The truth is that, despite initial expectations about learning environments, its use has not changed learning models very much. It could be argued that in most cases, its implementation has only meant the extension of a physical space—the classroom—into some kind of virtual space—the platform—in which the same teaching and learning practices are developed, however.

Many different factors could be taken into consideration as possible causes of this fact, but to our understanding, there is an easily noticeable one: the imbalance between the users’ experience in this kind of environment and their richer and more integrating daily use of the Web. This is so, despite the fact that in this kind of tool the users’ perception, how they live it, is absolutely determinant for their progress.

It is obvious that the net is spreading worldwide, and people are growing to use it from a very early age. This is to say that most university students do not arrive at their institu-
Given the multiplicity of scenarios and actors, the idea of a dominating platform is not something we could believe in or aim for.

It is not viable a never ending growth in modules and components that repeat the already existing functions from other tools.

Given the multiplicity of scenarios and actors, the idea of a dominating platform is not something we could believe in or aim for. It is not viable a never ending growth in modules and components that repeat the already existing functions from other tools.

For instance, it is quite usual that students and even large numbers of university staff have email accounts in free servers and they still prefer to continue using them instead of their insititutional accounts. They also prefer to use their regular instant messaging tools, where they centralize most of their communication needs. Given their daily experience on the web, users wish to share their resources easily and without break in continuity, as much as they are used to do with their bookmarks, documents, images, ... This is why whatever is posted within the academic environment—especially if it is a virtual environment—should not be kept from the rest of the world. On the contrary, users do not want to have to face multiple authentication processes, various passwords, specific search engines for each application or media technology different from the ones that they regularly use.

To use a spatial metaphor, we could say that the experience on the web is not like an apartment with multiple rooms and many access entrances: to be able to enter a room it is necessary to leave the apartment and access through a different door. It should be like an open plan loft, with differentiated areas but which are easily accessible, one could freely circulate among the different spaces without losing sight of the previous space or the loft as a whole.

Under these circumstances, the use of the regular platforms is still another compulsory task that extends the division between the academic and the personal context on the web. This minimizes the possibilities of a truly user-centered learning method, one that integrates the user’s outside experience. In this sense, one can’t but wonder if it is a semantic contradiction or not to label some platforms as more or less constructivist—such a fashionable term in pedagogical literature—than others, particularly since the real practice that their architectures allow does not differ much form traditional on-site learning models within separated spaces.

2.- Design alternatives: application-centered vs. user-centered

Trying that users’ experiences are incorporated into the learning process to enrich it has ample pedagogical implications. It also has design and technological implications. The most commonly used alternative is what we could call application-centered design which paradigm could be defined as “providing a platform with every conceivable thing...and more”. This paradigm is intrinsically linked to the ambition of placing the environment in a leading position in the market, as the best solution to the integration problem. Consequently, learning environments become specific universes with their own access, authorization, resource management, communication (e-mail, chat, etc,...) regulations; which still brings about important drawbacks.

On the one hand, we might have encountered the difficulties derived from the implementation of some of these environments in an institutional context, such as the issues that appear when trying to integrate them in other applications or the issues related with
student management: for the first group of difficulties it is often necessary to rewrite the application to some extent (e.g. OSP in Sakai) or we might need to use some twisted device that results in a more-appearing-than-real integration (e.g. LAMS in Moodle); for the second group of difficulties there is no other solution than using amendments that keep the various databases in synchrony or multiplying the loading processes.

Thus, given the multiplicity of scenarios and actors, the idea of a dominating platform is not something we could believe in or aim for. It is not even viable, not in the medium term, because of the never ending growth in the modules and components that repeat the already existing functions from other tools.

On the other hand, the second option, that we could call user-centered design aims to make the platforms permeable, to make some progress towards their transformation into a blurred-limits-environment, just like the experience on the web. All that should be done without undermining security and other considerations. This implies wondering which architectures can be the most appropriate when placing users at the center of their multiple experiences, which, in short means dealing with identity-centered architectures and collaborative applications.

Interoperability is the key: Interoperability is the degree to which a provider and consumer can successfully interface having never met [1], or in simple terms, collaboration among applications, and the available technologies are well known: APIs, standards such as OSIDs-OKI, identity technologies...

Using existing worldware in learning is not new [2] e.g. suggesting the use of spreadsheets as assessment tools—but it is only recently that the possibility of integrating different origin tools in a personal environment has become feasible, thanks to the technologies afore mentioned.
In a previous article we presented the Agora Virtual platform as a ‘collaborative’ tool. Collaborative in this case has a double meaning: It is an environment for the collaboration among users but it is also a platform that collaborates with other applications and services to minimize the need to ‘reinvent the wheel’.

The initial authentication implementation of the OSIDs and the use of the API in Google Maps for one of the modules (see figure 2) or an external Jabber server as instant messaging server (firstly jabberd2, and currently Openfire [4]) with a gateway to its similar services (see figure 3) were inserted into Agora Virtual. Other services keep their traditional design, such as the Rubric module which was developed to try out a training assessment following the regulations defined by the ESHE [5].

This is why, after two years of intensive use in different projects and training activities, and once a satisfactory standard was reached it was necessary to consider which way to go to make progress in the desired direction. Thus we identified two models which largely correspond with the aforementioned design alternatives; that is, we could focus on the tool to continue adding gadgets in a never ending race or, conversely, we could assess the current architecture to focus it on the user, re-orientating it towards two intrinsically linked directions: the collaboration among different applications and identity technologies.

To expand the interoperability of Agora Virtual® is planned to broaden the use of OSIDs to the rest of modules and functions.

The Harmoni services, besides the PHP implementations of most OSIDs, they offer an added functionality through the intermediate level services of APIs.
4.- Interoperability: OSIDs-OKI and the Harmoni framework

The OSIDs (Open Service Interface Definitions) from the OKI project are a list of guidelines that define how the different components of a software environment interact with one another and with other systems. Even from its initial version, Agora Virtual has been using its own implementation of the authentication OSIDs—and the other required OSIDs, such as Shared—The next step to expand its interoperability will be to broaden the use of OSIDs to the rest of modules and functions.

The new architecture is focused on the services and this is why it is based on the Harmoni framework [7] developed by the Curricular Technologies group at Middlebury College to provide an infrastructure based in OSIDs that supports the development and maintenance of the applications which are typical of educational environments. The Harmoni framework is made up of two parts: Harmoni architecture and its services, that include the implementation of OSIDs-OKI (see figure 5).

Both parts could be used together or separately since the services and the OSIDs implementations are designed to function independently from the application structure so they could be integrated into other architectures, as in the case of Agora Virtual.

Besides the PHP implementations of most OSIDs, the Harmoni services offer an added functionality through the intermediate level services of APIs, such as Database Manager (for building and running secure SQL queries), Sets (for maintaining sets of IDs), Tagging (for social bookmarking and folksonomies of any item with a unique ID) Image Processor (for image resizing and thumbnail generation), GUI y DataManager (for the repository management).

It is of particular interest the use of the module from Language Service manager which, being based on GNU Gettext as our first implementation, has allowed us to translate the platform to minority languages such as Bare and Ye’kwana within the framework of an international cooperation project funded by the AECID (Spanish Agency for International Cooperation). [8]

All services can be included, configured and used when necessary, since, just as the OSID, their implementation is independent from the implementation of the other services, and in most cases they are not linked to any specific environment (for example, they do not use environment variables typical of PHP such as $_REQUEST).
This new architecture (see figure 6) offers several advantages: the application modules can opt to access the Harmoni services APIs or to access the OSID directly, as appropriate; modules and services can be modified and they can be implemented again without affecting the rest since are supported by interfaces and not by implementations. Finally, the uncoupling of authentication and authorization OSIDs from the rest of the application allows for an optimal integration of the platform with the external middleware services. In fact, authentication is one of the least successful points in the second version of the OSIDs and it is being checked for the third version—along with Agents—this is why uncoupling it and connecting it to a service which is unconnected to the implementation will facilitate the adjustment of predictable changes.

5.- Identity: OSIDs + phpPoA + SimpleSAMLphp

Once we have decided on the decoupling of functionality modules and authentication and authorization devices, we still need to choose the implementation model that is to be applied.

The current architecture of Agora Virtual defines a sole authentication and authorization point at the ‘front controller’ of the application which facilitates the integration with external devices. Firstly we considered the possibility of using an OpenID server to act as an identity provider and against which the implementation of the authentication OSID would act as a consumer. This solution, however, was ruled out for two reasons: the current low security level of the OpenID protocol [9] and particularly because a simplified and very versatile PAPI PHP version was available [10]. It had been tested and it provided excellent integration opportunities with other identity tools such as SimpleSAMLphp.

SimpleSAMLphp is a light PHP library based on the Sun OpenSSO Extensions code (also known as Lightbulb) which allows to easily integrate a service in this language with an identity management structure that uses SAML 2. The most common solution to introduce an SP SAML 2.0 is to use an interface that has to be in the same language of the application so that it can communicate with the SP (see figure 7).

With these elements (OSIDs, phpPoA and SimpleSAMLphp) the final integration is simple: the authentication OSID acts as an interface between the application and the SP SAML 2.0. Version 2 of this OSID defines the following methods:

- authenticateUser(Type AuthenticationType)
- destroyAuthentication()
- destroyAuthenticationForType(Type AuthenticationType)
- getAuthenticationTypes()
- getUserId(Type AuthenticationType)
- isUserAuthenticated(Type AuthenticationType)
However, among the many advantages of the OSID-OKI architecture, the most important is its versatility: It is not necessary to develop all the methods defined by the relevant OSID since it will depend on the installation that is going to be carried out. In our case, the `isUserAuthenticated()` method is basically made up of a call to the PAPI access point in PHP, a phpPoA instance, which verifies the user’s authentication through its `check_Access()` method in its non-auto mode.

Because of its design, phpPoA requires a GPoA that carries out the real communication task with the IdP (SimpleSAMLphp in this case). This is why a modified GPoA, which we will call SimpleSAMLGPoA, acts as a hybrid component that functions as a SP SimpleSAMLphp regarding the IdP from which it receives the user’s attributes. The modified GPoA has also their own functions towards the phpPoA (preparing the ciphered assertion with the received attributes as the established configuration requires).

In any case, the IdP SimpleSAMLphp offers the possibility to use different plugins to check the user’s identity—LDAP and Radius by the time this text was written—which provides it with a great versatility. The authentication architecture is then as shown in figure 8.

The SimpleSAMLGPoA is used as a connector between a PAPI environment and a SAML 2.0 one. Other applications and services can be plugged to this connector and they will thus share the same authentication process and method.

Communication between these modules to carry out the initial authentication process is described in a simplified manner in figure 9.

1. The user tries to access a restricted section of the application and the request is detected by the authentication OSID which calls the PHPoA to check the user’s authentication.
2. PHPoA redirects the SimpleSAMLGPoA requesting a PAPI assertion.
3. SimpleSAMLGPoA redirects the IdP SAML 2 to authenticate the user.
4. The IdP provides a login form to the user.
5. The user fills in the login.
6. The IDp checks the identification and sends the attributes back to the SimpleSAMLGPoA.
7. SimpleSAMLGPoA constructs a valid assertion and sends it back to the user with redirection of the requested resource.
8. The user accesses the new resource.
9. The phpPoA sends the user a cookie.
10. The phpPoA allows access to the requested resource.

The whole authentication infrastructure is hidden from the application by the OSID interface.

The user could benefit from the advantages of having an only authentication point that is shared with other applications that are compatible with Web SSO architectures and federation devices.
Although it may seem complex, the whole process is transparent to the user who only receives the login form—managed by the IdP and thus decoupled from the application—and then the user receives the requested resource. Subsequent requests would be authenticated by the cookie, as it is usually done in PAPI, and secondly by the SimpleSAMLphp session.

The whole authentication infrastructure is hidden from the application by the OSID interface. For this type of authentication implementing an authenticateUser() method is not necessary because the login process is delegated to the IdP and authentication itself is delegated to the phpPoA that would send the user the appropriate cookie. Thus the whole process is external to the application which only knows about it through the result from the call to isUserAuthenticated(). For the IdP, however, the application-phpPoA-GPoA set is the only SP with which it communicates using SAML 2.0.

This model offers several possible points that could be integrated into an already-existing infrastructure thus providing different alternatives for its use:

In general, a specific authentication OSID could be always developed.

Counting on a PAPI infrastructure, using the implementation of this OSID and communicating the phpPoA with the already-present GPoA could be an option. Thus, it would be integrated with the rest of the infrastructure.

A different option could be installing an alternative GPoA to carry out the authentication against some other device, even HTTP Basic.

An implementation like this could be fully used against the IdP SAML 2; Thus one would only have to choose the most appropriate plugin, LDAP or some other, or one could develop their own, for instance for SQL.

Finally, if the service to be integrated uses SAML 2.0 instead of PAPI, the IdP SAML could be provided with a PAPI plugin to perform the opposite process: that is, integrating the SAML 2 service into a PAPI infrastructure.

Subsequently, the user could benefit from the advantages of having an only authentication point that is shared with other applications that are compatible with Web SSO architectures and federation devices: PAPI, Shibboleth, SAML 2.0 etc. which would serve as the first necessary step towards the implementation of user-centered services.

Environments which are far richer and less restrictive than the current ones could thus be constructed through pluggable components, reusing much of the already-existing software, enriching the user’s experience without reinventing the wheel, i.e. rewriting the same applications once and again to integrate them into each new platform.
6.- An example of integration

DokuWiki [12] is a well-known tool and one of the most simple and easiest wikis to use. This does not mean that it is less powerful though, since it consists of a set of plugins that enable DokuWiki to be used for almost anything. It is mainly orientated to prepare all kinds of documents but it is equally useful to elaborate teaching materials, particularly for groups (see figure 10).

Using the afore mentioned components it is easy to integrate the wiki into Agora Virtual since both of them, the environment and the wiki, share the same identity management device. (See figure 11).

The required architecture for this integration is simple: the DokuWiki application has been slightly modified so that it deploys the PoA PAPI for access control. At the same time GpoA acts as the only SP as opposed to the IdP of SimpleSAMLphp, as you see in figure 12.

This procedure is slightly different and, to our understanding, it has some advantages over the one that was originally used for the adaptation of DokuWiki to SimpleSAMLphp: firstly, only GpoA has to be configured as an SP SAML 2, whereas the adaptation of applications is quite more simple with the PAPI PoA. Secondly, the role of aggregating different applications done by the GpoA fits well with the model of integrating different tools in an only environment, providing a solution to problems such as the global signoff.

As it has been mentioned above, this is not the only possible solution. The other way round could also be possible: integrating an SP SAML 2 into a PAPI, as shown in figure 13.
7.- Conclusions

Learner-centered learning should be oriented towards a holistic vision of the subject’s experience if it is to be truly innovative. In this context the user’s everyday network experience is growingly relevant. Thus, state-of-the-art applications, and much more so learning environments, must interoperate within a new framework in which the real platform is the Web, whose center is the user.

Middleware technology advances are breaking the barriers among the users and also between institutions. The goal is that they can all collaborate with one another with the least possible inconveniences. It is not enough having collaboration applications—i.e. groupware applications—they also have to be collaborative with one another to be able to display their processes and share their resources starting with the user’s identity.

Adapting the development of learning environments to this new context means acting in a double front: on the one hand, identity managements and its derivatives (SSO, federation); and on the other hand, the interoperability standards of applications such as OSID-OKI; finally, as basic supports for the development of user-centered environments. These identity and interoperability technologies allow us to integrate great part of the existing worldware to move from the current concept of virtual learning environments to a really personal space for work, learning or collaboration.

Acknowledgements

We would like to thank Diego R. López (RedIRIS), Adam Franco (Middlebury College) and Andreas Ækre Solberg (Uninett) for their valuable help in developing the current text.

References


---

**APPROACHES**


---

**José Alfonso Accino**  
( accino@uma.es)  
SCI - Central IT Services  
Universidad de Málaga

**Manuel Cebrián de la Serna**  
( mcebrian@uma.es)  
Faculty of Education Sciences  
Universidad de Málaga