Enabling Programmability of Smart Learning Environments by Teachers

Asterios Leonidis[1], Margherita Antona[1(✉)], and Constantine Stephanidis[1,2]

[1] Institute of Computer Science, Foundation for Research and Technology – Hellas (FORTH), N. Plastira 100, Vassilika Vouton, 700 13 Heraklion, Crete, Greece
{leonidis,antona,cs}@ics.forth.gr
[2] Department of Computer Science, University of Crete, Crete, Greece

Abstract. The evolution of Information Technology (IT) and the emergence of the Ambient Intelligence paradigm have drastically affected the way users live and learn. Ambient Intelligence is a vision of the future that offers great opportunities to enrich everyday activities (e.g., on the road, at home, at work, etc.) and has been proven to play an important role in education. In smart learning environments, learning activities are enhanced with the use of pervasive and mobile computing. This paper presents an extensible software infrastructure that empowers teachers to design and program purposeful and engaging learning activities for formal and informal learning environments, by combining and orchestrating cloud-based, ambient and pervasive facilities and services.

Keywords: Visual programming · End-user development · Ubiquitous environments · Smart learning environments

1 Introduction

Ambient Intelligence is a vision of the future that offers great opportunities to enrich everyday activities (e.g., on the road, at home, at work, etc.) through the pervasive presence of a variety of objects – such as RFID tags, sensors, actuators, technologically enhanced artifacts, etc. – which are able to interact with each other and cooperate with their neighbors to reach common goals [17]. Such novel paradigm, also known as Internet of Things (IoT), is rapidly gaining ground [2] and aims to revolutionize the way people interact with computers, as smart environments will anticipate and react to human needs even without users' explicit commands [41].

In the meantime, people have also changed the way they learn due to the rapid pace of life and the strong dependence on technology for their daily activities. Transferring knowledge only through traditional classroom activities is considered obsolete, and new learning methodologies, which make use of technology, have emerged to improve the learning process by allowing learning in different locations. This sort of learning occurs anytime and anyplace, when and where the learner desires. Smart learning environments, rooted in intelligent tutoring and adaptive systems [10], context-aware ubiquitous learning [22], and mind tools [12], can be regarded as technology-supported learning environments that make adaptations and provide appropriate support (e.g., guidance, feedback, hints or tools) in the right places and at the right time, based on individual learners' needs, which might be determined via analyzing their learning behaviors, performance and the online and real-world contexts in which they are situated [23].

A key aspect for such environments to achieve their full potential is the ability to be open and extendable [16]. From an engineering perspective, these concepts outline the need for appropriate middleware frameworks and communication technologies that facilitate the introduction of new devices, services and software components [3]. On the other hand, from a user perspective, they rather emphasize the scarcity of programming methodologies and tools that could facilitate the construction of intelligent systems using existing technologies [45].

Operators of learning environments are not experienced programmers. They are teachers with limited, if not any, experience in computer programming, whereas the programmability of the environment is more complex than creating a rule that turns on the room's light when someone enters [29, 46]. This paper aims to demonstrate a prototype system that empowers teachers to create learning scenarios by reviewing and modifying the high-level "business logic" of a smart learning environment in a user-friendly manner through a visual programming platform.

2 Related Work

Within smart learning environments, various learning activities take place that make extensive use of ICT technology. Contrary to the past, where the e-learning paradigm dictated that digital technology was mainly used to gain access to learning content from a stationary device (e.g., portable or desktop computer) and interact in a sandboxed environment with it through specialized applications (e.g., e-learning portals), nowadays with the emergence of the Ambient Intelligence paradigm learners are able, and often required, to interact with multiple devices (either purely digital or augmented with technology) in order to accomplish predefined learning objectives, whereas in many cases learners are required to participate in kinesthetic learning activities, i.e., physically engaging classroom exercises such as moving to a certain place to accomplish a task [4].

For instance, [20] requires learners to use their mobile devices along with specialized equipment to analyze several poor quality power supply occurrences and then share and discuss their findings with their classmates; the latter requires sharing content with other devices in real time. In [14] the authors propose various learning activities that require collaboration among multiple applications and multiple users in real-time. Chang [9] taught recycling principles and [13] studied the effects of mobile blogging, both applied in the wild. Finally, [47] have implemented a system aiming to increase interactivity in the classroom by using mobile technologies. The emergence of this novel paradigm is also supported by the fact that various EU-funded projects aim at developing blended educational spaces where physical and digital artifacts are combined in learning activities, while digital and learning frameworks are developed to modernize and improve education by motivating learners' participation.

From the point of view of the learning experience, [43] identifies a set of requirements for smart learning environments which comprises effectiveness, efficiency, scalability, autonomy, engagement, flexibility, adaptiveness, personalization, conversation and reflection. Many of the above requirements however are closely related to the emerging paradigm of end-user development which dictates that at some point the end-user should be able to modify a software artifact [21, 31]. To that end, many alternatives have been proposed based on the visual programming paradigm, that has been proven to facilitate inexperienced users to quickly learn how to build simple programs [19]. Gray and Young [18] describes Virtuoso, a multi-user programming environment built using the Valve's Source engine that functions as a tool to allow non-professional users to create interactive educational video games. Maloney et al. [32] presents Scratch, a visual programming environment that allows users (primarily ages 8 to 16) to learn computer programming while working on personally meaningful projects such as animated stories and games. Chin et al. [11] reports a programming environment for customizing smart home environments where the user demonstrates the desired behavior and the system encodes it as a set of rules to be executed in real-time. Kubitza [28] argues that building environments with heterogeneous interconnected devices still remains a challenging task and proposes a toolkit to cover this technical complexity, so that designers and users of a smart environment can focus on the interaction design and the programming of intelligent and useful behavior. The majority of those systems employ a custom scripting language, embracing the concept, stemming from the gaming engineering community, that "smarter, more powerful scripting languages will improve game performance while making gameplay development more efficient" [48].

3 Framework Requirements to Support SLEs

The emerging trend of mobile and ubiquitous computing has attracted numerous researchers and vendors to build educational applications that benefit from innovative technological affordances. Nevertheless, the majority of those mobile and ubiquitous applications often do not meet their potentials due to the lack of tools that simplify their interplay with the environment [5] and the contained affordances. Therefore, to enhance the programmability of smart learning environments, a set of requirements were elaborated during the implementation of the overall framework.

Currently, the majority of educational applications targeting ambient and ubiquitous environments offer limited functionality as they operate within their own sandbox [1, 6, 36, 37, 49]. However, as intelligent environments blend into our daily activities and life-long learning becomes a necessity [42], the demand for federated educational services and applications is constantly increasing, and the need for appropriate facilities and tools becomes imperative. Thus, the proposed framework enables the development of complex learning scenarios in which educational applications cooperate with existing services/applications and benefit from ambient facilities (e.g., sensors, artifacts, etc.).

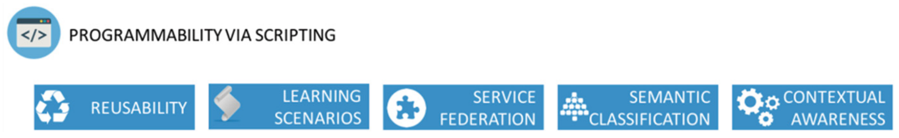


Fig. 1. High-level requirements for the AmIClass SDK

Composition and module reusability are desirable characteristics, which in the domain of software engineering partially determine the product quality. For that to be achieved, the communication channels through which applications can exchange information and the rules that mediate their interactions when forming complex federations are formally specified. Consequently, the proposed framework supports: (a) seamless integration of new learning applications and services, (b) reuse of existing facilities through composition & cooperation (based on semantic classification, compatibility checking, etc.), and (c) different degrees of openness [39] that will allow the dynamic discovery and use of semantically equivalent services (e.g., based on availability, QoS, preference, etc.) (Fig. 1).

Within the available learning infrastructures [5, 7, 34], teachers and learners cannot configure, let alone define, learning scenarios and activities. However, in the broader domains of Ambient Intelligent and agent-based computing, various solutions have been proposed [38] that facilitate the orchestration and dynamic adaptation of "execution scripts" that govern the entire process. Based on such well-known practices, the proposed framework provides: (i) a mechanism that facilitates the semantic classification of learning-oriented intelligent artifacts and services, and (ii) a "scripting" library that will support the orchestration and customization of the various learning facilities within ambient and ubiquitous learning environments. The library supports the definition of an appropriate context-sensitive decision-making logic, which can be dynamically modified either explicitly by the teacher or implicitly through activity monitoring (e.g., modification of the learning context, availability of available services and artifacts, recognition of undesirable situations, etc.).

The concept of Ambient Intelligence is built around the notion of multiple objects, embedded in the environment, being capable of recognizing and responding to the presence of different individuals; those entities (i.e., people, objects) and their current state of interaction are defined as contextual knowledge [15]. In order to design and apply suitable learning strategies in the context of ambient environments, the exploitation of contextual information is crucial. The proposed framework offers access to contextual knowledge to support context-aware decision- making. Within smart learning environments, the context of use includes [5, 15, 42]: (i) learner-related attributes (e.g., schedule, performance, skills, etc.), (ii) intelligent objects and their facilities, (iii) learning applications and services, and (iv) learning activities and their requirements. Contextual knowledge is used for appropriately adapting the learning process within various environments (e.g., benefit from technologically rich environments with multiple affordances, minimize interaction while on the move or when available time is limited, etc.).

4 Implementation Details

4.1 Overview

To satisfy the aforementioned requirements, the proposed work roots its core principles in the game scripting paradigm that has been successfully applied within the last decades and proposes a user-friendly scripting environment through which teachers can monitor and modify the “high-level” business logic of the learning environments they are in charge of through a visual programming language.

The AmIClass SDK offers a programming and a runtime environment that facilitates the definition, deployment, execution and monitoring of various learning activities that make use of ambient facilities within smart learning environments. Figure 2 provides an overview of the proposed software infrastructure.

AmIClass SDK aims to enable the design of purposeful and engaging learning activities for formal and informal learning environments, by combining and orchestrating cloud-based, ambient and pervasive facilities and services (Fig. 2). To that end, the following high-level components were implemented:

- **A Service Mediator Agent (SMA)** that integrates and provides access to ambient and pervasive services. It can resolve services offered by: (a) smart objects such as sensors, technologically augmented artifacts, interaction devices, etc., that expose their functionality in the form of software services, (b) Knowledge hubs that collect and provide personalized access to learning material from various content providers or learning management systems, (c) Context-sensitive observers that facilitate environmental monitoring and controllers that enable remote management, (d) Software-as-a-Service [44] Learning Applications that deliver their functionality over the network, (e) Profiling agents that simplify logging, enable personalization and promote social interaction, and (f) Security safeguards that implement access management policies.

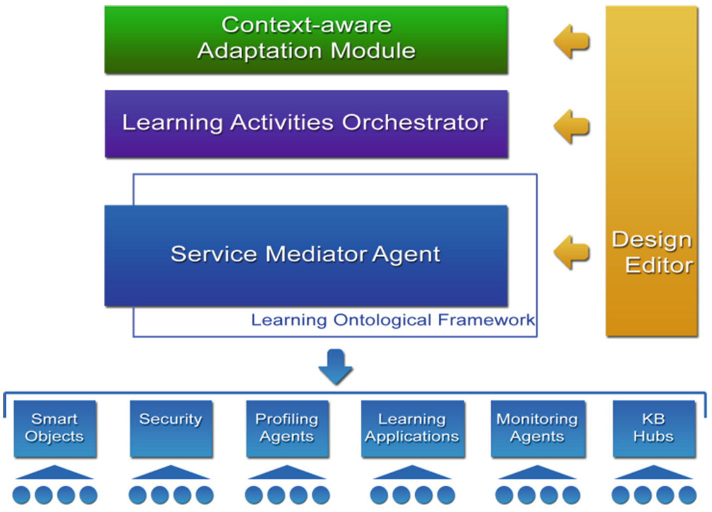


Fig. 2. Overview of the AmIClass SDK

- **A distributed Learning Activities Orchestrator (LAO)** that, through the SMA services, monitor and orchestrate the learning process by controlling the physical environment, as well as the learning services and applications. LAO facilitates: (i) dynamic service binding and use, (ii) conflict resolution, and (iii) adaptation of the activity workflow. Additionally, it supports the installation of external composite modules that extend existing functionality and offer new features. Finally, it supports the implementation of meta-services that can be integrated in the SMA to enable the realization of composite learning scenarios.
- **A Context-Aware Adaptation Module (CAAM)** that facilitates the decision-making process by evaluating rule sets that, based on the available contextual knowledge, determine which scenario alternatives should be applied or how active scenarios should be reconfigured.
- **An extensible Learning-Oriented Ontological Framework (LOF)** that: (a) Facilitates the classification of the environment and its smart objects, devices and services, (b) Allows semantic mapping of data and services extending existing knowledge, (c) Enables high-level semantic reasoning (e.g., is device X appropriate for displaying private content, etc., (d) Delivers a unified profiling model, (e) Simplifies services federation and reuse (including services dynamically added at run-time), and (f) Enables communication of heterogeneous systems.
- **A Design Environment (DE)** for defining and configuring learning activities' scripts.

4.2 Teacher-Friendly Features of the ClassScript Language

The ClassScript language aims to empower the definition of various learning scenarios in smart learning environments. Based on the Service Mediator Agent and the various distributed Orchestrators, any connected services are dynamically identified and using appropriate programming facilities (e.g., reflection) are exposed, via dynamic code generation, as external ClassScript modules that offer various functions. To support both professional and non-professional programmers, two different visibility levels are available, namely full access and teacher-friendly access, that aim to hide unnecessary complexity from novice end-users (Fig. 3).

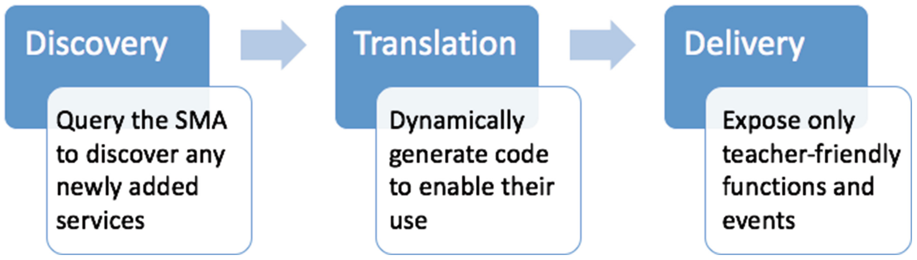
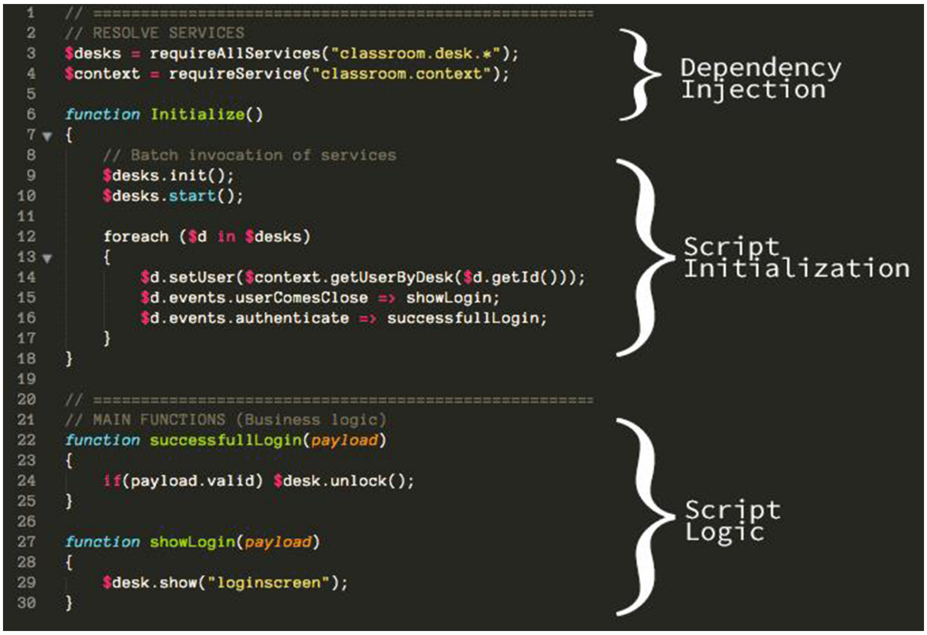

Discovery
Query the SMA to discover any newly added services
Translation
Dynamically generate code to enable their use
Delivery
Expose only teacher-friendly functions and events

Fig. 3. Inspection process to export teacher-friendly functions and events



```
// ======================================================
// RESOLVE SERVICES
$desks = requireAllServices("classroom.desk.*");
$context = requireService("classroom.context");

function Initialize()
{
    // Batch invocation of services
    $desks.init();
    $desks.start();

    foreach ($d in $desks)
    {
        $d.setUser($context.getUserByDesk($d.getId()));
        $d.events.userComesClose => showLogin;
        $d.events.authenticate => successfullLogin;
    }
}

// ======================================================
// MAIN FUNCTIONS (Business logic)
function successfullLogin(payload)
{
    if(payload.valid) $desk.unlock();
}

function showLogin(payload)
{
    $desk.show("loginscreen");
}
```

Fig. 4. A sample ClassScript program

A valid program in ClassScript has a well-defined structure (depicted in the Fig. 4) which not only streamlines the parsing procedure, but most importantly facilitates its manipulation from the visual editor by the teachers acting as a master template that guides them. The structure is influenced by that of a program written in the C programming language [25] and requires import statements to precede function declaration that in turn precede the script's initialization function. In more details:

- **an import statement** defines the service dependencies and is equivalent to an include statement in C (e.g., find and resolve the necessary external services before executing a single line of code).
- **function declarations follow a C-like scope system.** Function are not hoisted, thus they can only be used if they have been declared beforehand [25]
- **a script's initialization function (i.e., the main function) defines the initial entry point** that prepares the necessary local structures and performs the registration of event handling functions for local and remote events. Finally, each function (including the **init** function) can contain all the usual programming constructs such as arithmetic expression, variable definition, function calls, etc.

4.3 Web-Based Management Suite

Teachers can control the smart environment through a web-based Management Suite by visually exploring the available programmable artifacts alongside with their scripts, while they can adapt their business logic either by modifying existing or by introducing new scripts. The teacher can browse through both stationary (e.g., Smart Desk [40], Educational Mini-games station [26], the Book of Elli [33]) and mobile artifacts that can be found in the environment. However, in such fluid environments, it is not only the physical manifestation that matters, but also the overall functionality offered. Therefore, teachers can browse through either physical instances (i.e., environment monitoring and management) or through conceptual "service containers" that could be instantiated anywhere at anytime (i.e., business logic manipulation).

For every artifact type, the teachers can explore: (i) the compatible services and (ii) the associated scripts, while for every artifact instance they can further examine the status of any deployed services as well (e.g., active, busy, idle, stopped). Similarly, for every service, teachers can explore the exposed functions and the events that could be triggered, alongside with the scripts that either consume any of the functions (i.e., direct use through dependency injection) or listen for any of the events to react accordingly [8].

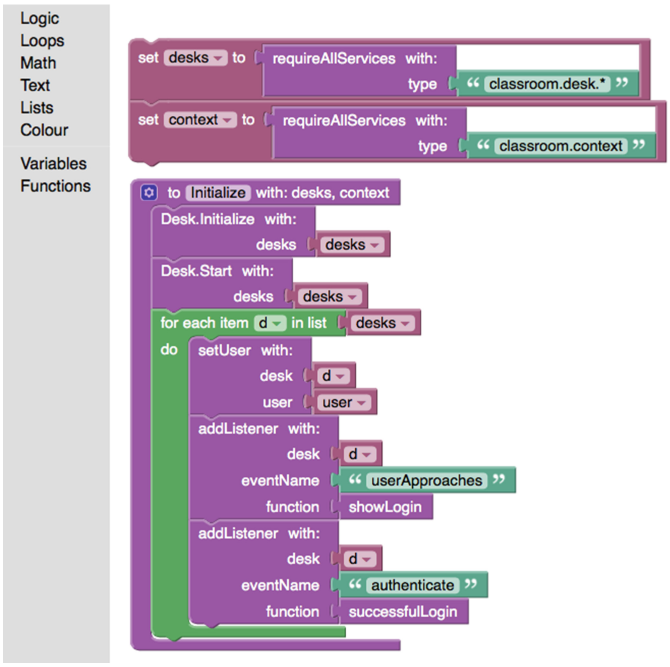
To facilitate programming by non-professionals, a visual editor is provided through which teachers can view and modify the existing scripts or create new from scratch. Teachers combine graphical blocks [24] that correspond either to basic programming structures (e.g., loops, variable definitions, arithmetic expressions, etc.) or functions stemming from service containers, in order to define the script's sequential logic while event-based programming is supported by connecting the appropriate event handlers to the available hooks. Finally, upon script creation the teacher is able to immediately deploy it to a single artifact instance or a family/group of instances or schedule its later deployment.

An illustrative example is depicted in Figs. 4 and 5, which provide the textual and graphical equivalent of the same program. Its objective is to discover all the smart desks available within the current context of use, and for each one install the appropriate event handlers to be called when a person approaches the physical device or when that person successfully authenticates himself as the authorized student for that desk.

4.4 Distributed Runtime Environment

Smart learning environments inherently follow a distributed computing paradigm where the various software components are located on networked computers or smart artifacts that communicate and coordinate their actions by passing messages. Those devices expose a set of core learning services (e.g., Smart Desk services such the PUPIL [27] and the ClassMATE [30] frameworks) along with an instance of the AmIClass Runtime Environment that can host the orchestration scripts written in ClassScript and defining learning scenarios. Such orchestration scripts can be deployed either locally on multiple targets to balance the overall workload (e.g., a script that initializes every desk after a successful student authentication), or centrally in the AmIClass cloud (e.g., the script that orchestrates the entire lecture and communicates with the AmI-RIA subsystem [35]).

Therefore, the teacher can define at any time whether a script will be deployed on a single or on multiple targets. The system validates whether the appropriate



```
set desks to requireAllServices with:
                          type " classroom.desk.* "
set context to requireAllServices with:
                          type " classroom.context "

to Initialize with: desks, context
  Desk.Initialize with:
              desks desks
  Desk.Start with:
              desks desks
  for each item d in list desks
  do setUser with:
              desk d
              user user
     addListener with:
              desk d
              eventName " userApproaches "
              function showLogin
     addListener with:
              desk d
              eventName " authenticate "
              function successfulLogin
```

Fig. 5. The graphical equivalent of the above sample ClassScript program

requirements are met (e.g., the required services are available in that host). When the target of a script is defined, the main classroom orchestrator deploys an instance of that script by appropriately notifying the local runtimes to download, parse and execute the appropriate script. From that point on, the execution control is performed solely by the local(s) orchestrator(s), while the main classroom orchestrator can manage it if necessary by propagating the necessary commands (e.g., suspend the script that controls the assistant facility when a quiz test has been started).

In case of a script modification that is currently active, the runtime environment ensures that any pending activities are finalized before restarting all of its instances in order to deploy the latest version. Currently, the update policy defines that during restart any events emitted in between will be dropped, therefore scripts reinstate their last state based on what has been saved before termination without cascading intermediate effects. In a future extension, a more sophisticated cascading policy has been planned; any events transmitted between script restarts will be stored in appropriate buffers and will be propagated in a timely manner.

5 Conclusions and Future Work

The emerging trend of mobile and ubiquitous computing has attracted numerous researchers and vendors to build educational applications that take advantage of the innovative technological affordances. Nevertheless, the majority of the available educational applications often do not meet their potentials due to the lack of tools that simplify their interplay with the environment and the contained affordances. To this end, this paper has presented an extensible software infrastructure that enables the design of purposeful and engaging learning activities and speedups prototyping for formal and informal learning environments, by combining and orchestrating cloud-based, ambient and pervasive facilities and services.

The following high-level components have been implemented: (i) a Service Mediator Agent that will integrate and provide access to ambient and pervasive services, (ii) a distributed Learning Activities Orchestrator that will monitor and orchestrate the learning process, (iii) a Context-Aware Adaptation Module that will facilitate the decision-making process, and (iv) a Design Environment for defining and configuring learning activities' scripts, while interoperability will be facilitate through a common Learning-oriented Ontological Framework.

Future work includes: (i) the extension of the programming environment to make use of the available semantic description of the services to validate their compatibility and provide useful insights for the end-users, (ii) the extensive testing of the overall infrastructure, (iii) the enhancement of the scenario modification process to support on-the-fly updates that support cascading of events that happened during the update process and finally (iv) the evaluation of the entire environment by HCI experts and teachers in terms of usability and acceptance and by experienced developers of ubiquitous applications in terms of completeness.

References

1. Attewell, J., Carol, S.-S.: Mobile Learning Anytime Everywhere. Learning and Skills Development Agency, London (2004)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
3. Bandyopadhyay, S., et al.: Role of middleware for internet of things: a study. Int. J. Comput. Sci. Eng. Surv. (IJCSES) **2**(3), 94–105 (2011)
4. Begel, A., Garcia, D.D., Wolfman, S.A.: Kinesthetic learning in the classroom. ACM SIGCSE Bull. **36**(1), 183–184 (2004). ACM
5. Brown, E.: Education in the Wild – A Comprehensive Overview of Location-Based Contextual Learning. STELLAR Network of Excellence (2008)
6. Caballé, S., Lapedriza, À.: Enabling automatic just-in-time evaluation of in-class discussions in on-line collaborative learning practices. J. Digit. Inf. Manag. **7**(5), 290–297 (2009)
7. Caballé, S., et al.: Towards a generic platform for developing CSCL applications using grid infrastructure. In: IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2004. IEEE (2004)
8. Cahill, V., Haahr, M.: Real + virtual = clever: thoughts on programming smart environments (1999)
9. Chang, C.-S.: WebQuest: M-learning for environmental education. In: Chen, T.-S., Hsu, W.-H. (eds.) IEEE International Conference on Wireless Mobile and Ubiquitous Technology in Education. IEEE (2010). doi:10.1109/WMUTE.2010.35

10. Chen, N.S., Graf, S., Hwang, G.J.: Adaptive learning systems. Knowledge management, organizational intelligence and learning and complexity. In: Encyclopedia of Life Support Systems (EOLSS), Developed under the Auspices of the UNESCO, Eolss Publishers, Oxford, UK (2012). http://www.eolss.net. Website eolss.net. Accessed 1 Feb 2013
11. Chin, J., Callaghan, V., Clarke, G.: End-user customisation of intelligent environments. In: Nakashima, H., Aghajan, H., Augusto, J.C. (eds.) Handbook of Ambient Intelligence and Smart Environments, pp. 371–407. Springer, New York (2010)
12. Chu, H.C., Hwang, G.J., Tsai, C.C.: A knowledge engineering approach to developing mind tools for context-aware ubiquitous learning. Comput. Educ. **54**(1), 289–297 (2010)
13. Cochrane, T.: Exploring mobile learning success factors. Alt-J **18**(2), 133–148 (2010). doi:10.1080/09687769.2010.494718. Routledge
14. Datta, D., Mitra, S.: M-learning: mobile - enabled educational technology. Innovating (2010)
15. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum. Comput. Interact. **16**(2), 97–166 (2001)
16. Dowling, C., Lai, K.-W. (eds.): Information and Communication Technology and the Teacher of the Future. LNCS (IFIP), vol. 132. Springer, New York (2003)
17. Giusto, D., et al. (eds.): The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications. Springer, New York (2010)
18. Gray, O., Young, M.: Video games: a new interface for non-professional game developers. In: ACM International Conference on Computer-Human Interaction (CHI 2007) (2007)
19. Green, T.R.G., Petre, M.: Usability analysis of visual programming environments: a cognitive dimensions framework. J. Vis. Lang. Comput. **7**(2), 131–174 (1996)
20. Guerra, M.A., Francisco, C.M., Girão, P.S.: PortableLab: implementation of a mobile remote laboratory for the android platform. In: 2011 IEEE Global Engineering Education Conference (EDUCON). IEEE (2011)
21. Holloway, S., Julien, C.: The case for end-user programming of ubiquitous computing environments. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. ACM (2010)
22. Hung, P.H., Hwang, G.J., Lin, Y.F., Wu, T.H., Su, I.H.: Seamless connection between learning and assessment- applying progressive learning tasks in mobile ecology inquiry. Educ. Technol. Soc. **16**(1), 194–205 (2013)
23. Hwang, G.J., Tsai, C.C., Yang, S.J.H.: Criteria, strategies and research issues of context-aware ubiquitous learning. Educ. Technol. Soc. **11**(2), 81–91 (2008)
24. Fraser, N.: Blockly: a visual programming editor (2013)
25. Kernighan, B.W., Ritchie, D.M.: The C Programming Language, vol. 2. Prentice-Hall, Englewood Cliffs (1988)
26. Korozi, M., et al.: Ambient educational mini-games. In: Proceedings of the International Working Conference on Advanced Visual Interfaces. ACM (2012)
27. Korozi, M., et al.: Towards building pervasive UIs for the intelligent classroom: the PUPIL approach. In: Proceedings of the International Working Conference on Advanced Visual Interfaces. ACM (2012)
28. Kubitza, T.: Towards a toolkit for the rapid creation and programming of smart environments. In: Workshop on End User Development in the Internet of Things Era. EUDITE (2015)
29. Leonidis, A., Korozi, M., Margetis, G., Grammenos, D., Stephanidis, C.: An intelligent hotel room. In: Augusto, J.C., Wichert, R., Collier, R., Keyson, D., Salah, A.A., Tan, A.-H. (eds.) AmI 2013. LNCS, vol. 8309, pp. 241–246. Springer, Heidelberg (2013)
30. Leonidis, A., et al.: ClassMATE: enabling ambient intelligence in the classroom. World Acad. Sci. Eng. Technol. **66**, 594–598 (2010)

31. Lieberman, H., et al.: End-user development: an emerging paradigm. In: Lieberman, H., Paternò, F., Wulf, V. (eds.) End User Development, vol. 9. Springer, Dordrecht (2006)
32. Maloney, J., et al.: The scratch programming language and environment. ACM Trans. Comput. Educ. (TOCE) **10**(4), 16 (2010)
33. Margetis, G., et al.: Enhancing education through natural interaction with physical paper. Univ. Access Inf. Soc. 1–21 (2014). doi:10.1007/s10209-014-0365-0
34. Martin, S., et al.: M2Learn open framework: developing mobile collaborative and social applications. In: UBICOMM 2010, The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (2010)
35. Mathioudakis, G., et al.: Ami-ria: real-time teacher assistance tool for an ambient intelligence classroom. In: eLmL 2013, The 5th International Conference on Mobile, Hybrid, and On-line Learning (2013)
36. De Marcos Ortega, L., et al.: Using m-learning on nursing courses to improve learning. Comput. Inform. Nurs. **29**(6), TC98–TC104 (2011). Topical Collection
37. Parsons, D.: Combining e-Learning and M-Learning: New Applications of Blended Educational Resources, vol. 154. Information Science Reference, Hershey (2011)
38. Peltz, C.: Web services orchestration and choreography. Computer **36**(10), 46–52 (2003)
39. Poslad, S.: Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction. Wiley, Chippenham (2009)
40. Savvaki, C., Leonidis, A., Paparoulis, G., Antona, M., Stephanidis, C.: Designing a technology–augmented school desk for the future classroom. In: Stephanidis, C. (ed.) HCII 2013, Part II. CCIS, vol. 374, pp. 681–685. Springer, Heidelberg (2013)
41. Schmidt, A.: Implicit human computer interaction through context. Pers. Technol. **4**(2–3), 191–199 (2000)
42. Sharples, M., Taylor, J., Vavoula, G.: Towards a theory of mobile learning. Proc. of mLearn 2005 **1**(1), 1–9 (2005)
43. Spector, J.M.: Conceptualizing the emerging field of smart learning environments. Smart Learn. Environ. **2014**(1), 2 (2014)
44. Turner, M., Budgen, D., Brereton, P.: Turning software into a service. Computer **36**(10), 38–44 (2003)
45. Uckelmann, D., Harrison, M., Michahelles, F.: Architecting the Internet of Things. Springer, Heidelberg (2011)
46. Ur, B., et al.: Practical trigger-action programming in the smart home. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems. ACM (2014)
47. Wang, M.: Learning anytime, anywhere: using mobile. Learning **9**, 1–7 (2008)
48. White, W., et al.: Better scripts, better games. Commun. ACM **52**(3), 42–47 (2009)
49. Yau, J.Y.-K., Joy, M.S.: Designing and evaluating the mobile context-aware learning schedule framework: challenges and lessons learnt, pp. 85–92 (2010)