Knowledge Acquisition System based on JSON Schema
Implementation of a HCI for Actuation of Biosignals Acquisition Systems

Nuno Costa[1], Tiago Araujo[1,2], Neuza Nunes[2] and Hugo Gamboa[1,2]
[1]*CEFITEC, Departamento de Física, FCT, Universidade Nova de Lisboa, Lisbon, Portugal*
[2]*PLUX - Wireless Biosignals, S.A., Lisbon, Portugal*

Keywords: Knowledge Acquisition System, Human Computer Interaction, Ontology, Schema Language.

Abstract: Large amounts of data, increasing every day, are stored and transferred through the internet. These data are normally weakly structured making information disperse, uncorrelated, non-transparent and difficult to access and share. Semantic Web, proposed by the World Wide Web Consortium (W3C), addresses this problem by promoting semantic structured data, like ontologies, enabling machines to perform more work involved in finding, combining, and acting upon information on the web. Pursuing this vision, a Knowledge Acquisition System was created, written in JavaScript using JavaScript Object Notation (JSON) as the data structure and JSON Schema to define that structure, enabling new ways of acquiring and storing knowledge semantically structured. A novel Human Computer Interaction framework was developed with this knowledge system, enabling the end user to, practically, configure all kinds of electrical devices and control them. With the data structured by a Schema, the software becomes robust, error – free and human readable. To show the potential of this tool, the end user can configure an electrostimulator, surpassing the specific use of many Electrophysiology software. Therefore, we provide a tool for clinical, sports and investigation where the user has the liberty to produce their own protocols by sequentially compile electrical impulses.

1 INTRODUCTION

The easiness of access, storage, transmission of data and the exponential proliferation of internet users enclose new complexities: authentication; scalable configuration management; security; huge masses of high dimensional and often weakly structured data (the main problem addressed in this project). Structuring data pursued by Semantic Web leaves many opportunities open because there is always room for improving and to develop more adequate languages. Methods and approaches to solve the problem emerged from research in Human-Computer Interaction (HCI) (Prabhu, 1997), Information Retrieval (IR), Knowledge Discovery in Databases and Data Mining (KDD) (Fayyad et al., 1996). These methods assist end users to identify, extract, visualize and understand useful information from data.

The establishment of ontologies is one of the methods to solve the problem, possible with Semantic Web, holding much promise in manipulating information in ways that are useful and meaningful to the human user. Ontologies are collections of information with specific taxonomies and inference rules to define relations between terms. A taxonomy defines classes of objects and relations among them, and inference rules provide advanced ways of relating information by deduction. Classes, subclasses and relations among entities are a very powerful tool for Web use. We can express a large number of relations among entities by assigning properties to classes and allowing subclasses to inherit such properties. The structure and semantics provided by ontologies make it easier for an entrepreneur to provide a service, making its use completely transparent. Ontologies can enhance the functioning of the Web in many ways, like relating information on a Web page to the associated knowledge structures and inference rules, thus creating robust and clean applications (Berners-Lee et al., 2001).

Mechanisms to specify ontologies have spring in the late years, the Schema Language is one of them. Given the fact that Xtensible Markup Language (XML) allows users to add arbitrary structure to their documents, but lacks in describing the structures, Schema was developed as a notation for defining a set of XML trees (Thompson et al., 2000). From this concept, a set of specifications were established

Costa N., Araujo T., Nunes N. and Gamboa H..
Knowledge Acquisition System based on JSON Schema - Implementation of a HCI for Actuation of Biosignals Acquisition Systems.
DOI: 10.5220/0004166702550262
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications and Wireless Information Networks and Systems* (WINSYS-2012), pages 255-262
ISBN: 978-989-8565-25-9
Copyright © 2012 SCITEPRESS (Science and Technology Publications, Lda.)

to create a Schema for JavaScript Object Notation (JSON), a self descriptive language for data storage and transmission, enabling the description of the data structure. A useful Schema notation must have some specific properties: identify most of the syntactic requirements that the documents in the user domain follow; allow efficient parsing; be readable to the user; concede limited transformations corresponding to the insertion of defaults; be modular and extensible to support evolving classes (Klarlund et al., 2000). This language aids in the creation of structured data and automated tools to present the data in a human readable form, making easier the extraction and visualization of useful information from data. Therefore, in this field of structuring data, Schema largely supersedes Document type Definitions (DTDs) for markup language.

One example of this procedure is Protégé, an open source Ontology Editor and Knowledge Acquisition System. Protégé is a framework written in Java and uses Swing (Java GUI widget toolkit) to create complex user interfaces. These interfaces provide the user with tools to construct domain models and knowledge-based applications with ontologies. As a graphical tool for ontology editing and knowledge acquisition, it can adapt to enable conceptual modelling with new and evolving Semantic Web languages (Noy et al., 2001). Protégé lets us, like the Schema, create domains in a conceptual level without having to know the syntax of the language ultimately used on the Web to create interfaces, passing information, between other features. We can concentrate on the concept types (integers, arrays, strings, ...) and relationships in the domain and the facts about them that we need to describe.

Using this mechanism in the project for structuring the data, a Human Computer Interaction was created with an innovative Knowledge Acquisition system for controlling the actuation of a Biosignals Acquisition System. One of the purposes, and as a practical example, is to control an electrostimulator enabling the user to create their own protocols, pursuing a non-specific software for Electrostimulation. Therefore, it allows the user to employ the software in different types of Electrostimulation applications:

- **Electrotherapy:** Rehabilitation (von Lewinski et al., 2009); Spinal cord injury, stroke, sensory deficits, and neurological disorders (Cogan, 2008); elicit electronarcosis, electrosleep and electroanalgesia (Lebedev et al., 2002);
- **Physical Conditioning:** fitness; active recovery; optimizing physical performance by improvement of maximum strength of a muscle (muscular tonus) in less time (Siff, 1990).

This work presents a novel Knowledge Acquisition System based on JSON Schema with the specific focus on creating user interfaces to configure biosignals acquisition devices, and with this information control the actuation of that device. Ultimately, the software will pursue usability and acceptability, because ease of use affects the users performance and their satisfaction, while acceptably affects whether the product is used or not (Holzinger and Leitner, 2005). Our proposal defines Application Programming Interfaces (APIs) and low-level infrastructures to create a system for controlling a biosignals acquisition device, where the acquisition and actuation parameters are acquired from the user. At the core of this system is a JSON Schema enabling validation (data integrity), interaction (UI generation - forms and code) and documentation.

2 DATA STRUCTURE

2.1 JSON

JSON is a simple, lightweight and human readable text-data structure for information exchange. The approach for information exchange is simpler than XML, by the less verbose structure of the notation. Interpreting JSON is native in some languages with the existence of several support libraries that make JSON a platform independent language (Crockford, D., 2006). JSON structure is composed of name/value pairs separated by comma, curly brackets holds objects and square brackets holds arrays. Values can be numbers, strings, booleans, arrays, objects and null. In the example below is an object containing information of an address and phone number:

```
{"address":{
     "streetAddress": "21 2nd Street",
     "city":"New York"
 },
 "phoneNumber":
 [{
     "type":"home",
     "number":"212 555-1234"
 }]
}
```

Considering these features, JSON was selected as the data structure of this work, and is defined by JSON Schema.

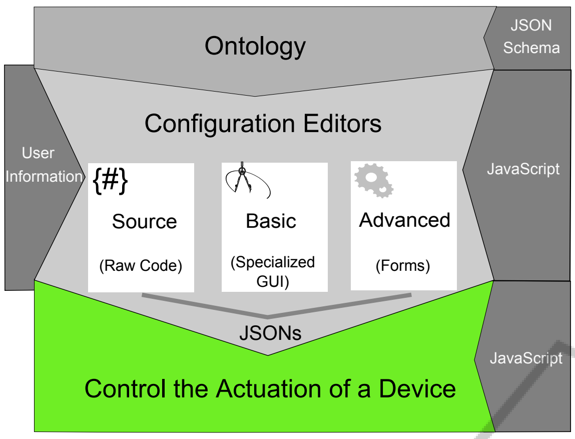


Figure 1: Generic diagram showing the flow of information.

2.2 JSON Schema

A JSON Schema is a Media Type (standard draft of options) that specifies a JSON-based format to define the structure of JSON data, providing a contract (set of rules) required in a given application, and how to interact with the contract. Accordingly, JSON Schema specifies requirements for JSON properties and other property attributes with the following intentions:

- Validation (data integrity);
- Documentation;
- Interaction (UI generation - forms and code);
- Hyperlink Navigation.

JSON Schema is also a JSON with a compact implementation and can be used on the client and server. Specifications are organized in two parts (Zyp, K., 2011):

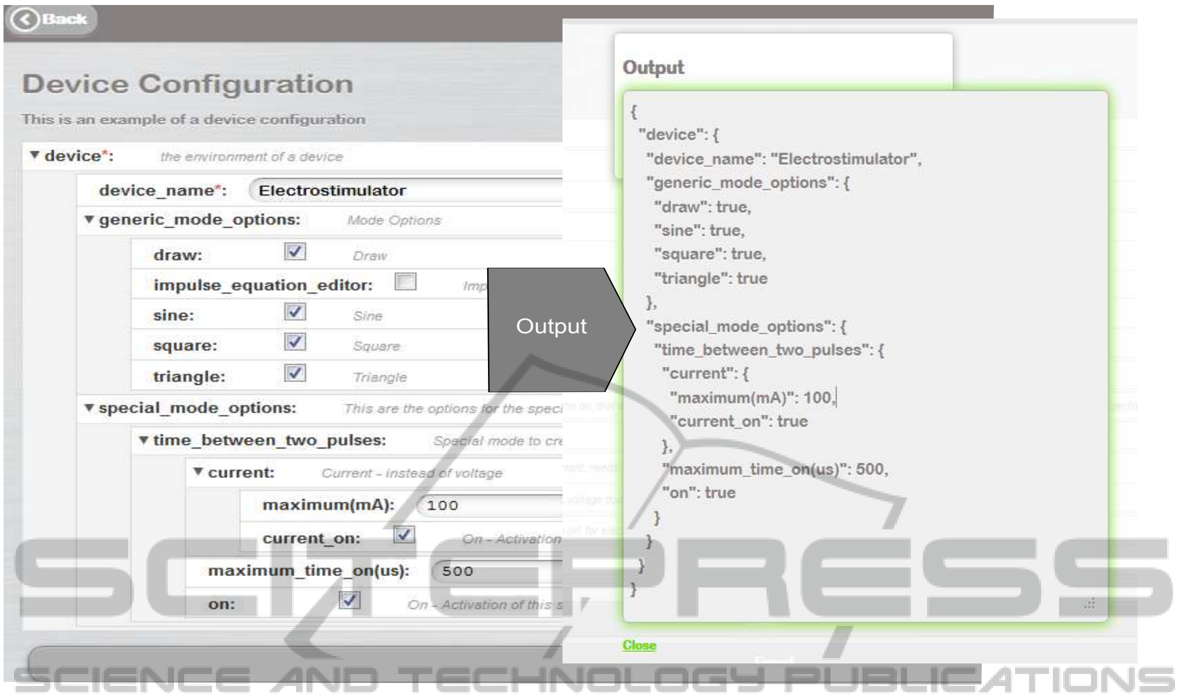
- **Core Schema Specification:** primary concerned with describing a JSON structure and specifying valid elements in the structure.
- **Hyper Schema Specification:** define elements in a structure that can be interpreted as hyperlinks, in others JSON documents and elements of interaction (This allows user agents to be able to successfully navigate JSON documents based on their Schemas).

Below is an example of a JSON Schema defining the structure for the JSON example showed before:

```
{"type":"object",
 "required":false,
 "properties":{
    "address": {
      "type":"object",
      "required":true,
      "properties":{
        "city": {
            "type":"string",
            "required":true
        },
        "streetAddress": {
            "type":"string",
            "required":true
        }
      }
    },
    "phoneNumber": {
        "type":"array",
        "required":false,
        "items":{
            "type":"object",
            "required":false,
            "properties":{
               "number": {
                  "type":"string",
                  "required":false
                },
                "type": {
                  "type":"string",
                  "required":false
                }
            }
}}}}
```

As shown in the diagram, Figure 1, JSON Schema allows the definition of ontologies and will be in the core of the program, enabling: **interaction**, Schema serves as blueprint to architect the necessary forms, with or without Graphical User Interfaces (GUI), and define the other editors for the Knowledge Acquisition System; **documentation**, APIs and low-level software infrastructures in JavaScript enable the transformation of the information acquired from the user in a JSON data structure defined by the Schemas, then the data (JSONs and Schemas) is stored in the server and retrieved to the client (JavaScript) through websockets (full-duplex communications channels over a single TCP connection); **validation**, JSON data can be imputed directly in a raw editor (a knowledge acquisition system) by the end user. Nevertheless, the data is only stored if the structure agrees with the Schema.

With these mechanisms the software becomes a powerful HCI system: robust; error - free; with human readable data structures, therefore, visualization and extraction of information is easier.



```
▾ device*:                 the environment of a device
    device_name*:          Electrostimulator
  ▾ generic_mode_options:  Mode Options
      draw:                 ☑   Draw
      impulse_equation_editor: ☐  Imp
      sine:                 ☑   Sine
      square:               ☑   Square
      triangle:             ☑   Triangle
  ▾ special_mode_options:  This are the options for the speci
    ▾ time_between_two_pulses:  Special mode to cre
      ▾ current:           Current - instead of voltage
          maximum(mA):      100
          current_on:       ☑   On - Activation
        maximum_time_on(us):  500
        on:                 ☑   On - Activation of this s
```

```
Output
{
 "device": {
  "device_name": "Electrostimulator",
  "generic_mode_options": {
   "draw": true,
   "sine": true,
   "square": true,
   "triangle": true
  },
  "special_mode_options": {
   "time_between_two_pulses": {
    "current": {
     "maximum(mA)": 100,
     "current_on": true
    },
    "maximum_time_on(us)": 500,
    "on": true
   }
  }
 }
}
```

Figure 2: Configuration in Advanced Editor the mode options for an electrostimulator. Initial version of the software.

3 HUMAN COMPUTER INTERACTION

Based on the method for data structure, described in the last section, a Human Computer Interaction was developed with a novel Knowledge Acquisition system for controlling the actuation of Biosignals Acquisition Systems. As shown in Figure 1, an ontology is defined with the help of JSON Schemas, then these Schemas are interpreted in JavaScript enabling the conception of three editors. These editors provide different means to acquire the information from the user. Afterwards, this information is saved as JSONs in the server. In the control, the JSONs are retrieved from the server, through websockets, and parsed in JavaScript. Finally, the necessary information is sent to the device via APIs. As a practical example, the HCI provides mechanisms to control an electrostimulator by enabling the user to configure protocols/sessions for actuation of the device.

The HCI is integrated in a software for biosignals acquisition and processing from PLUX - Wireless Biosignals, S.A., enabling the control of a generic device. In this way, actuation, acquisition and processing is possible in a closed-loop cycle (Broderick et al., 2008).

The main purpose of this project is to provide the end user the liberty to create and sequentially compile electrical impulses. This enables the user to create actuation sessions for electrical devices. In a particular example, enables the user to configure an electrostimulator and design sessions to test different types of Electrostimulation protocols.

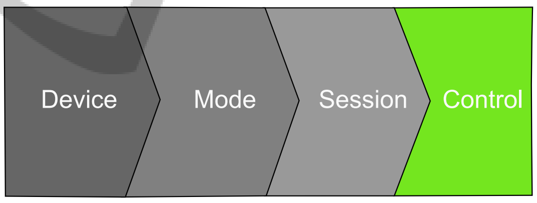
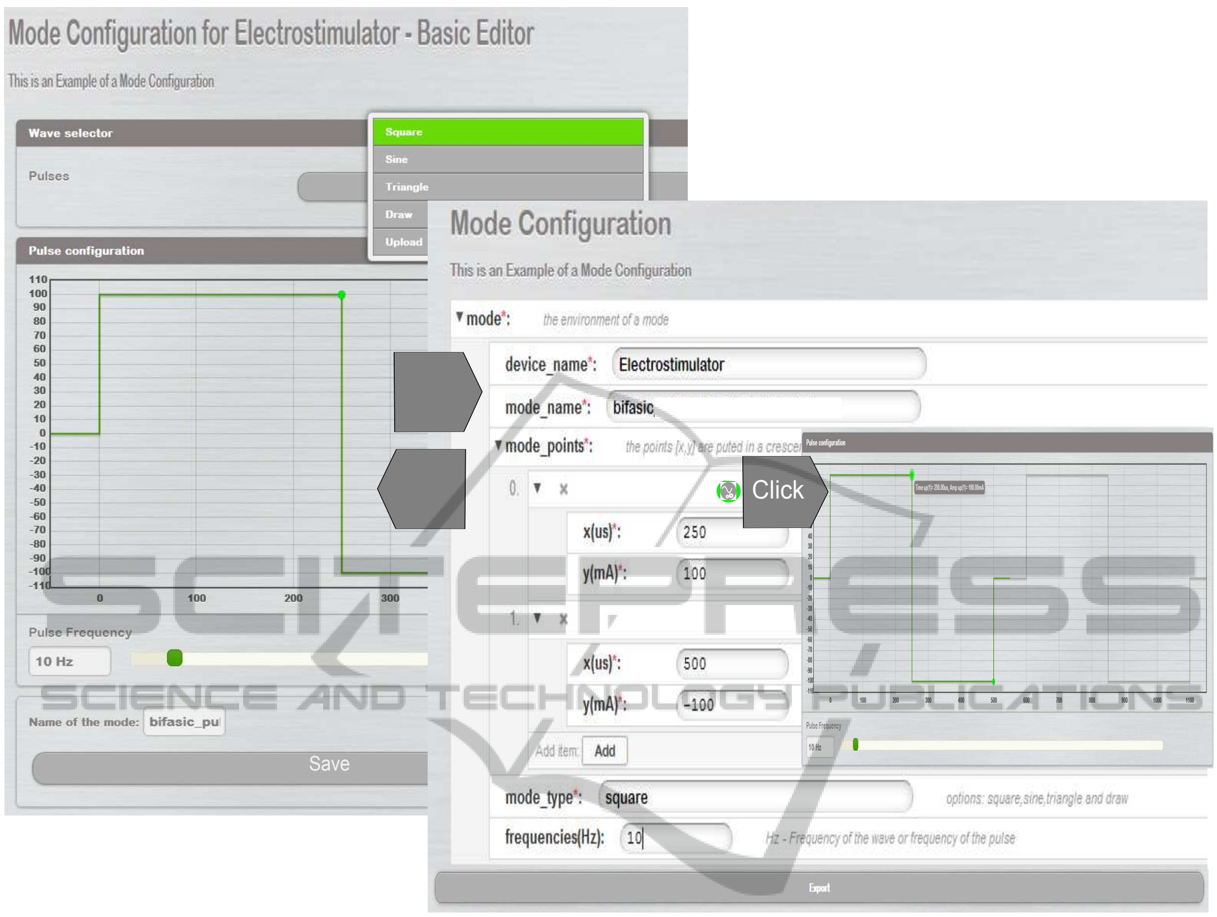


Figure 3: Diagram representing the direction to configure then control. First, configure a new device, then modes and finally the sessions using the modes created.

The software is divided in configuration (the knowledge acquisition system based on JSON Schemas), and control for the actuation of a device. Basically, as shown in Figure 3, to control we need to configure a device, and create temporal sessions, i.e sequence of impulse modes, for each channel of that device. Consequently, the configuration is divided in device, mode and session, each one with the respective JSON Schema to structure the data, and directly or indirectly create the editors. In the next sections, these ideas are explained in more detail accompanied with images for the specific case of an electrostimulator.


Mode Configuration for Electrostimulator - Basic Editor

This is an Example of a Mode Configuration

Wave selector: Square / Sine / Triangle / Draw / Upload

Pulses

Pulse configuration

Pulse Frequency: 10 Hz

Name of the mode: bifasic_pul

Save

Mode Configuration

This is an Example of a Mode Configuration

mode*: the environment of a mode

device_name*: Electrostimulator

mode_name*: bifasic

mode_points*: the points [x,y] are puted in a crescent

0. x(us)*: 250 y(mA)*: 100

1. x(us)*: 500 y(mA)*: -100

Add item: Add

mode_type*: square — options: square,sine,triangle and draw

frequencies(Hz): 10 — Hz - Frequency of the wave or frequency of the pulse

Export

Figure 4: Configuration of the mode in Basic Editor and Advanced Editor. Initial versions of the software.

3.1 Configuration

3.1.1 Device

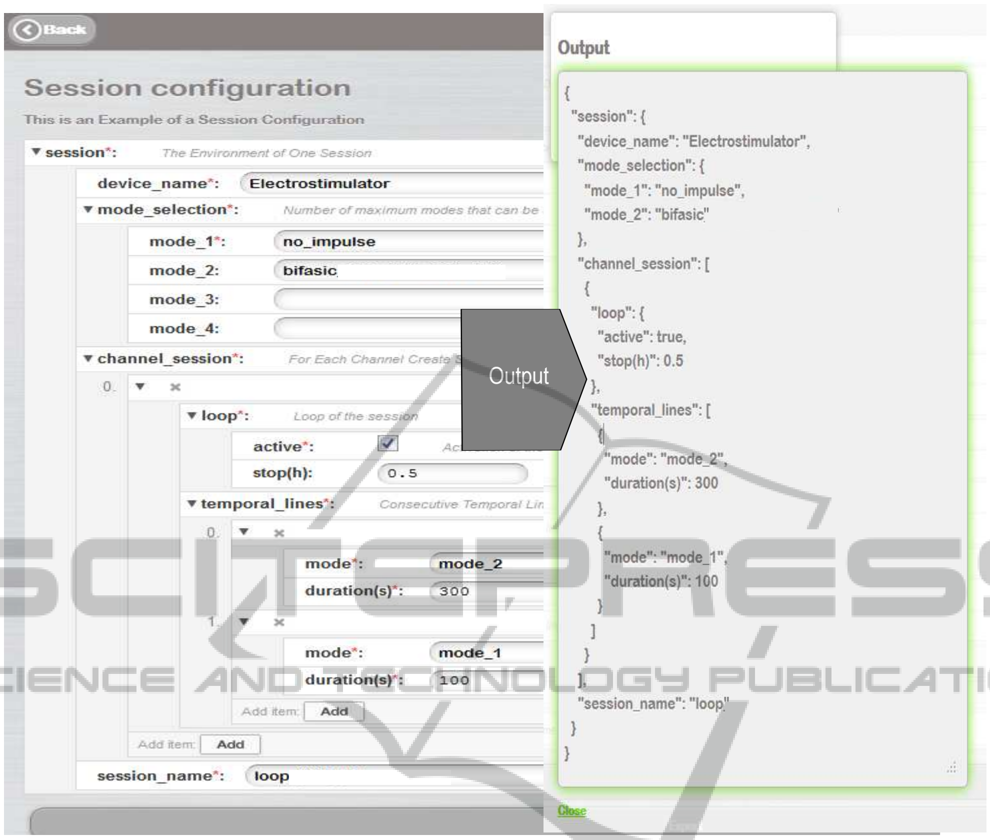
The user can compose a new device by setting the mode options for that device. There are two types of options, generic and special. Special stands for special modes, in this case the special mode enables the creation of pulses instead of only periodic waves, unlike a generic wave generator. For example, the special mode is necessary to configure an electrostimulator. We can see in Figure 2 the configuration with the special parameters for Electrostimulation: maximum current, 100 mA, and maximum time on, 500 µs. After fill in the form, created directly from the Schema, a JSON is exported and stored in the server.

3.1.2 Mode

The user can project a new mode for a specific device. The mode permits the configuration of pulses (special mode options activated), and waves (special mode options deactivated) depending on the mode options. In Figure 4 is possible to view how a mode can be configured in two different editors (described in the editors section). The figure shows the creation of bifasic mode for the electrostimulator, a pulse with positive and negative time on of 250 µs, amplitude up 100 mA and amplitude down -100 mA, and 10 Hz of pulse frequency.

3.1.3 Session

The user can create a new session for a specific device. A session is a sequence of modes, and each channel of a device can have one session programmed. A device has a maximum number of programmable modes, but they can be repeated infinitely. Figure 5 shows the configuration of a session. Primarily, the definition of the modes that will be used, four is the maximum that the hardware can memorize. Afterwards, the creation of a session for the channels required, in this case, only one channel is programmed with two modes, bifasic, 300 seconds activated and, no_impulse, during 100 seconds. This sequence is repeated during half an hour by activating the loop for



Session configuration

This is an Example of a Session Configuration

```
▾ session*:                 The Environment of One Session
    device_name*:           Electrostimulator
    ▾ mode_selection*:      Number of maximum modes that can be
        mode_1*:            no_impulse
        mode_2:             bifasic
        mode_3:
        mode_4:
    ▾ channel_session*:     For Each Channel Create
      0. ▾ ×
        ▾ loop*:            Loop of the session
            active*:        ✓
            stop(h):        0.5
        ▾ temporal_lines*:  Consecutive Temporal Lin
          0. ▾ ×
            mode*:          mode_2
            duration(s)*:   300
          1. ▾ ×
            mode*:          mode_1
            duration(s)*:   100
          Add item: Add
      Add item: Add
    session_name*:          loop
```

Output

```
{
 "session": {
  "device_name": "Electrostimulator",
  "mode_selection": {
   "mode_1": "no_impulse",
   "mode_2": "bifasic"
  },
  "channel_session": [
   {
    "loop": {
     "active": true,
     "stop(h)": 0.5
    },
    "temporal_lines": [
     {
      "mode": "mode_2",
      "duration(s)": 300
     },
     {
      "mode": "mode_1",
      "duration(s)": 100
     }
    ]
   }
  ],
  "session_name": "loop"
 }
}
```

Figure 5: Configuration of a session in Advanced Editor. Loop during half an hour with bifasic mode, 300 seconds activated, and no_impulse 100 seconds activated. Initial version of the software.

that specific session. The session is named protocol1.

3.1.4 Editors

The core for each configuration stated before is a Schema, and the end user has the possibility to choose between three editors (like in Figure 1):

- Basic Editor (Specialized GUI): simple and user-friendly interface with the minimum required fields to fill. The information is saved in a JSON format defined by the Schema. This editor is intended for the basic users. It can be seen in Figure 4.
- Advanced Editor: a form generated automatically from the Schema specifications, with the possibility of edition in GUI. Information is also saved in JSON format defined by the Schema. The most profitable editor because it shows directly how the information will be saved in the JSON. This way, users understand the architecture of the data. Figures 2, 4 and 5 have the configuration of a device, a mode and a session, respectively, in the Advanced Editor.
- Source Editor: an editor to upload or create JSONs, where the user writes the JSON (like the JSONs exported in figures 2 and 5) or copy and paste the JSON in the editor. JSONs are valid and can be saved if they agree with JSON Schema, if not, reports with errors are generated. This Editor is intended for users with more knowledge of the data structure, or for the ones that just want to copy and paste information or upload.

  Note that it's possible to travel between the three editors, and the information is inherited between them. Also, in Advanced Editor, GUIs are directly related with special types from the JSON Schema, and can be used to fill the necessary fields. Like in the example Figure 4, clicking on the specific button opens a graphic tool (canvas) for edition of points by moving them with the mouse, identical to the canvas from Basic Editor.

3.2 Control

The control is the area for actuation and acquisition of biosignals. In here, the user first needs to set up the device, for that he must choose the device and the ses-

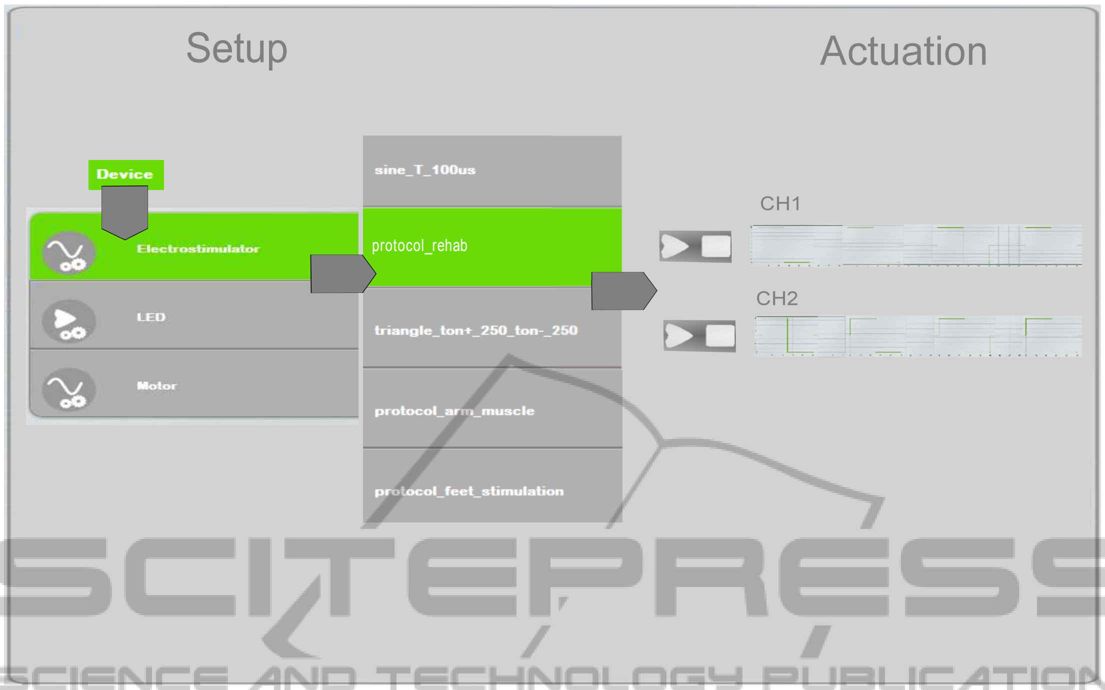


Figure 6: Actuation of an electrostimulator with the session protocol_rehab. This session has two channels with different sessions configured.

sion to program. When the device is ready, start/stop of the session is feasible. In the course of the session is possible to acquire real time biosignals and synchronize them with the session. This part is in development stage: Figure 6 presents a moke-up of what could be in the final stage. The control area will have many device sessions templates. Therefore, a quick approach by the user will enable the simple selection of a pre-defined session and start the actuation immediately.

4 CONCLUSIONS

This project envisions a software with powerful capabilities. The most important parts are the infrastructures that conduct the flow of information between the Schema, the editors, the GUIs and finally the process of saving in JSON data. After the APIs and low-level infrastructures are programmed is easy and fast to create information editors with other Schemas, develop configuration interfaces for that editors and allow the engineering of new human computer interactions. This explains the impressive usability of this knowledge acquisition system. For this reason, is important to refute that the main mechanisms behind this software are JSON, the human-readable data structure, and JSON Schema that defines the structure of JSON, allowing interaction (creation of forms with specialized GUIs), documentation and validation, producing an error-free, semantically structured and human-readable knowledge acquisition system, contributing to the HCI system robustness.

In Biomedicine, the configuration part should only be manipulated by clinical professionals, where they can adapt sessions to each patient. Overly complicated user interfaces and large, bulky designs can deter patients from using the device on a day to day basis, so, ultimately, the patient will only start or stop a session prepared for him. Following this idea the software was separated in configuration, where clinical professionals can configure a device with sessions for each channel, and control, where the user just need to choose the device and session (set up the device) then start the session. If for some reason, the session should not stop automatically (before the stop time is reached), the user can stop the session. This HCI is being tested in the Electrostimulation area to support clinical professionals, sportsman and investigators to control electrostimulators by enabling the creation of different kind of protocols, empowering the software usability.

This software, above all, pursues usability, for this reason, in future stages of the project, the following tasks will be conducted: user studies, extended unit tests, usability tests, and usability expert evaluation.

REFERENCES

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American Magazine*.

Broderick, B., Breen, P., and ÓLaighin, G. (2008). Electronic stimulators for surface neural prosthesis. *Journal of Automatic Control*, 18(2):25–33.

Cogan, S. (2008). Neural stimulation and recording electrodes. *Annu. Rev. Biomed. Eng.*, 10:275–309.

Crockford, D. (2006). The application/json media type for javascript object notation (json).

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Holzinger, A. and Leitner, H. (2005). Lessons from real-life usability engineering in hospital: from software usability to total workplace usability. *Empowering Software Quality: How can Usability Engineering reach these goals*, pages 153–160.

Klarlund, N., Møller, A., and Schwartzbach, M. (2000). Dsd: A schema language for xml. In *Proceedings of the third workshop on Formal methods in software practice*, pages 101–111. ACM.

Lebedev, V., Malygin, A., Kovalevski, A., Rychkova, S., Sisoev, V., Kropotov, S., Krupitski, E., Gerasimova, L., Glukhov, D., and Kozlowski, G. (2002). Devices for noninvasive transcranial electrostimulation of the brain endorphinergic system: Application for improvement of human psycho-physiological status. *Artificial organs*, 26(3):248–251.

Noy, N., Sintek, M., Decker, S., Crubézy, M., Fergerson, R., and Musen, M. (2001). Creating semantic web contents with protege-2000. *Intelligent Systems, IEEE*, 16(2):60–71.

Prabhu, P. (1997). *Handbook of human-computer interaction*. North Holland.

Siff, M. (1990). Applications of electrostimulation in physical conditioning: a review. *The Journal of Strength & Conditioning Research*, 4(1):20.

Thompson, H. et al. (2000). Xml schema. w3c working draft, may 2001.

von Lewinski, F., Hofer, S., Kaus, J., Merboldt, K., Rothkegel, H., Schweizer, R., Liebetanz, D., Frahm, J., and Paulus, W. (2009). Efficacy of emg-triggered electrical arm stimulation in chronic hemiparetic stroke patients. *Restorative Neurology and Neuroscience*, 27(3):189–197.

Zyp, K. (2011). A json media type for describing the structure and meaning of json documents.