Situation-Awareness in Action: An Intelligent Online Learning Platform (IOLP)

Jasser Jasser, Hua Ming(✉), and Mohamed A. Zohdy

Oakland University, Rochester, MI 48309, USA
{jjasser,ming,zohdyma}@oakland.edu

Abstract. *Situation* is a computational abstraction that encapsulates human-centric contexts, which can be human-oriented behavioral contexts as well as the relevant environmental contexts. Its potential applications in computer science range from artificial intelligence, service computing, human-computer interaction, pervasive computing to software engineering and software systems. In this paper, we introduce a user-centric *situation-aware* Intelligent Online Learning Platform (IOLP) that aims to provide a personalized learning experience for those conducting online-learning activities. It joins the engineering techniques from Machine Learning, Intelligent Systems (IS) and Human-Computing Interaction (HCI) to offer highly personalized services to the users. It serves as an experimental subject for, as well as a real-world implementation of *situations*. Under the strength of the abstraction of *situation*, various elements drawn from different areas in computer science are seamlessly integrated towards a user-centric Intelligent Online Learning Platform (IOLP). Our discussions on the IOLP are further supported by an online class instance it currently hosts featuring Python Programming.

Keywords: Situation · Situation-awareness · Human-computer interaction (HCI) · Massive Open Online Course (MOOC) · Intelligent Systems (IS) · Online learning · Adaptive learning

1 Introduction

Information sharing by the internet has vastly extended the philosophy of learning and therefore, the means to pursue learning has drastly changed. The pervasive accessibility of the internet, escalated by the explosive growth of mobile platforms and mobile apps prepared necessary conditions for the further development of online learning services. People no longer need to attend a class to learn a specific topic; the internet is full of tutorials, videos, and discussion forums to help them obtain knowledge to master a specific subject. MIT OpenCourseWare spearheaded the sharing of high-quality university-level courses [18]. This led to the rise of Massive Open Online Courses (MOOC), opening the way to popular online learning services including Coursera, edX, and Udacity [23]. However, the existing MOOC-based services mainly provide recorded videos, navigation menus,

M. Kurosu (Ed.): HCI 2017, Part II, LNCS 10272, pp. 319–330, 2017.
DOI: 10.1007/978-3-319-58077-7_25

and multiple-choice homework problems to facilitate independent student online learning activities. While having successfully migrated the learning experience from classrooms to the more open internet space, these existing services have been mostly recognized as one-size-fits-all [28] that provide less than sufficient capabilities to intimately interact with, guide and engage individual students. Indeed, there is an outstanding need for an infrastructure that is able to provide personalized online learning services based on individual clients, or client groups.

In this work, we present Intelligent Online Learning Platform (IOLP), an intelligent system-based online learning platform that aims to provide a fully personalized learning experience for each individual. IOLP and transitively all the services hosted on top of it are empowered by *situation-awareness* [7,8,21]. In addition, we present a concrete online class instance hosted by IOLP that features a personalized learning experience for Python Programming.

2 The Concept of Situation and Situation-Aware Computing

The concept of *situation* has its roots in mathematical logic [4,11,19] and analytical philosophy [5]. Over the years, it entered different areas in computer science including computational linguistics [12], logic programming [17], functional programming [20,21], artificial intelligence [19,25], human-computer interaction (HCI) [13,21], pervasive computing [26,27], service computing [8], software requirements engineering [2,3] and software evolution [8].

Of particular interest, [8] came up with a novel framework called *Situ*, empowered by the concept of situation, lends itself to defining situation as a rigorous computational abstraction towards automated service evolution. In [8], situation includes human-oriented cognitive and behavioral contexts, as well as the environmental contexts on a temporal basis. Indeed, *situation* thus defined is context-oriented [1,24] and in other words, a situation is a hub of context data lending itself to a wide spectrum of the-state-of-the-art machineries, such as machine learning techniques, during this age of big data and big data analytics.

Following the vein of work initiated by [8,21,22] further developed situation into a programmable abstraction that enables software reusability, as well as providing a straight-forward support for functional-programming originated, big data analytical tools such as MapReduce [10].
In particular, situation-awareness empowers IOLP with the following aspects:

1. Situation-centric design. The system scenarios are derived, combined and interchanged in the unified form of situations, which primarily consist of user's cognative, behavioral and environmental contexts. This design view ensures user-centric philosophy of the entire IOLP system, from macro level to micro level.
2. The underlying Intelligent Systems of IOLP are constructed based on situations, i.e., different user's are treated based on their collective context data that compose user-specific situations, which eventually leads to the generation of personalized learning experiences for individual users.

3. Dimensional situation analytics [22] was used to facilitate the information derivation and comprehension, while IOLP used situations to encapsulate and analyse user's raw context data.

3 The Intelligent Online Learning Platform (IOLP)

IOLP consists of two major Intelligent Systems (IS). The first is the Intelligent Assistant that provides real-time answers to student questions and tips for further exploration and study. It takes the student question as an input, compares it with a database full of related questions artificially trained using Bag-of-Words learning model, then provides an answer and a clue to what the student should ask about to understand the material in question. As a student progresses through the traditional videos, slides, homework, and exams, the history of his/her actions is tracked and recorded to generate individualized learning characteristics for each student. Our unsupervised machine learning techniques then cluster students based on his/her progress in the learning process among both current and previous students who took the same class. The second major IS is the Intelligent Advisor, which then directs students to enhance his/her performance in the areas where they are having difficulties, by recommending chapters for review, homework to retake, and topics to dedicate more focus. The IS will also generate exams containing questions related to areas where a student had trouble understanding and needed extra study time. We tested IOLP by designing a Python programming course, where our database is filled with videos, slides, questions, homework, and tests that are related to that material.

The main components of the IOLP are:

- The class view, which contains the lecture videos, presentation slides, homework and exam links, and extra resources to help the student.
- The Python Console, which is a full online Python Interpreter, where the student practices writing code.
- The Homework view, where the student is directed to write code for a specific homework using the Python Console.
- The *Student Help Portal*, where the student is directed if they are having difficulty solving a problem or want help regarding a specific topic.
- The Exam view, where the student is tested based on his/her knowledge of the current studied chapter.
- The Intelligent Assistant: The first major Intelligent System in IOLP where the students ask questions to get answers and hints.
- The Intelligent Advisor: The second major Intelligent System that assesses student performance to provide constructive feedback.

3.1 Intelligent Assistant

The Intelligent Assistant provides clear and efficient answers, and sometimes code examples about various topics regarding Python programming language. Figure 1 shows the Intelligent Assistant providing an answer to "How to write

hello world." The Intelligent Assistant also provides Speech Recognition and Text-to-Speech features, where it takes a user voice as input, and provides an answer for the student like any other smart assistant, such as Apple's Siri, Microsoft's Cortana, Google's Assistant, or Amazon's Alexa.

Fig. 1. The Intelligent Assistant

The question design is divided into three categories:

- **What** questions provide a core definition regarding the topic asked about, for example:
  - Q: What is the "Hello World" program?
  - A: Is a simple program that outputs the "Hello, World!" as-is to the standard output screen. It is usually the simplest working program, which can be written in most programming languages, to illustrate the basic syntax to write a computer program.
- **How** questions provide an explanation on how the topic is used and implemented with a real-life example, for example:
  - Q: How to write Hello World program?
  - A: The following is the Hello World program (with a code snippet on the side to show the code, as shown in Fig. 1)
- **Why** questions provide more explanation regarding a specific topic or a specific approach, for example:
  - Q: Why Hello World?
  - A: Hello World is considered the simplest form of a coding program and is used by many tutorials as the starting point to familiarize the student with writing code.

The hint module within the Intelligent Assistant suggests topics the students can explore deeper, in particular when a student asks a question that the system considers unconventional to his/her learning level based on the student's activity history. All questions in the database are connected in a k-ary tree structure, where within each level every node (question) contains 0 or k children (dependent questions) [6]. When a student asks a question, the IS will go through the tree path of the question history of the student. If it finds that the parent node of the current asked question is not registered in the student's question history, it will provide a hint to explore a topic that is related to the uncovered question node. Figure 2 shows an example. In the example, the student asked about Conditional

Statement. After the IS traced the student's question history, it discovered that the student didn't read, watch, or ask about Boolean Expressions, which is a prior topic to understanding Conditional Statements. Therefore, the IS provides a hint to the student to consider asking about this topic first.

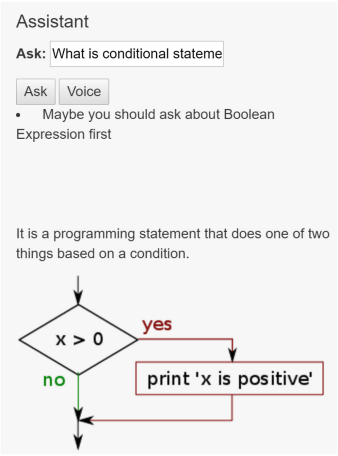

Fig. 2. The Intelligent Assistant hint module

The Bag-of-Words. Answers to student questions are retrieved from the database by measuring the similarity between the asked question and every question within the database. Together with a vector space model, the bag-of-words representation is used to define the similarity between the questions, where every question is presented as a multidimensional vector [9]. In this way, a vocabulary that contains all distinct terms after linguistic preprocessing is created. Each term, either in the student's question or database of questions, is weighted using w_{ij}, then both are expressed as t-dimensional vectors $d_j = (w_{1j}, w_{2j}, \ldots, w_{tj})$ where j represents the numbers of questions in the database, $j = 1, 2, 3, \ldots, n$. The database of all questions is represented by a term matrix (also known *term-frequency matrix*, as shown in Fig. 3).

To compute the weights, the frequency of a term i in question j is calculated using the formula (1), where the frequency is normalized by the frequency of the most common term in the document.

$$tf_{ij} = \frac{f_{ij}}{max_i(f_{ij})} \tag{1}$$

where f_{ij} represents the frequency of term i appearing in question j.

After calculating the term frequency, the inverse document frequency is used to specify the discriminative power of term i, as shown in formula (2).

$$idf_i = \log_2(\frac{n}{df_i}) \tag{2}$$

$$\begin{bmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{bmatrix}$$

Fig. 3. Term matrix for database of n questions and t tems

where df_i is the document frequency of term i.

Finally, the weights are computed by multiplying the term frequency by the inversed document frequency as shown in formula (3). The terms that occur most in the question are assigned the highest weights.

$$w_{ij} = tf_{ij} * idf_i \tag{3}$$

The question with the highest weight is considered the question that is most similar to the question the student asked, the index of which will be used thereafter by the Intelligent Assistant to search and retrieve an answer that is most closely relevant.

3.2 Student Activities History

A powerful feature in IOLP is the student activities history. A student history is collected from four types of activities: the student's asked questions, the student's code sessions done in the Python Console provided in the platform, the student's homework sessions (like the code sessions), and the student's test records. Having a record of all the activities helps the IS to build each student report after assessing his/her performance. The student activities history is the base of the Intelligent Advisor, since every action committed by a student is translated into learning features the Intelligent Advisor will use to perform the student clustering algorithm. The platform main view, shown in Fig. 4, always provides a panel to remind the student of his/her latest activities.

3.3 Intelligent Advisor

After the learning features are collected from the students activities, the Intelligent Advisor clusters the students that share similar feature values together, meaning these students are performing at the same level. The Intelligent Advisor will then provide every group with the appropriate level homework, exams, and tips. Each student will receive his/her own dynamically generated report that contains tips and reviews based on his/her individual history that will suit his/her learning experience. The Intelligent Advisor does not evaluate the student's performance until they progress through the learning material. The following features are used in assessing students performance:

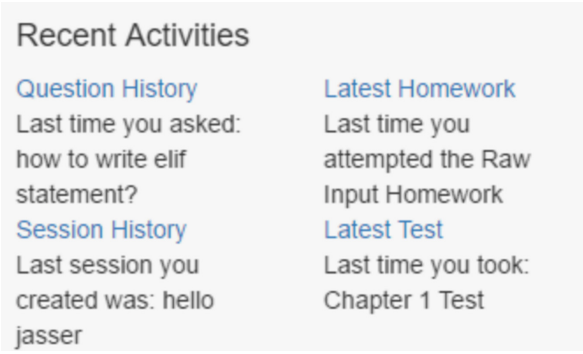

Fig. 4. The recent activities panel

1. **Attention to Details:** In this feature, the student's ability to deal with the Python Programming Language's syntax, i.e., the grammar, is measured.
2. **Breadth of Knowledge:** This feature measures the progress of the student's learning as they progress through the chapters.
3. **Problem Solving:** This feature measures the ability of the student to solve problems using the Python Programming Language.
4. **Abstraction:** This feature measures the ability of the student to learn advanced material and write complex code.
5. **Teamwork:** This measurement is for student participation on the student help portal.
6. **Creativity:** This measures a student's ability to provide new approaches to solving advanced problems.

These features are obtained as a student progresses through the online lectures, and by asking questions, writing code in the Python Console, doing homework, and passing exams. Each action the student takes is translated into points towards a specific feature. For example, asking *What is hello world program?* will gain the student a point towards the **breadth of knowledge** feature, since it is considered a general knowledge question, while asking *How to write hello world program?* will gain the student a point toward **attention to details**, since it deals with the obtaining the knowledge to write the syntax for this program. For the **creativity** feature, we follow a unique approach in determining the level of creativity in his/her work. For example, if the student is tasked to type the word "*hello*" 4 times (hellohellohellohello), it can be simply done by using the *print* statement, but a creative student would use the $*$ operation to multiply the string 4 times, since Python allows string operations such as multiplication:

- Normally: *print "hellohellohellohello"*
- With novelty: *print "hello"* $*$ *4*

Unsupervised learning is also known as clustering, or class discovery. It is one of the major categories in machine learning, the other two being the Supervised Learning, and Reinforcement Learning. What makes unsupervised machine learning different than the other two is that it deals with data that does not have

labels. In our case, the students are considered data with no label, because they are grouped based on similarity metrics; similar student activities lead to similar performance levels. We carefully studied and revised our list of features to ensure an optimum criterion. Our choice of algorithms for this problem are the centroid based algorithms, such as k-means and its varieties [14].

In our experiment, we used k-means clustering, also known as Lloyd's method [16], a versatile, and fast clustering method for large datasets. A set of points in a Euclidean space together with a positive integer k (i.e., the number of clusters) are provided as input to the algorithm, by which the data points are split into k clusters to minimize the total sum of the (squared Euclidean) distances between each point and its nearest cluster center.

Consider a set of data X $\subseteq \mathbb{R}$, and k clusters $C_1, C_2, \ldots, C_k$ with corresponding centroids $c_1, c_2, \ldots, c_k$. Then, for each element x $\in$ X assuming that the size of X is n, we need to find the value j that minimizes the Euclidean distance between x and c_j:

$$dist(x, c_j) = \sqrt{\sum_{i=1}^{n}(x_i - c_{ji})^2} \tag{4}$$

$$C_j = \{x : min_h dist^2(x, c_h) = j\} \tag{5}$$

To find the mean of all elements x that belongs to cluster C_j, we have:

$$c_j = \frac{1}{m} \sum_{x \in C_j} x \tag{6}$$

m_j is the number of elements in C_j.

Given formula (5), we minimize the summation of distance between each point in the cluster and the centroid in that cluster as follows:

$$\sum_{j=1}^{k} \sum_{x \in C_j} dist^2(x, c_j) \tag{7}$$

As an example, in Fig. 5 students are grouped into 3 clusters, (excellent students, good students, weak students), based on the student performance features.

In Fig. 5, the centroids are represented with grey dimension signs. *Excellent students* are represented by the blue dots. *Good students* are represented by the green multiplication signs. Finally, *weak students* are represented by the red asterisks. The Intelligent Advisor will provide all students in a cluster with the same homework and exams, but will also utilize **personalized** approaches to engage individual students to maximize their performances. For example, if a student demonstrates a lack of creativity, then the Intelligent Advisor will start providing them with more problems that they can solve with creative approaches.

3.4 Energy-Aware Situation in IOLP

We designed IOLP to be energy efficient whether working on PCs or mobile devices. Since the display is considered the most energy consuming component,

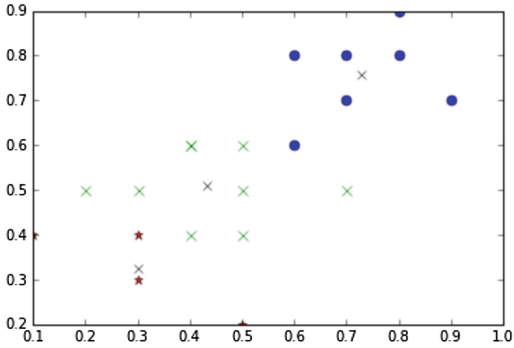

Fig. 5. Example: 20 students in 3 clusters by k-means

we implemented Li, Huyen, and Halford's approach [15] for automatically rewriting the Cascading Style Sheets (CSS) files that empower the webpages under IOLP. By this engery-aware feature, the light-colored background that consumes more energy is switched to more energy friendly colors, which are usually darker, without sacrificing the user experience. Figure 6 shows the usual user interface and the energy friendly user interface side by side.

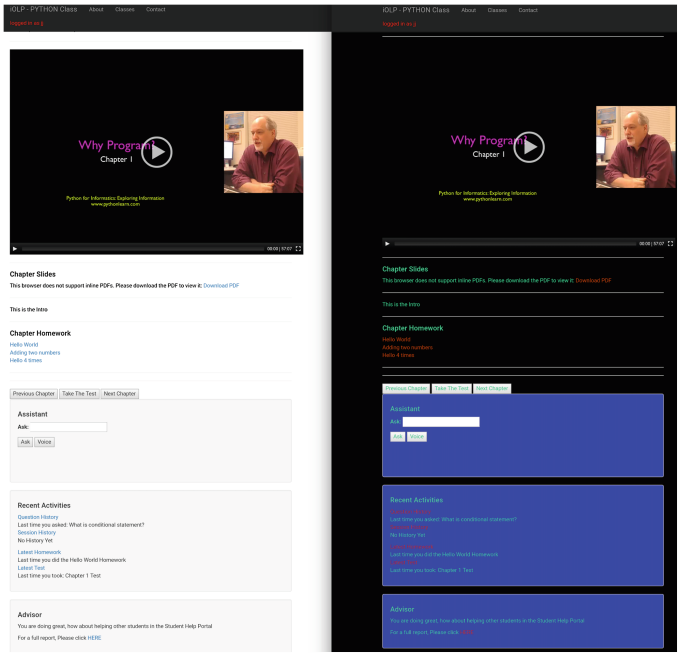

Fig. 6. Standard interface (left) vs. engergy-efficient interface (right)

4 IOLP vs. MOOC

What makes IOLP different from the other major MOOC platforms, such as Coursera, edX, and Udacity, is the intelligent components working in the background to provide students with the best learning experience. Our main goal in IOLP is to make sure that every student gets the right amount of feedback based on his/her learning activities. Other platforms generally provide only a recorded lecture, navigation menu, and multiple-choice question homework. Students navigate the material freely without making sure they understand the previous chapter before jumping to the next one. While it provides students with freedom from the traditional classroom, it also risks losing students who need more direction. IOLP has two vital intelligent systems to follow up with student progress and always direct them to the right track, an advantage that is missing in other platforms. There, the students need to wait for a feedback from the assistants monitoring the class, a response to his/her questions on the questions form, or not getting an answer at all due the massive amount of students participating in the class. Our platform also uses a student help portal so students can ask tough questions, such as code compiling errors, and code semantic errors, that are considered too advanced to our current state-of-art artificial systems to answer.

5 Conclustion and Future Work

Our future work is two-fold:

1. To facilitate students from all over the globe to use our platform to get as many question wording biases as possible, so that our platform can be fully trained to answer any question that is input from a student whose first language is not English. Also, we seek to have a dataset of student features that would help the Intelligent Advisor to accurately cluster the next batch of students based on his/her performance. We also plan to implement a massive online testing of the platform to verify its accuracy and efficiency.
2. To escalate the situation-awareness level the IOLP possesses. As the system grows more sophisticated, owning more features, this objective becomes especially challenging.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999). doi:10.1007/3-540-48157-5_29
2. Alkhanifer, A., Ludi, S.: Towards a situation awareness design to improve visually impaired orientation in unfamiliar buildings: requirements elicitation study. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE), pp. 23–32. IEEE (2014)

3. Atukorala, N.L., Chang, C.K., Oyama, K.: Situation-oriented requirements elicitation. In: 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 233–238. IEEE (2016)
4. Barwise, J.: The Situation in Logic. Center for the Study of Langauge and Information. Stanford University, Stanford (1989)
5. Barwise, J., Perry, J.: Situations and Attitudes. MIT Press, New York (1983)
6. Black, P.E.: Dictionary of Algorithms and Data Structures. National Institute of Standards and Technology Gaithersburg, Gaithersburg (2004)
7. Chang, C.K.: Situation analytics: a foundation for a new software engineering paradigm. Computer **49**(1), 24–33 (2016)
8. Chang, C.K., Jiang, H., Ming, H., Oyama, K.: Situ: a situation-theoretic approach to context-aware service evolution. IEEE T. Serv. Comput. **2**(3), 261–275 (2009)
9. Cios, K.J., Pedrycz, W., Swiniarski, R.W.: Data mining and knowledge discovery. In: Cios, K.J., Pedrycz, W., Swiniarski, R.W. (eds.) Data Mining Methods for Knowledge Discovery, pp. 1–26. Springer, New York (1998)
10. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. In: OSDI, p. 1 (2004)
11. Devlin, K.: Logic and Information. Cambridge University Press, Cambridge (1995)
12. Devlin, K.: Situation theory and situation semantics. Handb. Hist. Logic **7**, 601–664 (2006)
13. Endsley, M.R.: Toward a theory of situation awareness in dynamic systems. Hum. Factors **37**(1), 32–64 (1995)
14. Gentleman, R., Carey, V.: Unsupervised machine learning. In: Gentleman, R., Carey, V.J. (eds.) Bioconductor Case Studies, pp. 137–157. Springer, Heidelberg (2008)
15. Li, D., Tran, A.H., Halfond, W.G.: Making web applications more energy efficient for OLED smartphones. In: Proceedings of the 36th International Conference on Software Engineering, pp. 527–538. ACM (2014)
16. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
17. Loke, S.W.: Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. Knowl. Eng. Rev. **19**(3), 213–233 (2004)
18. Martin, F.G.: Will massive open online courses change how we teach? Commun. ACM **55**(8), 26–28 (2012)
19. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. In: Readings in Artificial Intelligence, pp. 431–450 (1969)
20. Ming, H.: Situ^{f}: a domain specific language and a first step towards the realization of situ framework. Ph.D. dissertation. ProQuest Dissertations & Theses Global. UMI 3539397 (2012)
21. Ming, H., Chang, C.K.: Can situations help with reusability of software? In: Kurosu, M. (ed.) HCI 2016. LNCS, vol. 9731, pp. 598–609. Springer, Cham (2016). doi:10.1007/978-3-319-39510-4_55
22. Ming, H., Chang, C.K., Yang, J.: Dimensional situation analytics: from data to wisdom. In: 2015 IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), vol. 1, pp. 50–59. IEEE (2015)
23. Pappano, L.: The year of the MOOC. New York Times **2**(12), 2012 (2012)
24. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Context aware computing for the internet of things: a survey. IEEE Commun. Surv. Tutorials **16**(1), 414–454 (2014)

25. Reiter, R.: The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In: Artificial Intelligence and Mathematical Theory of Computation, vol. 27, pp. 359–380 (1991). Papers in honor of John McCarthy
26. Yau, S.S., Liu, J.: Hierarchical situation modeling and reasoning for pervasive computing. In: The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA 2006), p. 6. IEEE (2006)
27. Ye, J., Dobson, S., McKeever, S.: Situation identification techniques in pervasive computing: a review. Pervasive Mobile Comput. **8**(1), 36–66 (2012). http://www.sciencedirect.com/science/article/pii/S1574119211000253
28. Yousef, A.M.F., Chatti, M.A., Schroeder, U., Harald Jakobs, M.W.: A review of the state-of-the-art. In: Proceedings of CSEDU 2014, 6th International Conference on Computer Supported Education, pp. 9–20 (2014)