Arduino compatible board was used, called the Teensy 3.1. It is capable of running up to 50 times faster when overclocked if compared to a simple Arduino. This addition in speed is necessary for reading, storing, manipulating, processing, and sending the gathered sensor data.

Triggering an Event

On the data processing side, the Arduino board has a so called *trigger event* that is executed when a certain threshold for registering an event is passed, such as a drastic change in the lighting, or a sudden change in the accelerometer readings. The way the threshold is calculated is first by getting all the values of from the sensor and storing them in an array of 200 readings long. Every couple updated array readings, the Arduino will take the data for

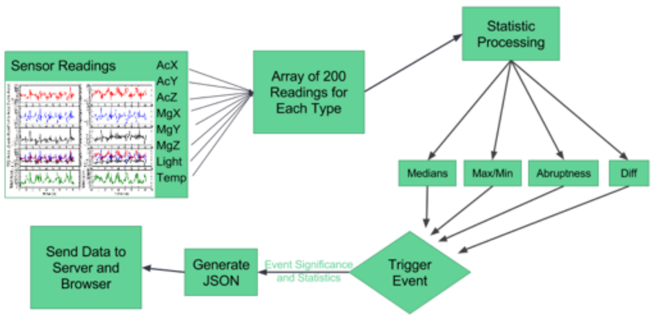

Figure 4. Demonstrates the way the Arduino processes the sensor data

each of the sensors and calculate basic mathematic statistics and relations between them such as maximums, minimums, and medians. Based on these simple attributes of mathematical analyses, it is possible to come up with with further calculations of how abrupt an event occurred and how many other attributes of the physical environment where effected at the same time. All these values effect the so called *event significance* which is calculated based on how dramatic the occurring event was. This is perhaps the most underlying variable that describes an event since it defines its importance and rate of change.

Parsing and Visualizing an Event

When an event gets triggered and its *event significance* is calculated, the Arduino parses and sends the numbers stored in the memory into a JavaScript Object Notation (JSON) data format which can then then be easily processed by a JavaScript engine. The projects intent is to send processed data to the user's web bowser for further visualization. For this to happen however, there must be a medium between the Arduino and the browser which can access the serial port on the computer and gather the data, however, a browser does not or has limited capabilities of accessing the hardware of the computer. To facilitate this, a p5.js library that runs on node.js, which is a cross-platform runtime environment that acts a server, effectively reads the JSON formatted data from the serial port and serves it to the browser so the data can then be manipulated and displayed (see Figure 5). Once the data is effectively parsed in the browser, d3.js and p5.js JavaScript libraries are used to manipulate the data and display. in.fORM is available online at https://github.com/mixania/in.fORM.

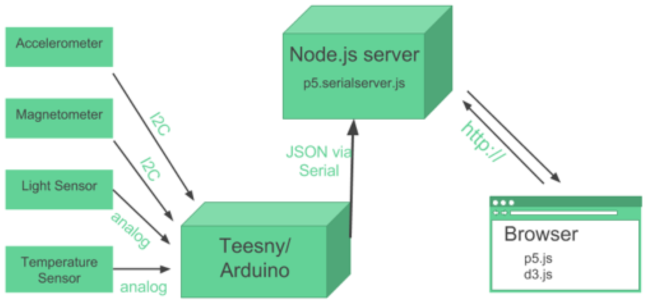

Figure 5. Shows the interaction between the hardware and how different technologies are used.

OBSERVATIONS

As of the alpha version of in.fORM (see https://github.com/mixania/in.fORM/tree/alpha) the project is only capable of displaying events as circles on the screen draws their size based on the value of their *event significance* (see Figure 4). Although this capability of this project is far from its ideal goal, it does demonstrate the basic essence of the functionality that the project is trying to achieve. The capability of making ORM relations between events is currently not developed to any significant extent, but some other goals relating to the design process where achieved. Some of the achievements and drawbacks of the alpha version of in.fORM are the following:

1. The project takes readings from the environment and parses them into events.
2. The projects visualizes these events in a simple way.
3. The project established a good foundation for further development of the project
4. From a user intractability standpoint, the project creates a will to engage with the visualized data.
5. Although the project gives some limited advantage of visualizing data, it does not draw connections between events on its own.

Gains from Iteration and Feedback

The first version of this project is perhaps the only functional prototype that has been created so far. The goals of this project where quite ambitious and almost impossible to incorporate under one whole project. Regardless, the results that the project achieved are quite compelling if inspected under closer detail. These results mainly concern the effective development of the concept rather than achieving a technical milestone. Going through both of these, it is has become possible to look at the project's drawbacks from a better perspective. Some good feedback was also given regarding the project which allows to develop the project towards a more concrete direction as opposed to relying on another unsuccessful iteration.

Roadmap for further Improvement

To make further improvements to the project, it is necessary to add more sophisticated ways of visualizing the data before considering it an advantageous data visualization tool. There are difficulties in making visualizations with the JavaScript libraries but once that is overcome, it will be also important to incorporate a basic relation analyses between the events with the application of David Hume's *necessary connexion* theory. This will most likely be a the most important area of further development. Improving the way events are processed through data is also be another area of improvement since it will allow a more precise way of mapping real life physical occurrences. There was also some feedback regarding the possible use of more sophisticated data processing techniques of such as Machine Learning. Although this project's goal is to refrain from the application of such complex data manipulation techniques, the project might proceed without such a complex data manipulating application for its next iteration. However, it might become inevitable to develop some sort of Machine Learning data processing for the 1.0 iteration of the project.

CONCLUSION

The alpha version of in.fORM has effectively demonstrated the intention of the project but has not effectively demonstrated its concrete functionality. The principles developed seem to be in place with the technical developments of the project. With the first iteration of the project, the roadmap for further development is clearer than it was before.