Medbiq xAPI Workshop Report

Authors:

David Topps[1], Corey Albersworth[2], Ellen Meiselman[3], Maureen Topps[1], Sarah Topps[4], Sandra Morrison[1]

[1] University of Calgary
[2] Athabasca University
[3] University of Michigan
[4] Simon Fraser University

Background:

How do we know our learners are competent? Upon what do we base this? Observations and judgments from teachers, using ITERs and MCQs form the basis of the vast majority of our assessments. While these work well for the majority of learners and teachers, we are less effective at early detection of maladaptive learners or those who are struggling. We tend to give the benefit of the doubt, thinking this a kindness, but delayed detection and diagnosis of learners in difficulty is doing nobody a favor.

In our high school math tests, we are concerned that little Johnny not only got the right answer but used the appropriate steps and techniques in deriving that answer. Similarly, in medical education, we should be concerned with assessment of more than memorization, and look at problem solving and efficient, effective task completion in a variety of contexts. We should look at what our learners do, not what they or their teachers say they do. Hence, the rise in interest in activity metrics. This concept is at the peak of Miller's Pyramid of Assessment (Miller, 1990)demonstrating professional authenticity, moving from "knows" to "knows how" to "shows how" to "does" (although doubt has been raised about the evidence to support Miller's pyramid, but we digress(Lalley & Miller, 2007)).

This is not a new concept, but our abilities to measure such activities with the necessary richness of detail has largely been compromised by cost and practicality and hence, we rely on compound observations and judgments of human sensors – our teachers. But now, with the plethora of cheap devices and sensors, the possibilities for gathering meaningful data about the learning process have greatly changed.

In our previous studies, using mixed simulation modalities in our Health Services Virtual Organization (HSVO) project, we were able to combine multiple learning tools and systems, using a sophisticated network-enabled platform. We combined virtual patients, high-fidelity simulators, low-latency video collaboration, virtual anatomy models, and light-field camera arrays to generate virtual points of view. We were able to create powerful learning designs for collaborating groups of learners across multiple disciplines, institutions, and continents, in real-time with many concurrent data streams tracking their every action.

Both learners and teachers enthusiastically adopted these multi-modality simulation designs. We observed a progression of skills development and collaborative problem solving approaches in all the groups. Analysis of the activity streams, which was largely based on manual coding of videotaped activities, proved to be extremely laborious, with a high cost in effort. We did observe intense clusters of activity, centered on boundary changes in context, somewhat analogous to phase changes in simpler biological and physical processes. The vigorous up-spike in such activities was often hard to keep up with, and sometimes missed by bored assessors skipping through long periods of minimal state change. Humans do not do well with this approach – machine sensors are much more reliable and efficient in such monitoring.

The HSVO project demonstrated some extraordinary learning activities but was a very costly endeavor, with high infrastructure needs (user-directed light pipes, high performance cluster computing), highly skilled personnel, and dependence on high levels of technology with limited portability. While the core middleware layer, Savoir, was designed to act as an intermediary between many different device types and systems, it was not very flexible and required an overly tight degree of integration and binding between Application Programming Interfaces (APIs).

In many ways, this was similar to the problems encountered with attempting to blend information streams across other educational platforms using SCORM. It was a bold attempt at the time to integrate data from many sources with the core Learning Management Systems (LMSs). Its was dependent on the central structure of the LMS - an architectural model which is not particularly suitable to curricular structure of most medical schools. Its restrictive data structures made SCORM vulnerable to all the ongoing changes at each side of every system interface.

This very challenge, facing the adopters of SCORM, drove the development of the Experience API (xAPI, also known as the Tin Can API). Commissioned by Advanced Distributed Learning (ADL), it was originally developed by Andrew Downes and team at Rustici Software. xAPI has a much simpler, more flexible and open structure based on Subject-Verb-Object triplets that are similar to the Resource Description Framework (RDF).

To promote the advance of data interoperability standards in this area of activity metrics, the Medbiquitous Learning Experience Working Group was formed. This group chose xAPI as their central mechanism to coordinate and connect such activity metrics, but is also monitoring other similar protocols. The group has been working with key players and organizations relevant to xAPI, with academic, government and industry representation.

This report describes the activities within and leading up to an intense workshop about xAPI and blended learning at the Medbiq annual conference in Baltimore, in May 2016. This report assumes some familiarity with the principles and processes surrounding the use of the Experience API. There are many other excellent articles out there explaining these basic principles better than we can in the space available here.

This report also does not follow the traditional IMRAD layout of a single research intervention. It is intended to be descriptive of the design-based research approach that we took in our development of the tools and resources that culminated in our workshop. See Discussion for more on this.

Developing xAPI Profiles

The Experience API is simple in concept: the Subject-Verb-Object construct has been described as:

Bob Did This

But, in order for the xAPI statements to be meaningfully compared across sources, when they are stored in the Learning Records Store (LRS), there needs to be some standardization around the vocabularies used. Within this simple structure, standardization of the Verbs used, along with their meaning and context is the first thing to tackle.

Groups of verbs, and other usage requirements, that are associated with a common set of activities are combined into sets known as Profiles. The Medbiq Learning Experience Working Group has taken on the task of defining a series of Profiles to support medical education activities such as simulations and scenarios. The group chose to start with the Virtual Patient xAPI Profile (Topps, Meiselman, & Strothers, 2016), as there already exists some excellent base work around the Medbiq Virtual Patient data standards, and the related activities are relatively well constrained with good commonality across platforms.

The group plans to continue to define a series of Profiles, relating to mannequins and task trainers, standardized patients, scenarios and blended simulations, and virtual worlds, in collaboration with other Medbiq Working Groups such as Competencies, along with other partners and interested organizations.

Developing a standard and designing the profiles that describe it requires a core structure and theoretical design, but if this is not grounded in the practicalities of implementation, it is easy for the standards to become isolated and esoteric. Accordingly, Medbiq has been very active at promoting learning sessions and workshops around xAPI over the past 3 years. For this year's workshop (Meiselman, Topps, & Albersworth, 2016), we assembled a learning design that would integrate a broad variety of learning activities, with activity metrics derived from multiple concurrent sources.

This endpoint and timeline greatly facilitated a more efficient and collaborative approach. It also generated much interest in the xAPI community and we were fortunate beneficiaries of many collaborative activities and problem solving, far exceeding our expectations. Frankly, we were astounded by how helpful everybody was in moving things forwards. This is definitely a great strength of working with xAPI at present.

Developing a workshop

Based on our experiences with the HSVO Project, and its power in combining multiple simulation modalities into effective learning scenarios, we created a simple scenario that would demonstrate a wide variety of activity metrics from multiple concurrent sources.

To contrast with the complexity and cost of the HSVO Project, we also chose to base our learning design on much simpler and cheaper devices. HSVO cost more than $2 million; the average "disposable income" for an educational research project is more like $5-10k, so we wanted to demonstrate what is possible on a much more restricted budget.

In particular, we were interested in exploring what could be done with the cheap sensors and simple computing cores available on the Arduino platform, which is designed for hobbyist use and very low budgets. There are other similar devices such as Raspberry Pi, which also have great promise, but a quick environmental scan showed Arduino to be the most suitable to our needs.

In our HSVO Project, we found the open-source virtual patient platform, OpenLabyrinth, to be a very effective integrator of simulation related activities. It was excellent at providing flexible, easily modifiable "contextual glue" to hold the various components of a learning scenario together. Tight coupling, using the Savoir middleware, proved too inflexible in the HSVO Project. The more open approach afforded by xAPI was more promising.

Developing a Quiver of xAPI sources

This entailed the incorporation of xAPI into OpenLabyrinth. We were fortunate in two ways: we had secured funding via a Catalyst Grant from the O'Brien Institute of Public Health at the University of Calgary to extend OpenLabyrinth with improved activity metrics and enhanced integration with other research platforms; and the inherent architecture of OpenLabyrinth, which already contained a useful set of internal activity metrics, and was highly conducive to integration of xAPI statement output. We were pleased to see that this was possible with less development time than we had budgeted for, and in a shorter timeline than anticipated because the existing data structures were so good.

We based the xAPI statement formatting on the Medbiq xAPI Virtual Patient Profile, and also took the opportunity to reflexively improve the Medbiq profile by considering some of the practical implications as they pertained to our cases and scenarios. This two-way reciprocal development was carried out with full collaboration with other working group members and the xAPI community. All source code is fully documented on Github at https://github.com/olab/Open-Labyrinth and we are happy to communicate with other development groups about some of the small challenges that we faced in our implementation.

While we were exploring other ways in which we could capture activity metrics in our scenario, using a number of different collection mechanisms, we came across the

GrassBlade xAPI Companion (www.nextsoftwaresolutions.com/grassblade-xapi-companion), a simple useful utility that provides xAPI statements from WordPress activities. Since we already use WordPress for our OpenLabyrinth support site and blog, this presented an ideal opportunity to extend our use of web platforms for additional data gathering and activity monitoring.

We then implemented the GrassBlade LRS on our servers. We were pleased to note that the GrassBlade LRS also uses MySQL for its internal database, as do both OpenLabyrinth and WordPress. This greatly simplified some of the integration and testing of data transfer between these software applications. We had been quite concerned about what type of database infrastructure to use, as our initial readings on the topic of LRSs seemed to suggest that a noSQL database, such as Mongo DB, was desirable (Abbey, 2016)(Kaplan, 2014)(Korneliusz, 2014; Mei, 2013). Since our development teams were much more familiar with SQL, this presented a significant potential barrier.

There was a third factor which we found attractive for choosing GrassBlade as our primary LRS for testing. There is a flat pricing structure per LRS instance, without additional costs based on the number of statements stored. In our early phases, we had very little idea of the number of statements that we would be storing or of the size of the databases we would generate. We were alarmed to hear of another similar project at a previous Medbiq conference that managed to generate 300,000 xAPI statements within 36 hours, which created a sizeable bill – something our funders were keen to avoid.

For our initial testing, another advantage of this approach became apparent: it was simple to generate xAPI statements from our WordPress site (http://openlabyrinth.ca) and examine them in our GrassBlade LRS. This gave us greater practical familiarity with the formatting of xAPI statements and some of the syntactical rules imposed by the LRS. We were fortunate in that the GrassBlade LRS is relatively forgiving in its xAPI statement interpretation, which afforded greater experimentation with data sources in the early phases.

During our explorations of WordPress and xAPI statement generation, we also came across H5P widgets (https://h5p.org). The H5P project has been developing interactive widgets that can be incorporated into a number of platforms, such as WordPress, Moodle, and Drupal. The initial attraction is that many of these H5P widgets generate xAPI statements of their own. For this, they require a simple intermediary framework – there is one available as a WordPress plugin. This meant that we were able to implement another source of xAPI statements with less than an hour's work.

We then noted that, in addition to being open-source, H5P makes it easy to incorporate their intermediary supporting framework into one's own application. This was ideal for incorporating H5P widgets into OpenLabyrinth and this was accomplished with only a modicum of developer time, giving us yet another source of xAPI statements.

The H5P widget design interface is very simple to work with, allowing us to rapidly generate a number of interactive user interface extensions, extending the utility of both the

WordPress and OpenLabyrinth platforms. We were also delighted to discover that widgets created for one platform are easily portable to another, further multiplying our development efforts.

Developing Sensor Hardware

From these multiple experiments with generating xAPI statements from all these sources, we were then more easily able to visualize how one might generate raw xAPI statements, using some biometric sensors and the very simple Arduino hardware platform.

Initially, this was only intended to be a proof of concept demonstration of the feasibility of generating such xAPI statements and the simplicity of their formatting. The Arduino platform is renowned for being cheap and easy to work with, and already has a wide variety of cheap sensors that are largely plug and play.

In accordance with the overall learning design for the workshop, where we hoped to demonstrate the ability to measure a broad variety of concurrent activities, we selected sensors that might give us some crude indication of stress levels in the participants. We anticipated that throwing such a plethora of activities into the mix might become quite chaotic (makes for a fun workshop!) and hard to follow.

Figure 1: showing the Arduino sensors in use (left hand) while the subject played an OpenLabyrinth virtual patient. Right screen shows sensor output in real-time

For the biometric sensors, we chose a combination of heart rate and galvanic skin response (GSR). These are well known to be associated with stress levels and are, indeed, two of the parameters used in a lie detector. What surprised us in our initial testing was the sensitivity of these indicators.

With a convenience group of co-investigators in the project, we examined how the heart rate and GSR varied as they navigated a challenging OpenLabyrinth virtual patient case. For the cases, we modified two existing VP cases, Gail's Dilemma (http://demo.openlabyrinth.ca/renderLabyrinth/index/727) and Rushing Roulette (http://demo.openlabyrinth.ca/renderLabyrinth/index/723). There were already some time pressures and intensity inherent in these cases but we decided to up the ante by using a new function in OpenLabyrinth, with timer-enforced popup warning messages and navigational jumps propelling them onwards at an ever-increasing pace. The first case generally lasts about 15 minutes, whereas the second case set of 20 mini-cases can usually be completed within 5-8 minutes. This gave us sufficient time to observe changes but was short enough to be able to run multiple iterations without participant burnout.

Our group consisted of three experienced physician teachers, a public health professional with extensive virtual patient experience, and a computer science student familiar with the platform performance. We videotaped the session for later analysis, while we recorded their performance on the VP cases. We also kept independent field notes during the session for later triangulation of observations. We connected each participant to the heart rate and GSR sensors then watched the changes while they each navigated the VP cases in turn.

It was fascinating to note how the different participants responded to the stress of the cases. As expected, with their varying backgrounds, each found different elements of the VP cases to be challenging in different ways. All participants found that the increasingly tight timing was stressful or annoying. Normally, the time interval allowed for decisions on the mini-cases is 30 seconds. For this series, we started at 25 seconds, subtracted a second for each subsequent case, leading to only 6 seconds for the final case, which is barely enough time to read it, let alone make an informed decision.

Other factors in the case play were apparently stressful to each participant in different ways. Some disliked having to make decisions based on incomplete data; some found that interface quirks that arose were quite annoying. But we were easily able to detect these changes in stress level, even when quite subtle. One participant, well known for cool performance under pressure, maintained a remarkably consistent set of parameters, until the time-out crossed the point of reasonable readability for the case. As soon as a forced jump kicked in, there was change in both HR and GSR.

Two of the participants in this group were on significant doses of beta blockers for an unrelated medical condition. We anticipated that this might blunt the stress response and limit the usefulness of the sensors. While both of them had very low resting heart rates (58 and 62 beats per minute (BPM) respectively), the sensors were still able to easily detect a change in their stress levels as they challenged the cases.

We were delighted by these early findings, which were much more sensitive than we suspected. Despite the mild but intentional stressors induced in this scenario, all participants reported that the cases were engaging and the experience was enjoyable. They were intrigued to see clear demonstration that there was an apparently tight association between their mild stress levels and sensor findings.

Developing the xAPI Sensor Interface

All the steps up to this point had been fairly easy to implement and were starting to show an interesting confluence of factors and data. We anticipated that completing the interface between sensor tracking and the LRS would also be quite simple.

We enlisted the assistance of computer science student (CA) in coding the Arduino sensor interface and xAPI integration. This was to be part of a self-learning project, with recognition for course credits.

Using the Processing Integrated Development Environment (PIDE), the initial steps with the sensor and the Arduino engine were quite straightforward. Using the Arduino to send out values via the serial port to the processing PIDE with a character in front of the Value so that PIDE knew what sensor the value was coming from. PIDE would then display this with a visualization that would show the heart waveform and BPM values. Due to the limitations of PIDE, it was unable to send a properly formatted HTTP post request to the LRS with a JSON statement. We had to convert the PIDE code to the more powerful Eclipse IDE. Various additional tools were needed to get the PIDE code up and running on the Eclipse IDE, including importing all of the proper libraries from PIDE.

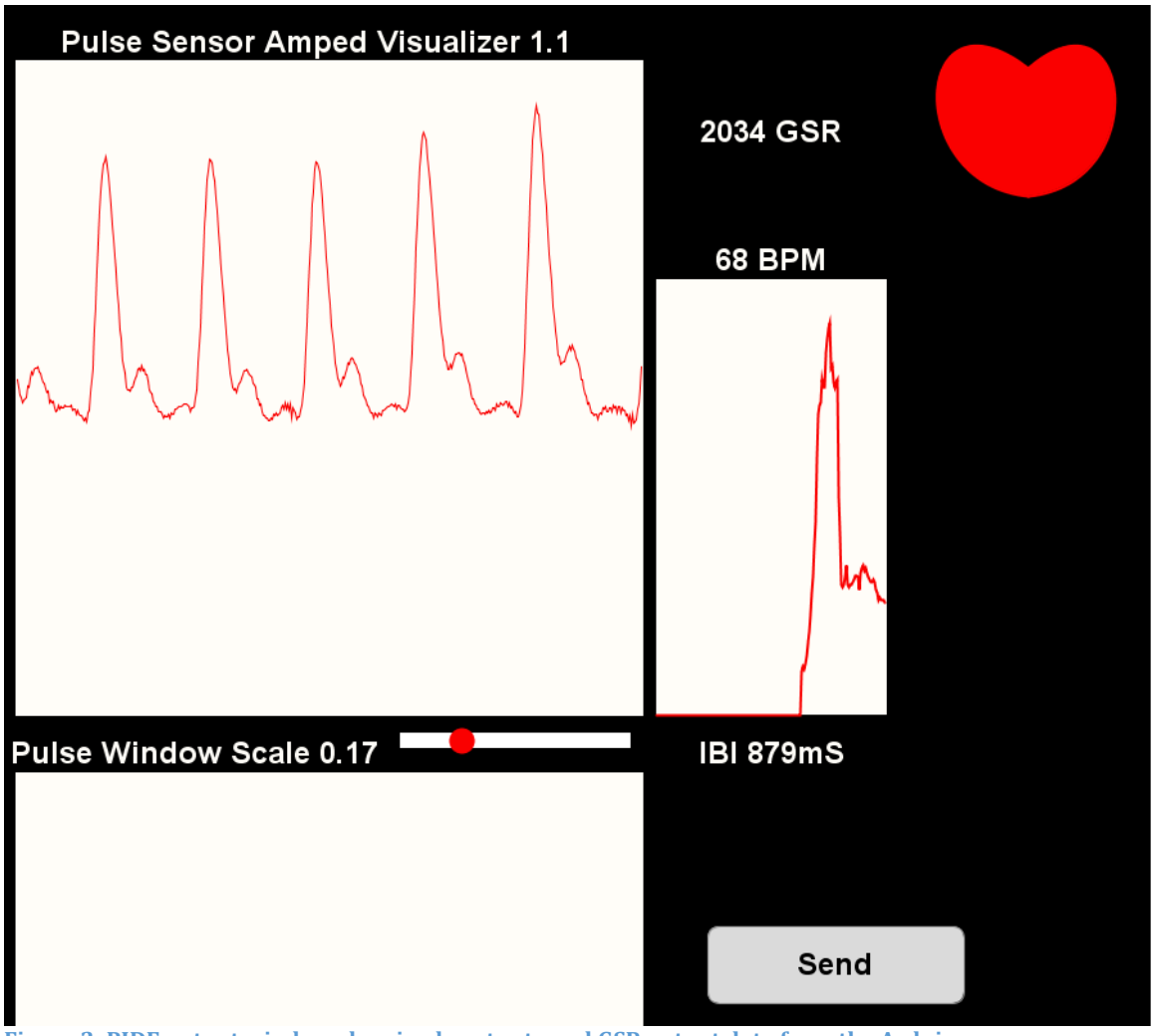

Figure 2: PIDE output window showing heart rate and GSR output data from the Arduino sensors

Now that the program was functioning in Eclipse, we were able to use the existing xAPI libraries to set up a client that would send xAPI statements to the LRS at any interval that we chose (in this case every five seconds). These values would then display side by side with what the learner was doing in the test case.

```
{
        verb : {
                id :  http://adlnet.gov/expapi/verbs/imported,
                display : {
                        en-US :  imported
                }
        },
        actor : {
                name :  medbiq,
```

```
            mbox :  mailto:info@openlabyrinth.ca,
            objectType :  Agent
        },
        object : {
            id :  http://demo.openlabyrinth.ca,
            definition : {
                interactionType :  choice,
                choices : [
                    {
                        id :  http://demo.openlabyrinth.ca,
                        description : {
                            GSR:3219 : BPM:153
                        }
                    }
                ]
            }
        },
        id :  a7eb9501-d0ae-4cf9-adbb-a912439f7727,
        stored :  2016-05-17T18:29:36.140Z,
        timestamp :  2016-05-17T18:29:36.140Z,
        authority : {
            account : {
                homePage :  http://openlabyrinth.ca/grassblade-lrs/xAPI/,
                name :  14-1e2c639ccc936c7
            },
            objectType :  Agent
        }
}
```


Figure 3: example xAPI statement generated by our Arduino device

Please note that Figure 3 is for illustrative purposes. It is likely there will be further changes to the verbs used by the device, and that the interactionType will be changed from 'choice' to 'device'. At present, the Profiles in this area are still evolving.

We had a few small problems with xAPI statement formatting but with help from members of the xAPI community, we were able to successfully link our GrassBlade LRS. In particular, we owe many thanks and wish to acknowledge the extensive help we received from Pankaj Agrawal at GrassBlade, Andrew Downes at Rustici, and Corey Wirun at Cardinal Creek.

The formatting of our xAPI statements from a device still needs to be refined. But the important aspect here was that the output was still useful and usable in its current temporary format.

Other potential xAPI sources

A number of other sources of activity tracking were considered, explored and partially incorporated into the workshop scenario design. For example, we initially explored the potential use of the Tobii range of eye-tracking sensors. Tobii makes a very sophisticated range of sensors for research purposes but most of these are quite expensive. One of the learning design parameters that we chose for this workshop was accessibility and affordability. We had hoped to use the simpler outputs from the Tobii EyeX Controller (www.tobii.com/xperience), which is much cheaper. However, the company was overwhelmed with demand for this level of sensor and was not able to assist us with this integration. This may be worth exploring in future.

We also noted that it is now quite possible to blend the actions and effects of a number of web-based applications, using software like Zapier (https://zapier.com) and If This Then That (IFTTT -- https://ifttt.com). We were able to create some interesting integrations between WordPress, Instagram, Evernote, Twitter and the smartphone camera itself, in ways that can further generate xAPI statements to be stored in our GrassBlade LRS. While integrations are very simple to implement, in the final set up of the workshop, we did not rely much on these integrations. They did provide some of the workshop participants with additional means of contributing and actively tracking, with the use of xAPI statements, the complex activity flows of this very dynamic session.

We were caught out by a discrepancy in the business model of Zapier. For demonstration purposes, we intended to set up the five free Zapier interactions that are advertised. However, we found that Zapier's billing structures required payment as soon as we set up the second interaction channel. We did not have time to explore this billing inconsistency and did not consider that the value added was worth it at this point, so we simply continued with the free IFTTT service.

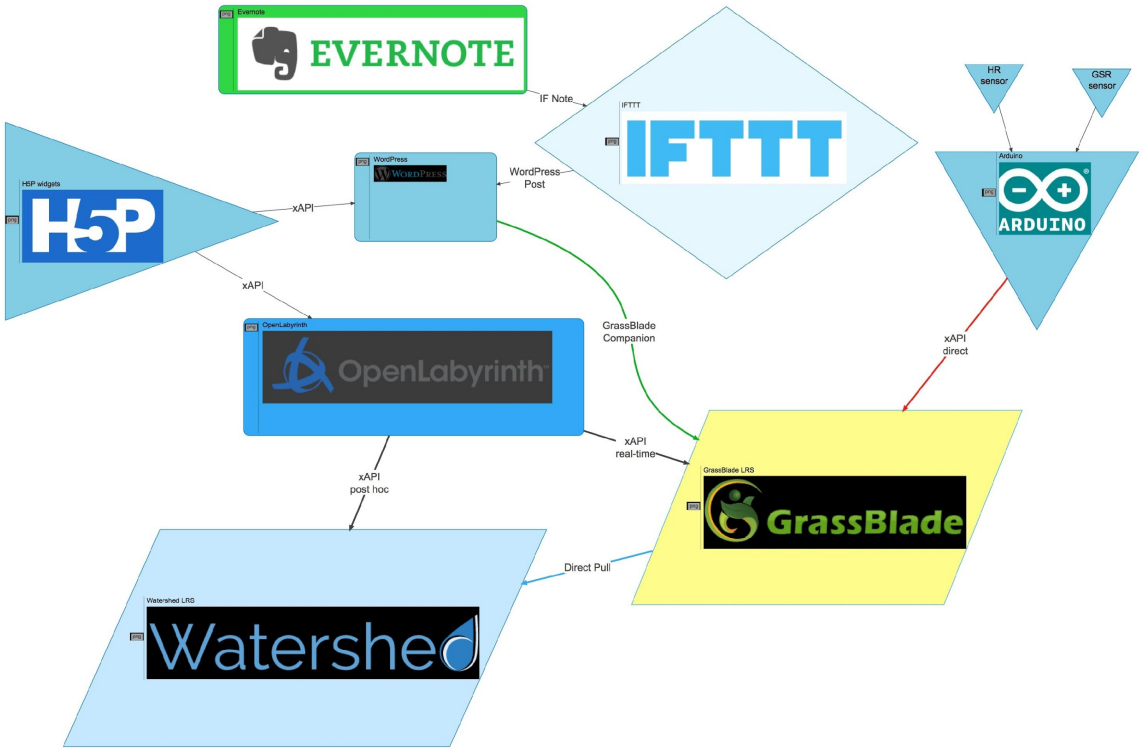

Figure 4: simple data flow diagram showing how xAPI statements were used between different applications

Deploying the Workshop

All of these multiple activities and data sources came together Just In Time. We are particularly indebted to CA for an extraordinary effort in bringing together the pieces needed for meaningful xAPI statements from our Arduino-based sensors. He was able to coordinate and collate multiple streams of advice from many in the xAPI community. Sometimes there is not a lot of sleep on the bleeding edge.

For the Medbiq workshop, we were able to demonstrate a live working example of tracking activity data from the following resources:

1. Heart rate, via Arduino sensor
2. GSR, from Arduino sensor
3. Mouse click timings from OpenLabyrinth
4. Decision tree responses from OpenLabyrinth
5. Question responses from OpenLabyrinth
6. Forced navigation jumps and timeouts from OpenLabyrinth
7. H5P widget input from OpenLabyrinth
8. H5P widget input from WordPress
9. Evernote notes via WordPress
10. IFTTT DO Camera images via WordPress

In the spirit of engaging as many workshop participants as possible, as well as tracking activity and stress sensors on our volunteers, we were also able to engage active

contributions and learning activities from parallel participants as they used the additional data sources to contribute to the overall data stream.

The workshop was successful in addressing a wide variety of participant needs. We had a mix of those who were primarily interested in the technical aspects of xAPI; those who were interested in exploring what could be done with such metrics; and those who were interested in exploring the enhanced variety of learning designs that could be supported by such a multi-modality approach.

There appeared to be a very high degree of engagement in all levels of the various activities, which sometimes made it difficult to keep everyone on track. When time was up, the room was slow to clear because of the continuing level of very active discussion. Note that this was one of the final sessions of the conference, when most people are starting to flag.

Analyzing the LRS Data

As soon as we started collecting xAPI statements in our GrassBlade LRS, we were able to perform some basic analysis using its built-in tools.

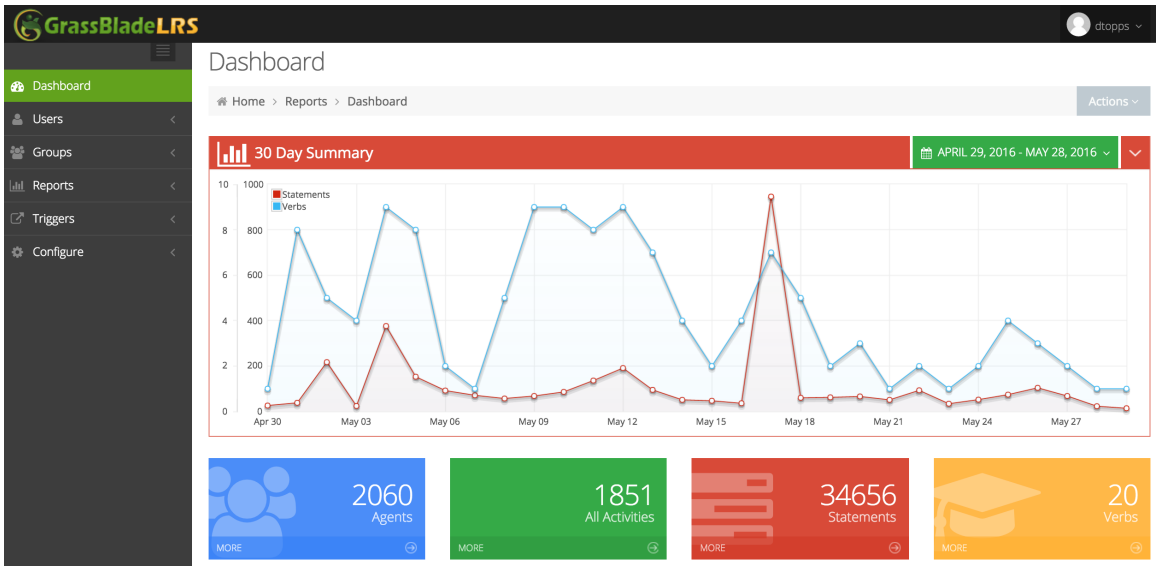

Figure 5: example dashboard from GrassBlade LRS

As we noted earlier, GrassBlade is very forgiving in its parsing of xAPI statements. This makes things much easier to set up, rather than just slogging through a long series of error statements, which we very much appreciated. We also greatly appreciated the accessibility of GrassBlade's creator, Pankaj Agrawal, who was very helpful in troubleshooting throughout all phases of the project.

We were also very fortunate to receive much help and advice from Rustici Software. They provided us with free access to sandbox accounts on both their SCORM Cloud (http://scorm.com/scorm-solved/scorm-cloud-features) and Watershed LRSs

(www.watershedlrs.com). SCORM Cloud has a number of testing functions which make it easier for developers to fine tune the syntax of their xAPI statements, including a manual xAPI Generator (https://tincanapi.com/statement-generator).

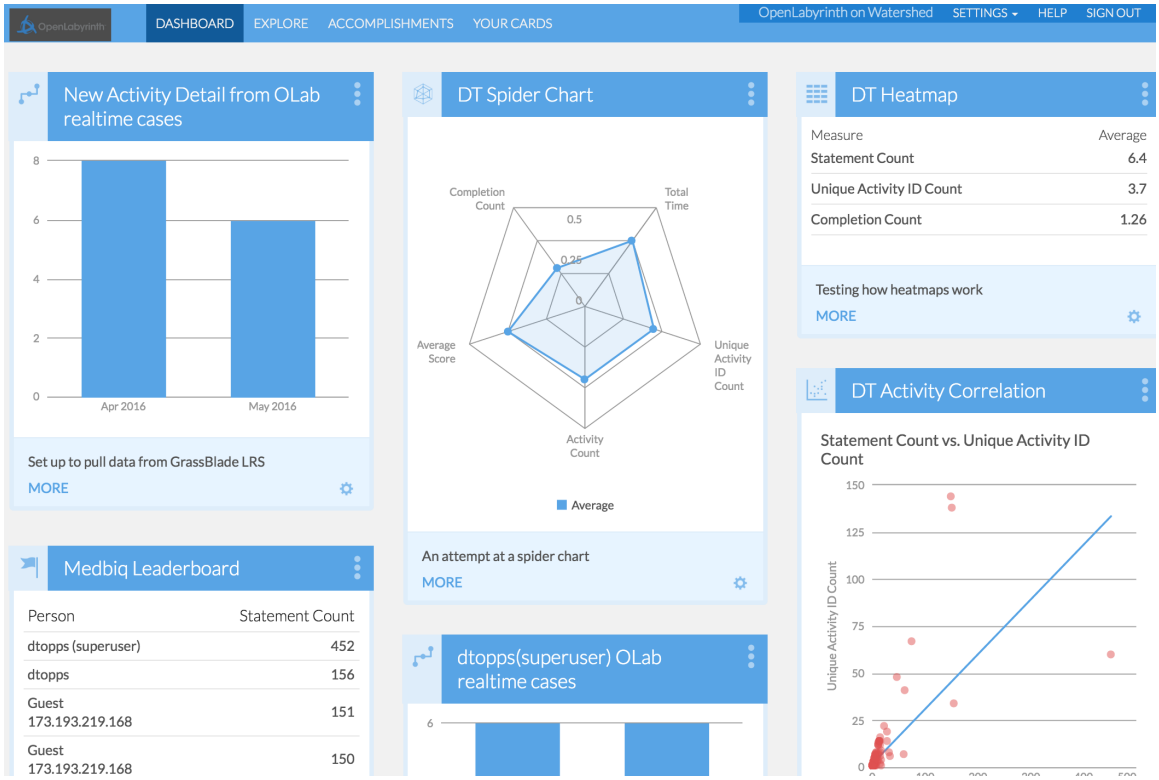

Figure 6: example dashboard from Watershed LRS

The Watershed LRS is a much more powerful animal, with serious horsepower and powerful visualizations. This was largely overkill for the simple aspects of this project and well beyond the budget available. However, Andrew Downes from Rustici, and Pankaj, were extremely helpful in fine tuning the interfaces in their respective LRSs so that we could explore direct data integration between Learning Record Stores. This gave us access to some data visualizations available in Watershed, by directly pulling xAPI statements from the GrassBlade LRS, and allowed us to explore the power of big league LRS, while still using the simple flexibility of GrassBlade for our initial data capture. We cannot applaud enough the collaboration evident in the xAPI community in surmounting the challenges that we found in taking this multi-layered approach.

We briefly explored the higher levels of assessment beyond activity metrics, based on xAPI statements, that might be afforded by Mozilla OpenBadges (http://openbadges.org). The Watershed LRS does provide some basic functionality that supports integration of Badges. However, on further discussion with the Rustici team, it was clear that few in medical education are interested in using Mozilla Badges at this point and therefore we redirected our efforts.

We were also fortunate to have access to a free trial account on the Saltbox Wax LRS (www.saltbox.com). We initially thought that it would just be interesting to simultaneously send xAPI statements to 3 different LRSs from OpenLabyrinth. This was surprisingly easy to set up and did not exact too much of a performance hit on our OpenLabyrinth server. It has since been pointed out that this is a somewhat redundant approach. It is in the nature of LRSs to be directly federated. So it would be more effective to use LRS triggers and filters to send selected statements from one LRS on to others, rather than to expect the source software to handle statement sending to all 3 LRSs.

The testing with the Wax LRS had an additional benefit: it is much more strict in its parsing of xAPI statements. After some initial grumbling on our part, we quickly realized the benefits of this: it enabled our developer teams to be much more exact in their xAPI statement parsing, which resulted in a much cleaner set of xAPI statements from OpenLabyrinth and our H5P widgets. We thank the Saltbox team for their assistance in this.

OpenLabyrinth itself has quite sophisticated internal tracking and reporting of activity metrics, along with some simple visualizations.

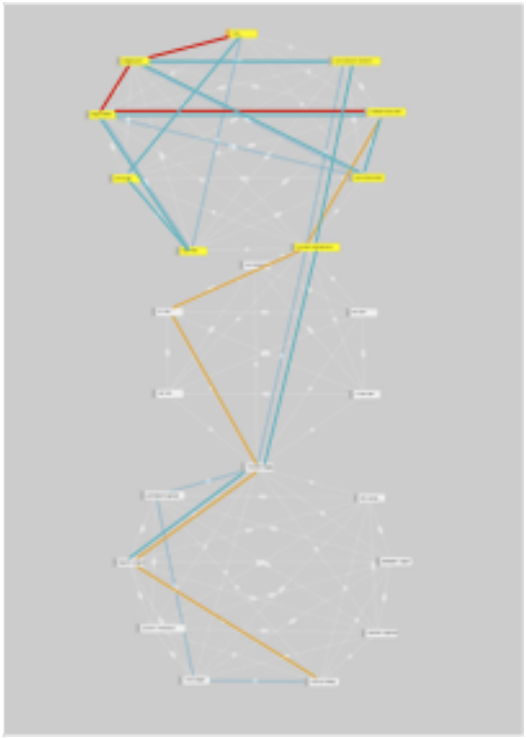

Figure 7: pathway analysis report generated internally by OpenLabyrinth

Indeed, the OpenLabyrinth developer team was initially perplexed as to why we were introducing the additional complexity of xAPI statement reporting since, at that stage, we were able to achieve the same level of activity reporting using its own functions. However, since we implemented xAPI reporting directly into OpenLabyrinth, several advantages have become clear.

Firstly, it has afforded simultaneous activity tracking across a much wider range of learning activities, not just OpenLabyrinth, as described in the paragraphs above. Secondly, it has relieved our developer team of the continuing burden of creating custom reports for various research groups. This is the bane of many databased research tools and can become a huge drain on resources. But thirdly, and most importantly, it has given us access to a much wider range of data visualization tools.

With the integration of xAPI, we have now moved into the Big Data world in learning analytics (Topps, Meiselman, Ellaway, & Downes, 2016). We state this, based not upon the Volumes of data we are generating (although these are rapidly increasing), but more upon the other Vs that are common in the principles of Big Data analytics: Velocity, Variety, Veracity and Visualization (Kobielus, 2013).

By placing our activity metrics in a common format, within an LRS that is designed to accept large volumes of data from multiple sources and federate them with other services, we can now take advantage of some of the more powerful visualization and analytic tools beyond the world of medical education.

It also allows us to explore both powerful commercial visualization tools like Tableau (www.tableau.com), which is designed for use with very large data sets, and also cheap but versatile visualization libraries like D3.js (https://d3js.org), an open-source, componentized visualization library.

Discussion

For this Technical Report, we have not adhered to the rigid IMRAD structure common to most academic articles, in the belief that highlighting some of the design issues that arose during the design-based research elements of the project was more consistent with our discovery oriented approach. Hence, several of the discussion points have already been covered earlier. We would however like to further highlight some important points that arose from the project as a whole.

It was very clear that the lowered cost of integrating different systems using a single API that tracks events and metadata is a big advantage of using xAPI. We found that xAPI makes it possible to adapt low cost, open source tools for sensors and learning interactions, and we can now track data that may illuminate the learning process far beyond what was possible with SCORM. What we did in measuring stress parameters in learners, while engaged in an educational exercise was not particularly new (Hardy, Mullen, & Jones, 1996)(Arora et al., 2010). But previous efforts have required a sophisticated simulation laboratory with expensive suites of sensor equipment, which are largely beyond the budget of most education researchers. Other attempts have used subjective measures of perceived stress (Cohen, Kamarck, & Mermelstein, 1983) which, while still relevant, have their own biases and lack of sensitivity. Our approach, using simple sensors and cheap computing devices, along with a flexible approach for gathering activity metrics from multiple sources represents an interesting advance in this area.

Conclusions

Data from multiple sources greatly extends our ability to track what learners actually do, not what they say they do or what their teachers say they do, thus potentially reducing many sources of bias.

Loose aggregation of activities via xAPI is more practical than the tight coupling previously envisioned by SCORM as a translation mechanism. By loosely coupling the data flows, we expect that there will be greater flexibility in learning designs and less dependency on keeping structural changes within communicating systems such as LMSs, LRSs, virtual patients, all aligned with each other.

Cheap computing devices like Arduino improve researcher access to sensor data through greater access to a broad developer community with an open-source attitude to design recipes; reduced cost of hardware and software development; and an ever expanding range of sensor modalities.

Bibliography

Abbey, D. (2016). Learning Locker on Github. Retrieved June 18, 2016, from https://github.com/LearningLocker/learninglocker/releases

Arora, S., Sevdalis, N., Nestel, D., Woloshynowych, M., Darzi, A., & Kneebone, R. (2010). The impact of stress on surgical performance: a systematic review of the literature. *Surgery*, *147*(3), 318–30, 330.e1–6. http://doi.org/10.1016/j.surg.2009.10.007

Cohen, S., Kamarck, T., & Mermelstein, R. (1983). A Global Measure of Perceived Stress. *Journal of Health and Social Behavior*, *24*(4), 385. http://doi.org/10.2307/2136404

Hardy, L., Mullen, R., & Jones, G. (1996). Knowledge and conscious control of motor actions under stress. *British Journal of Psychology*, *87*(4), 621–636. http://doi.org/10.1111/j.2044-8295.1996.tb02612.x

Kaplan, M. (2014). When to Use MongoDB Rather than MySQL (or Other RDBMS). Retrieved June 18, 2016, from https://dzone.com/articles/when-use-mongodb-rather-mysql

Kobielus, J. (2013). The Four V's of Big Data. Retrieved June 18, 2016, from http://www.ibmbigdatahub.com/infographic/four-vs-big-data

Korneliusz. (2014). MongoDB vs MySQL: Comparison Between RDBMS and Document Oriented Database. Retrieved June 18, 2016, from https://blog.udemy.com/mongodb-vs-mysql/

Lalley, J., & Miller, R. (2007). THE LEARNING PYRAMID: DOES IT POINT TEACHERS IN THE RIGHT DIRECTION? *Education*, *128*(1), 64–79.

Mei, S. (2013). Why You Should Never Use MongoDB. Retrieved June 18, 2016, from http://www.sarahmei.com/blog/2013/11/11/why-you-should-never-use-mongodb

Meiselman, E., Topps, D., & Albersworth, C. (2016). Medbiq xAPI workshop. In V. Strothers & P. Greene (Eds.), *Medbiq Annual Conference*. Baltimore, MD. Retrieved from

http://www.slideshare.net/topps/medbiq-xapi-workshop2b
Miller, G. E. (1990). The assessment of clinical skills/competence/performance. *Academic Medicine : Journal of the Association of American Medical Colleges*, *65*(9 Suppl), S63–7. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/2400509
Topps, D., Meiselman, E., Ellaway, R., & Downes, A. (2016). Aggregating Ambient _Student Tracking Data for Assessment. In *Ottawa Conference*. Perth, WA: AMEE. Retrieved from http://www.slideshare.net/topps/aggregating-ambient-student-tracking-data-for-assessment
Topps, D., Meiselman, E., & Strothers, V. (2016). *Medbiq xAPI Virtual Patient Profile*. Baltimore. Retrieved from http://groups.medbiq.org/medbiq/display/XIG/Profile%3A+1.+Virtual+Patients