Niels Pinkwart, Johannes Konert (Hrsg.): Die 17. Fachtagung Bildungstechnologien,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 187

A mobile campus application as a sensor node for Personal Learning Environments

Hybrid cross-platform User Contextualization

Hendrik Geßner[1] (0000-0002-7786-2587) and Alexander Kiy[1] (0000-0002-5885-9657)

Abstract: For many web-based applications, there is at least one corresponding mobile application. By leveraging a mutual exchange of the cross-device user context between mobile and web-based application, guidance and processes can be improved and therefore simplified. This article presents an infrastructure to enable cross-device and cross-service personalization and adaption while aiming at high interoperability between heterogeneous systems. As a proof-of-concept, an existing mobile campus app framework was extended by a hybrid context framework to capture user data, which is stored in a Learning Record Store (LRS) by the use of the Experience API.

Keywords: mobile application, personal learning environment, xAPI, informal learning, context sensitivity

1 Motivation

While a few years ago there were only a few device classes that played a role in student life, there has been a rapid increase in device diversity. With full wireless coverage at universities coming into reach, the expansion of further mobile services to support teaching and learning is still at the beginning. To meet the individual challenges, a variety of services and mobile applications are used. As part of the university, formal and informal teaching and learning contexts are becoming increasingly blurred so that there can be no comprehensive use of information and data. The different contextual user data offers the opportunity to adapt the acquired institutional teaching and learning environment to the individual needs of the users in the best possible way. In the area of HCI, a large number of model- and rule-based variants for adapting mobile user interfaces already exist [FL15]. So-called context frameworks are increasingly used, which make it possible to configure both the mobile sensors used and the rules applied. However, these solutions are usually limited to the mobile device. On the other hand, the cross-device and multi-service capture and use of context information requires an underlying infrastructure. Hu et al. proposed a context aware architecture for mobile crowd sensing using a mobile SOA framework for mobile devices and a dedicated server component [HLN+14]. The approach makes it possible to capture context data across different mobile devices and to adapt the application to the device. However, the data management is decentralized so that

[1] University of Potsdam, Department of Computer Science, August-Bebel-Str. 89, 14482 Potsdam, {givenname.surname}@uni-potsdam.de

doi:10.18420/delfi2019_356

contextual data cannot be reused for other applications. This article introduces an infrastructure and a proof of concept that enables the use mobile context information for the adaptation of a web-based application. The hybrid mobile application Mobile.UP is extended by a context framework and the mobile context information is used for adaptation in the existing web-based personal learning environment Campus.UP [KLL17]. Thus it is conceivable that the last work status can be seamlessly resumed after a device change (mobile application to corresponding web application) and sensor information can also be used to enrich other services and mobile applications.

2 Conceptualization of the sensor node for PLEs

In order to find suitable use cases for the prototype implementation, we examined work from three different adjacent areas, collected ideas as user stories and enriched these with required context data and portlets before selecting two for prototype implementation. Economides [Eco09] provides an extensive list and classification of context sources in the field of pervasive and ubiquitous learning and Scheffel et al. [STD16] present a xAPI vocabulary. Mobile.UP [KLL17] contains features suitable for context extension. These sources resulted in two user stories[2]. User story 1, *current course*, states that if the user is currently attending a course his or her PLE should highlight the courses workspace. User story 2, *profile completion*, states that when completing the personal profile, the user should be supported by his or her PLE through suggestions[3].

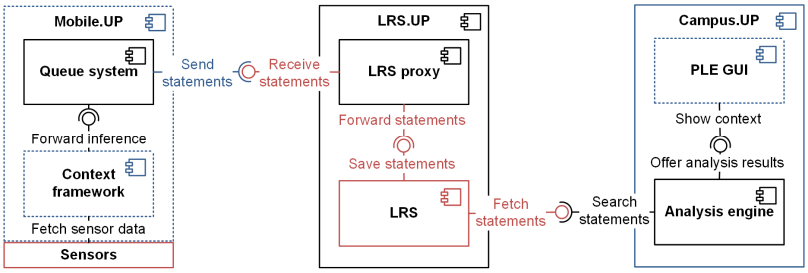

Fig. 1: System architecture with standardized components (red) and internal developments (blue).

The architecture consists of an extension of the mobile application Mobile.UP with a context framework for the acquisition of sensor information and the subsequent processing. Then the context information is transformed into xAPI statements. To handle changeable network connections, the application can be operated largely without a network, as a queue system sends the information to a central storage location, the so-called Learning Record Store (LRS), via the xAPI-specified interface [ADL16] as soon as possible. In

[2] A complete list of use cases can be accessed at https://dx.doi.org/10.5281/zenodo.3263934
[3] However, the evaluation of user story 2 showed that the implemented profile suggestions made no difference in efficiency, effectivity or user experience while only a small portion of suggestions were adopted at all. This led to the decision to omit any further details on the second use case in this article.

order to draw conclusions from the data stored in the LRS, an analysis component is used which contains knowledge about the structure and context of the data. As part of the previously selected use case *current course*, the display in the PLE is extended by additional information which is retrieved by the analysis engine via REST and then presented in the PLE. Taking into account the existing components, the architecture presented in figure 1 was designed (dashed lines symbolize the extension of an existing framework or system).

3 Implementation

The embedded *sensor and context framework* transforms the low-level context of the sensors into high-level context, which is then transformed into xAPI statements. The embedding of the framework allows access to smartphone and user-specific data and enables decentralized, network-independent processing. Several existing context frameworks were considered [Bar05, FKD15, MFN+13, MLL15, SDA99, Win17]. To ease integration into the Mobile.UP app, a mobile hybrid framework with offline capability and Cordova compatibility was required. This only left contactJS [MLL15], a JavaScript port of the Context Toolkit framework.

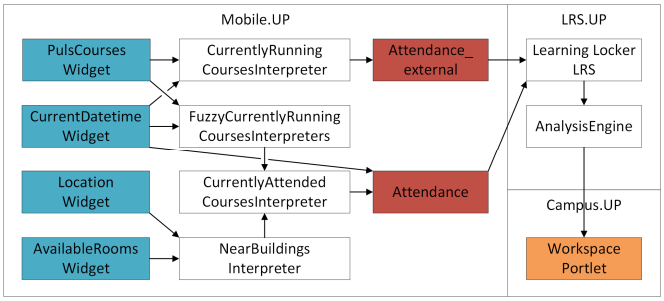

Fig. 2: Component-wise information flow from context framework to workspace highlighting

ContactJS employs so-called widgets to capture the low-level context, interpreters for further processing potentially low-level context to high-level context and rules to trigger conditional actions. In total, eight widgets, five interpreters and four rule sets were implemented of which a selection is presented in figure 2. The widgets retrieve building data as GeoJSON objects, current time in ISO 8601, device position following the HTML5 Geolocation API and enrolled user courses in a custom format, which is returned by the campus management system's web service. By matching high-level context with stored rules, xAPI statements are generated and sent to the *LRS*. The Experience API (xAPI) [ADL16] is a standard for capturing learning activities. For reusable scenarios, recipes exist which define the permitted vocabulary [BSD+16]. The communication between

mobile app and LRS follows the *ATTENDANCE* recipe[4][5]. To select an appropriate LRS implementation, the xAPI adopter list[6] was used. Mandatory criteria were free availability and the possibility of a local stand-alone installation. Only Learning Locker and IxHive fulfilled the requirements with Learning Locker providing an advanced query interface. As a result, Learning Locker was chosen. Communication between *analysis engine* and LRS was initially done through the xAPI query interface and the provided xAPI Java library. However, in addition to the intended statements, the LRS response also contained invalidated statements, their invalidation, and referenced statements. This behavior is correct with regards to xAPI, but unfavorable. Therefore, it was decided to use Learning Locker's aggregation API, which proved to be much more flexible and reliable. The analysis engine supports a requests for current user courses. Currently attended courses are determined by querying the LRS for all xAPI statements that match the detailed attendance recipe grouped by event ID and replaying the actions. The existing *PLE* Campus.UP is based on the open source portal software Liferay. The adjustment of features can be done via Hooks or by creating self-contained Portlets. Two existing Portlets, the workspace overview and the portlet for user profiles, were extended as part of this project. Highlighting the current courses workspace is performed by leveraging the already existing workspace overview portlet and extending it to regularly poll the *current course* REST interface and highlight designated workspaces in red.

4 Evaluation

The evaluation of the *current course* use case maintains a technical view by focusing on the time it took until the workspace linked to the attended course was highlighted in the PLE. An analytical evaluation was done, which yielded best and worst case intervals. Afterwards, these bounds were used to interpret the recorded durations collected from several experimental runs. Two starting points were covered: The user already arrived at the venue 10 minutes before the event start (starting point 1) OR the user is on his or her way to the venue at the 10-minute-mark (starting point 2). The starting point affects the expected system reaction times and delays. Figure 2 highlights the time-critical components. *CurrentDatetimeWidget* refreshes every 30 seconds, *Attendance* rules are rechecked in a 20 second interval and the Workspace portlet queries for new data every 30s. The *LocationWidget* refreshes every 60s but depends on the underlying locations plugin, whose refresh rate is not predictable due to mechanisms like geo fencing. The data processing and transfer happens in a millisecond range and will therefore be neglected in the analysis. Maximum runtime is reached when each component has to wait a full refresh cycle before processing prior results. In starting point 1, this results in a remaining theoretical worst-case runtime of 30s + 20s + 30s = 80s. The experimental runs yielded runtimes of 22s, 40s and 50s, the average case is 37.3s. All values are below the analytical

[4] https://registry.tincanapi.com/#profile/48
[5] A list of implemented xAPI statements can be accessed at https://dx.doi.org/10.5281/zenodo.3263934
[6] https://xapi.com/adopters/

maximum or worst case of 80s which strengthens the value of 80s as the worst case limit. In practice, these values denote that in an implementation which may recognize a course as active 10 minutes before it starts, the associated workspace is highlighted no later than 8.5 minutes before it starts. A maximum delay of 1.5 minutes can be seen as acceptable or good. The theoretical worst-case maximum runtime in starting point 2 includes the *LocationWidget* refresh interval. This results in max(30s, 60s) + 20s + 30s = 110s, but may be higher due to the underlying location plugin. In the experimental evaluation, the test device was taken to a location about 2.3 km away from the designated event venue well before the 10 minute mark. Upon re-arrival, the time until highlighting of the expected course was measured and the timestamp of the *joined* statement sent by the test device was noted. Two runs were done which yielded runtimes of 0s each and *joined* statement send timestamps of 62s and 102s before venue arrival. The average case is 0s with a statement sent 82s before arrival. In practice, this means that an implementation which updates its device position every 60s can highlight the associated workspace immediately on arrival at an event venue. This represents the optimal result for starting point 2.

5 Conclusion & Outlook

An architecture for cross-system data gathering, processing, transfer and usage of context information by using standardized and conventional technologies was presented. The user information gathered by the app were utilized to automatically adapt the PLE. For that purpose two representative use cases were selected and implemented prototypically. The designed architecture bases on standards like xAPI while remaining flexible enough to allow additional use cases. The prototypes evaluation showed that it is very suitable to highlight current courses in the PLE. The worst case and average case time limits result from refresh intervals within the app and the PLE. One can expect to reduce these limits by implementing higher refresh rates. Nevertheless, one should keep in mind that higher refresh rates come at the expense of increased power consumption and data traffic, which are precious resources on mobile devices systems such as smartphones and laptops. Cordova cannot guaranty JavaScript background execution, which is necessary for continuous context gathering and processing. Additionally, there is no prior experience on the power consumption of a hybrid Cordova app when compared to a native solution. In each case a native approach may be more suitable. As part of this prototype it has been shown that a number of auxiliary functions are required (cf. xAPI *opened* statement) that could be provided by other infrastructure (e.g. campus or learning management system) via active push of user context information into the LRS. The increased crosslinking of systems is now so far that fields like Learning Analytics rearrange themselves because services linked by standards like xAPI yield more detailed and more comprehensive insights than ever before. There is hardly any long-term experience and no broad consensus regarding cross-platform and cross-service vocabulary [BSD+16].[7]

[7] The research of Hendrik Geßner has been partially funded by Deutsche Forschungsgemeinschaft (DFG) through grant CRC 1294 “Data Assimilation”, Project Z03 “Information Infrastructure for data assimilation”.

Bibliography

[ADL16] Advanced Distributed Learning (ADL) Initiative, U.S. Department of Defense (2016) xAPI Specification. https://github.com/adlnet/xAPISpec

[Bar05] J. E. Bardram: The java context awareness framework (JCAF) - a service infrastructure and programming framework for context-aware applications. PERVASIVE'05 Proc. of the 3rd Int. Conf. on Pervasive Computing. Springer-Verlag, Berlin, Heidelberg, pp. 98-115, 2005. doi:10.1007/11428572_7.

[BSD+16] A. Berg; M. Scheffel; H. Drachsler; S. Ternier & M. Specht: The dutch xAPI experience. In: (D. Gašević, G. Lynch, S. Dawson et al., Eds.) LAK '16 conference proceedings: The 6th Int. Learning Analytics & Knowledge Conference, The University of Edinburgh. The Association for Computing Machinery, New York, pp. 544–545, 2016.

[Eco09] A. A. Economides: Adaptive context-aware pervasive and ubiquitous learning. IJTEL 1(3): pp. 169-192, 2009. doi:10.1504/IJTEL.2009.024865.

[FL15] J. Feng and Y. Liu: Intelligent context-aware and adaptive interface for mobile LBS. Journal. 2015.

[FKD15] D. Ferreira; V. Kostakos & A. K. Dey: AWARE: Mobile Context Instrumentation Framework. Frontiers in ICT. 2, 2015. doi: 10.3389/fict.2015.00006

[HLN+14] X. Hu; X. Li, E. Ngai; V. Leung & P. Kruchten: Multidimensional context-aware social network architecture for mobile crowdsensing. IEEE Communications Magazine. 46(6): pp. 78-87, 2014. doi:10.1109/MCOM.2014.6829948.

[KLL17] A. Kiy; C. List and U. Lucke: A virtual environment and Infrastructure to ensure future readiness of Data Centers. European Journal of Higher Education IT (EJHEIT), 2017.

[MFN+13] M. E. F. Maia; A. Fonteles; B. Neto; R. Gadelha; W. Viana & R. M. C. Andrade: LOCCAM - loosely coupled context acquisition middleware. In: Proc. SAC '13 Proc. of the 28th Annual ACM Symposium on Applied Computing. ACM New York, NY, USA, pp. 534-541, 2013. doi:10.1145/2480362.2480465.

[MLL15] T. Moebert; S. Lemcke & U. Lucke. ContactJS -- A Cross-Platform Context Detection Framework. In: IEEE 15th Int. Conf. on Advanced Learning Technologies, pp. 108-110, 2015. doi:10.1109/ICALT.2015.83.

[SDA99] D. Salber; A. K. Dey & G. D. Abowd. The context toolkit: aiding the development of context-enabled applications. CHI '99 Proc. of the SIGCHI Conf. on Human Factors in Computing Systems. pp. 434-441, 1999. doi:10.1145/302979.303126.

[STD16] M. Scheffel; S. Ternier & H. Drachsler: The Dutch xAPI Specification for Learning Activities (DSLA) – Registry, 2016. http://bit.ly/DutchXAPIreg

[Win17] J. Winterfeldt: Context-Aware Mobile Crowd Sensing using Mobile Hybrid Application Frameworks. Masters thesis. Universität Ulm, 2017. http://dbis.eprints.uni-ulm.de/1486/