BUILDING A SERVICE PLATFORM FOR ELECTRONIC LEARNING

GOTTFRIED VOSSEN, PETER WESTERKAMP
European Research Center for Information Systems (ERCIS)
University of Muenster
Leonardo-Campus 3
D-48149 Muenster
Germany
[vossen|westerkamp]@uni-muenster.de
Tel: +49 251 83 38 156
Fax: +49 251 83 38 159

Abstract

Elearning platforms and their functionalities resemble each other to a large extend. Recent standardization efforts in elearning concentrate on the reuse of learning material, but not on the reuse of application functionalities. The LearnServe system under development in our institute builds upon the assumption that a typical learning system is a collection of activities or processes that interact with learners and suitably chosen content, the latter in the form of learning objects. This enables a decomposition of the main functionalities of an elearning system into a number of stand-alone applications which can be realized individually or in groups. A proper implementation of these applications enables their reuse and gives learners a bigger flexibility of choosing content and functionalities to be included into their learning platform. Several technical possibilities exists to realize the interaction of these applications and are well-known from distributed systems construction. As will be shown in this paper, most of them are not appropriate in the elearning context. However, Web services exhibit enough flexibility, which is why they form the basis of the LearnServe system.

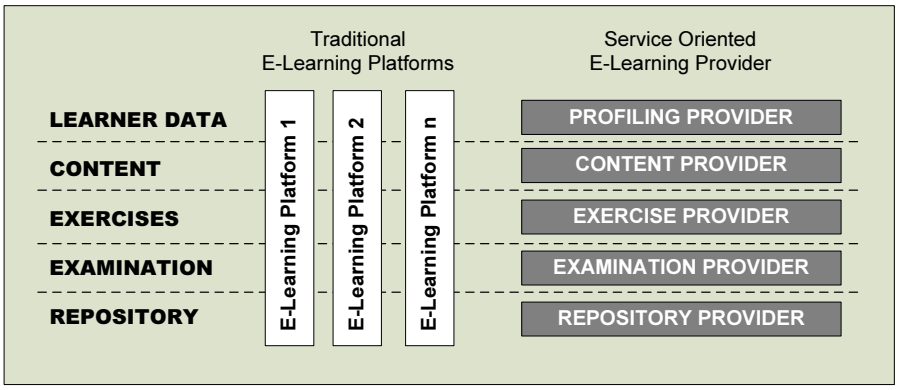
Keywords: Learning Objects, Web services, elearning, virtual elearning platform

1 Introduction

The optimization of processes in the value chain is a key factor for the survival of an enterprise. To do so, more and more organizations concentrate on their core competence by offering only those parts of the value chain the respective enterprise has special know-how, technologies, or abilities that are very valuable for a customer and not imitable for competitors. By using modern Internet technologies several of these companies are able to combine their abilities to produce products very efficiently. As the combination of these enterprises is not obviously transparent for a customer who interacts with such an organization, these combinations are known as *virtual companies* [1]. Virtual companies are flexible in their configuration and are able to change partners on demand to optimize the output for the customer.

The concepts of the virtual company can be transferred to the field electronic learning (elearning) to offer virtual elearning platforms. Although the concept of a virtual company is not new, elearning manufacturers still only concentrate on their core competence in limited areas. Traditional elearning platform resemble one another to a large extend and implement the various functionalities in very different qualities. In particular, all systems comprise a maintenance of user data, a tracking of user actions, a module to display learning content, authoring features, exercise modules, and search mechanism for the discovery of content. However, each platform implements these functionalities and a specialization can only be found for authoring tools that try to build Learning Objects (LO, see [2, 3]) to be used in different Learning Management Systems (LMS) and in the offering of LOs to be bought and included into a system. The reuse of LOs in different systems is only possible if they adhere to common standards. For other functionalities this process of offering specialized applications has just begun (see, e.g., [4, 5, 6]).

By subdividing the functionalities of an elearning platform into several applications that can henceforth be included by various virtual elearning platforms via the Internet, providers of functionalities would be able to offer only those parts of platforms they are really experts in. This includes, for example, content in special fields as separate applications, advanced search functionalities for content, special simulations to deepen explanations



of content, as well as administration services like tracking, authorization etc. The new organization of the company offerings is sketched in Figure 1 and is obviously not exhaustive. Important is the focus on a selected part of a traditional platform instead of an entire system.

By using distributed functionalities as shown in Figure 1, virtual elearning companies can offer very flexible platforms. The provision of parts of elearning functionalities by different manufacturers leads to new demands in the design of market strategies and pricing mechanisms because not a real software product but a service is sold. Technically the difference is that the functionality or the content is offered on demand by a remote server somewhere on the Web and that there is no centralized elearning server and no centralized content storage anymore. Instead, services with equal functionalities can be exchanged on the fly depending on availability and response time. The scope of this paper is to describe the technical background of elearning services as it is possible to implement functionalities based on various techniques. This includes the use of well established approaches from the field of distributed systems like Remote Procedure Calls (RPC), CORBA, DCOM as well as message based approaches, in particular Web services.

The organization of this paper is as follows: Section 2 covers user requirements for a virtual elearning platform. Section 3 sketches the already mentioned approaches to implement functionalities for a virtual elearning platform and points out why traditional middleware approaches are not the optimal choice for implementing decomposed applications. It also motivates the use of Web services; a work-around for most of the limitations is presented in Section 4, where the Web services based LearnServe system under development at the University of Muenster is presented. Section 5 concludes the paper.

2 A Learner's View

Many institutions nowadays offer courses for tertiary education. In order to pass a course, participants get checklists that describe the content of teaching. Based on these descriptions, the learners are relatively free to choose one or more content providers. Some courses may end in exams, which can be taken at different institutions a learner can again choose from. Even though there is typically a well-defined order of taking exams of each part within a course, the order of exams on smaller units may not be fixed, and even the combination of exams on these units might not be fixed (e.g., take two out of four).

Traditional elearning platforms do not provide the flexibility a learner needs in tertiary education. Platforms are normally centralized and offer courses with well-defined content instead of checklists. A major problem for learners is that platforms implement closed communities, leaving no room for including further, personally preferred features. Often learners are not allowed to upload additional content to be used as Learning Objects. Some systems enable a simple upload mechanism, but do not provide functionalities to offer this content again for a self-directed learning of other learners in the system. Instead, learners are forced to use what is allocated by tutors and have no flexibility to choose for a self-direction - neither in respect of functionality nor content.

If elearning providers would concentrate on their core competencies and would offer elearning functionalities as components that can be used via the Internet as a service instead of physical software components, this would optimize the platform for a learner. Indeed, learners would have the ability to choose among content from different authors and styles within a course, and the content can be selected and adapted to a learners needs. For example, this can be a simulation of a certain fact or special content to be learned in a self directed way and provided by the author of choice.

3 A Technical View

Already in 1968, McIlroy has had the vision of a software component industry that would offer groups of routines for any given job [7] to be reused in various software engineering projects. This idea was the first step to provide software offerings by concentrating on specialized functionalities:

"..., yet software production in the large would be enormously helped by the availability of spectra of high quality routines, quite as mechanical design is abetted by the existence of families of structural shapes, screws or resistors."

Starting from this concept and exploring the idea for many years now, researchers and engineers have still not been able to agree on a precise definition of a component in computer science. Many modern programming languages help to fulfill McIlroy's dream at least to some extend, as they provide small scale programming libraries (e.g., *java.util*), technology abstractions such as ODBC and JDBC, special purpose software components like XML parsers, large scale standardized frameworks (e.g., *java.swing*) or large scale standardized containers such as database engines. However, these components largely depend on their providers because their documentation, component structures, interface definitions and behaviour descriptions are mostly proprietary. It goes without saying that in software development, object-oriented technologies contributed decisively to an increase of the reuse and encapsulation of software. Nonetheless, it clearly suffers from several drawbacks, for example the fact that objects often only can be composed and can cooperate if they are written in the same language. Moreover, they have to be tightly coupled if they are executed in the same process and data space. In addition, their interface descriptions focus on the incoming interface, whereas the outgoing interface is mostly implicit. This makes it more difficult to deploy objects independently.

Current software systems are often distributed over networks and different platforms. Particularly in the future, they will have to be able to be integrated and to interact with each other, like it is the case in the distributed elearning system. Apparently, objects are not able to handle these upcoming challenges of such information systems on their own. Instead, more and more components (which can be made of objects) will offer their functionality to be called remotely by providing well defined interfaces and communication mechanisms; they can then be accessed by other components via the Internet and enable, for example, a construction of a (distributed) virtual elearning platform, where each functionality is offered by another manufacturer.

3.1 Usage of Middleware

Different approaches of how components can be used and located to build distributed systems are in use today. Basically all of them build on the client-server paradigm. The construction of distributed systems can be simplified by leveraging middleware. Middleware is layered between the operating system and the application components and can cope with heterogeneity. It enables software developers to build distributed systems across networks by facilitating communication and coordination of distributed components at a higher level than the one an operating system offers. Based on the techniques which middleware products use for the interaction between distributed components, they can be classified into RPC-based systems, transaction processing (TP) monitors, object-based systems and message-oriented systems [8]. The interaction of RPC-based systems is based on remote procedure calls, whereas TP monitors are an enhancement of RPC-based systems by distributed transactions. Object-based systems (e.g., CORBA and DCOM) use remote object requests as the underlying interaction paradigm. Message-based systems communicate by passing messages and include Web services, which are the fundamental of our LearnServe system described in Chapter 4.

To build a virtual elearning platform, the logical level of integrating functionalities is important for the choice of the technique, since it is not possible to use approaches that have to be integrated at a programming language or component level, where the learner has to (re-)compile a client system after adding new components. This is the case with RPC-based systems and for TP monitors which makes them irrelevant for a virtual elearning platform. Even sometimes in CORBA and DCOM this is necessary if dynamic models are not used. The level of integration should be higher than in these approaches, and an integration should be doable in a plug-and-play manner, at least for functionalities such as content or exercises. Another problem with conventional approaches is that there is no obvious place to put the respective middleware, since the basic idea was to place the middleware between the applications that have to interact [8]. Obviously, in case of a virtual elearning system it is difficult to properly position middleware since the learner should be able to select the functionalities depending on his own preferences on the fly. When offering this flexibility to the learner, not all providers can be known at the time of implementing the client software.

An installation of middleware systems on the client side would be a very challenging task for a learner without advanced information technology knowledge, due to the complex nature of these systems, e.g., due to security and transaction handling. In an elearning offering, as assumed here, there is neither a central instance nor an administrator to install and supervise the system. This may be different in campus-wide systems or where a Web portal serves as client, but is not the case in an open system where also stand-alone applications might implement the client. Web services as described below are installed on the provider's machine. The only thing a user of the service needs is a Web service client to use the services. If the client is offered as a portal on the Web a learner just needs a Web browser, which makes things easy to install and maintain.

Many of the technologies mentioned above are not compatible to each other, not even inside the same middleware category. As a consequence, all peers in a given environment must use the same form of RPC, the same object model (CORBA, DCOM, etc.), or a unique form of messages. Thus, the operating system is implicitly forced to the connected clients as most of them are not platform-independent, in particular in the case of DCOM. DCOM is used primarily on Windows machines, although there are some implementations for other operating systems. Apparently, the participants of a distributed elearning environment do not have agreed upon a special object model nor on an operating system. In addition, they are not able to agree on a certain message format. However, even if all users had agreed on using a CORBA system, problems would occur due to the fact that CORBA implementations of different suppliers may be incompatible [9]. The situation is even worse in open scenarios that should work across company boarders or for learners that work at home and should offer a plug and play integration of components as no company guidelines exists that regulates systems hardware and software.

In the communication process, frontiers of companies may become a problem as Internet connected platforms will typically be shielded by firewalls. An open elearning platform also serves company employees to do training on the job. The usage of protocols like CORBA to communicate with elearning providers outside company boarders may cause errors due to problems resulting from closed ports etc.. Indeed, CORBA uses about 100 ports which cannot be considered to be open. Using DCOM through firewalls also causes trouble because it dynamically allocates one port per process (configurable through the registry), and additionally requires the ports for UPD and TCP to be open. To use DCOM via port 80 and enable a use with firewalls, tunnelling TCP/IP as the underlying transport protocol can be used. However, this is not very reliable, does not work through all firewalls, and introduces additional limitations (e.g., lack of callback support) [10] as well as administration efforts that cannot be handled by average users. By opening further ports, security guidelines may be disregarded. A direct connection of two peers is not possible, either, if the company uses a proxy. The proxy problems can be fixed, but the performance will go down. On the other hand, Web services do not need additional ports as their communication is based on transport protocols which are already in use (in particular HTTP).

The overall finding from the above discussion is that conventional approaches bear a couple of problems that make them not easy to implement and even to use. Particularly in the field of elearning, where learners should be able to select distributed components at runtime, conventional approaches suffer from an easy handling and from compatibility problems. In conclusion, DCOM and CORBA can be used in local area networks with little heterogeneity and a central administration. RPC and TP monitors have to be included on a programming level and are not flexible enough for a plug and play integration. As described next, Web services are a reasonable choice in open environments that act across the Internet, where a lot of heterogeneity exists. It should be mentioned at this point that all techniques can be wrapped by Web services using approaches like J2EE, .NET and others and can thus still be used.

3.2 Web Services to the Rescue

Web services [11] generally enable partners to easily (re-)use applications via the Internet. Web service are characterized as real services that hide all details concerning their implementation and the platforms they are based on. A Web service is essentially a stand-alone software component that has a unique URI (the Uniform Resource Identifier is a unique address) and that operates over the Internet and particularly the Web. The basic premise is that Web services have a provider and (hopefully) users or subscribers. Web services can be combined to build new ones with a more comprehensive functionality. Clearly, Web services need to be interoperable. Moreover, they have to be independent of the operating systems; they should work on every Web service engine regardless of their programming language; and they should be able to interact with each other.

To achieve these goals, Web services are commonly based on standards; currently, the most common ones are the XML-based specifications SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration), and WSDL (Web Services

Description Language). For the composition of Web services and in order to build more complex services out of given ones, XML-based languages such as BPEL4WS for specifying the process logic are used, often in connection with subordinate standards such as WS Coordination or WS Transaction for the clarification of tasks (such as communication coordination and observation of transactional boundaries, resp.) that arise during service activation. The benefits of a Web services architecture is well recognized in the business-to-business (B2B) area, where companies already use it for enterprise application integration, B2B integration, application construction and a flexible approach to outsourcing, a better access to business functions, a free choice of the best technology platform in each situation, and location and device independence. Even in terms of interoperation of business-to-consumer (B2C) systems, Web services as the basis of a service-oriented architecture (SOA) are currently seeing a growing importance.

because Web services are independent of the platform, of the operating system, and the programming language and are not based on a protocol of a special company because the entire communication is based on XML. Messages are transmitted using a standardized format and the Internet via port 80 which can be considered to be open. With the open standardization, a plug-and-play integration of services at runtime can be achieved; a corresponding architecture for a virtual elearning platform will be presented in the next chapter.

4 Elearning Services: The LearnServe Architecture

Our LearnServe system starts from the perception that a typical learning system is a collection of activities or processes that interact with learners and suitably chosen

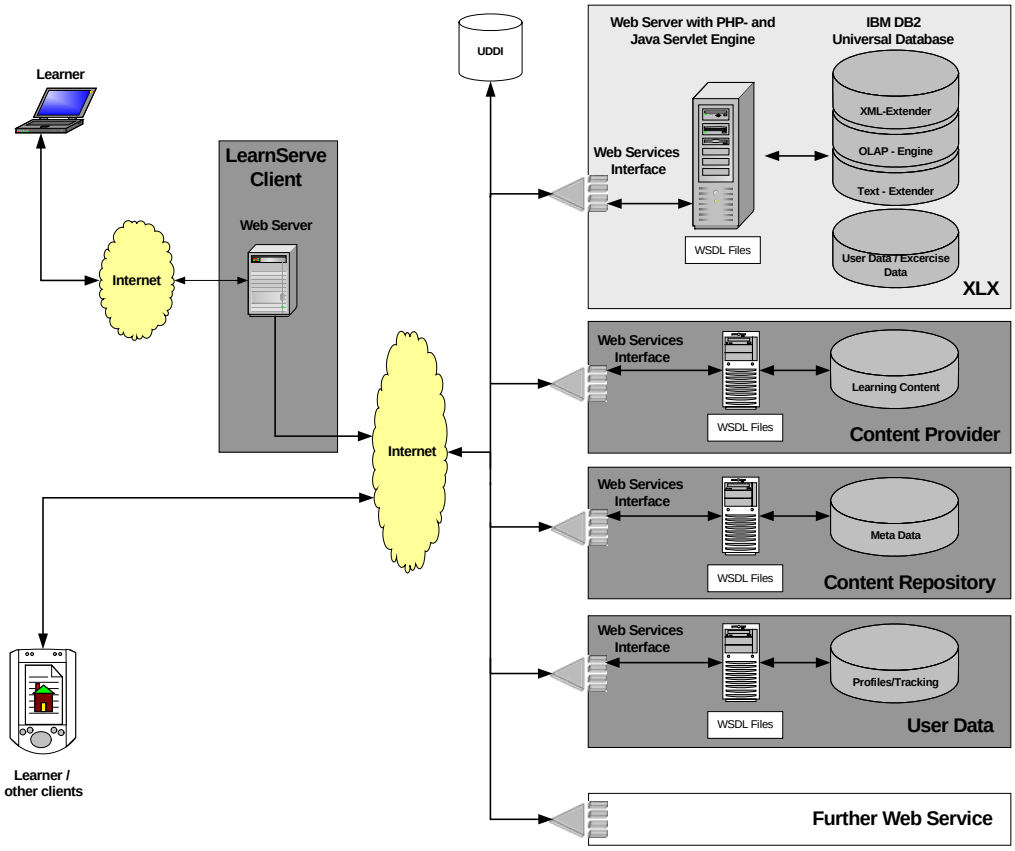


Figure 2: High-Level LearnServe Architecture.

The problems of conventional middleware can be solved

content, the latter in the form of learning objects. This enables us to decompose the main functionality of an

elearning system into a number of stand-alone applications, which can then be realized individually or in groups as Web services. Conversely, relevant Web services are offered by a number of providers, and can be composed to build the functionality of a traditional elearning platform — now as a virtual elearning platform. By the same token, content offered as Web service can be composed into course units or complete courses. Intuitively, learners can search for content that matches their needs, book it, pay for it, and finally consume it, all by composing Web services appropriately.

Web services for elearning can be classified into user facing, presentation oriented services that should be able to be integrated in a plug-and-play manner. Data oriented services, on the other hand, can be seen as functionalities that cover administration aspects such as authorization, tracking, etc. and do not have to be included by the learner on the fly but are used by the integration client as specified by the administrator or programmer. Generally, we try to design such services using common tools and languages as far as possible. Consequently, we rely upon established Web service standards (e.g., UDDI, WSDL), since they appear sufficient for our purposes. The implementation of Web services in the area of elearning facilitates a considerable flexibility for the users of the system both in usage of functionalities and selection of content.

As shown in Figure 2, LearnServe is divided into two parts: client software and Web services provided by several suppliers. A LearnServe client is the access point for users who utilize the learning services. These services are implemented on distributed servers and in particular include authoring, content, exercise, tracking, and discovery services as well as communication services such as email and message boards. The exercise services are provided by our xLx system [12], that was enhanced to offer its functionality as a Web service and can thus already be used in external systems. Of course, the use of learning services is not limited to our clients because the implementation of the entire functionality as Web services enables an integration of the elearning functionality directly into a business application (e.g., a CRM or an ERP system) to interact with applications, processes and information. The learning Web services can also be used on mobile devices if there is an appropriate client for that device.

Building a non-centralized system by combining several Web services to achieve the same functionality as in traditional elearning systems leads to the problem of managing the content for the learners and searching for services to gain the desired functionality in the moment of demand. To this end, LearnServe uses a UDDI registry [13] to search for Web services as is common in the area of Web services. However, UDDI is not appropriate for content services since the storage of additional meta-data about the content is not supported adequately. In such an organization, learning objects cannot be imported to a particular learning management system, either. Instead, content needs to be stored on distributed servers and be called on demand. This leads to presentation problems since typical Web services are data-oriented, but the presentation aspect is important to understand the content to be learned. To mitigate the compatibility problems of content integration, the system uses recent standardizations for reuse, discovery and exchange of content.

The discovery process is supported by the LearnServe repository [14] for learning object publication and search, and essentially adapts the UDDI framework used for commercial Web services to an elearning context. It distinguishes itself by the fact that the repository itself contains centralized data about learning objects, i.e. all meta-information, while the actual content that it refers to can be arbitrarily distributed. To use any content, the underlying platform calls the desired learning object, which is then executed by a presentation Web service and delivered to the learner. This presentation service enhances the information about the content as described in the WSRP standard [15] for the plug-and-play integration and thus additionally enables an adaption of presentation information depending on the learners' needs. We are thus able to tackle some of the problems arising when realizing a service platform, including (1) storing learning content in a distributed fashion, and (2) dynamically exchanging content if necessary or appropriate. For example, this can be based on the individual profiles of the learners and the course definitions an author has published in the LearnServe repository.

5 Conclusions

We have implemented a first prototype of LearnServe [16] and are currently enhancing it to provide the complete elearning functionalities of traditional platforms. Our attempts have been motivated by two major observations: on the one hand, many custom elearning platforms can only present their material inside the platform; and on the other hand, Internet-based Web services are becoming ubiquitous, both at a professional and at a personal level. A service-oriented elearning system results from a perception of the various tasks and activities that are contained in such a system as processes or as workflows; using appropriate encodings of objects and tasks in UDDI and WSDL forms and documents enable broad exchanges, flexible compositions, and highly customized adaptations possible. This even allows a reuse of services already offered on the Web, such as payment and cashing

services, chat rooms, or conferencing (via platforms such as Webex).

References

[1] Porter M. E. (1985): Competitive Advantage, Collier Macmillan Publishers, London.

[2] Vossen, G., P. Jaeschke (2002). Towards a Uniform and Flexible Data Model for Learning Objects. In Proc. 30th Annual Conf. of the Int. Bus. School Computing Assoc. (IBSCA), Savannah, Georgia, July 2002, pp. 99-129.

[3] Vossen, G., P. Jaeschke (2003): Learning Objects as a Uniform Foundation for E-Learning Platforms. In Proc. 7th International Conference on Database Engineering and Applications (IDEAS), Hong Kong, China, IEEE Computer Society Press, 2003, pp. 278-289.

[4] Bry, F., N. Eisinger, G. Schneemeyer (2003). Web Services for Teaching: A Case Study.. In Proc. First International Conference on Web Services (ICWS'03), Las Vegas, USA, June 2003.

[5] Blackmon, W.H., Rehak, D. R. (2003): Customized Learning: A Web Services Approach. In Proc. Ed-Media 2003, June 2003.

[6] Chen,W. (2002): Web Services - What Do They Mean to Web based Education?. In Proc. Int. Conf. on Computers in Education, Auckland, Newzealand, December 2002, pp.707-708.

[7] McIlroy, M. Douglas (1968): Mass Produced Software Components. In Report of NATO Conference on Software Engineering, NATO Science Committee, Garmisch, Germany, pp. 138-155.

[8] Alonso, Gustavo; Casati, Fabio; Kuno Harumi; Machiraju, Vijay (2004): Web Services. Concepts, Architectures and Applications. Springer Verlag, Berlin, Germany.

[9] Coenraets, Christophe (2001): Web Service: Building the Next Generation of E-Business Applications. Macromedia White Paper.

[10] Wasznicky, Martin (2002): Using Web Services Instead of DCOM. http://msdn.microsoft.com/library/default.asp?url=/library/enus/dndotnet/html/webservicesdcom.asp. 2004-11-16.

[11] Newcomer, E. (2002). Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley.

[12] Huesemann, B.; Lechtenboerger, J.; Vossen, G.; Westerkamp, P. (2002): XLX - A Platform for Graduate-Level Exercises; in Proc. International Conference on Computers in Education (ICCE2002), Auckland, New Zealand, December 2002, pp. 1262-1266.

[13] Universal Description, Discovery and Integration (UDDI). http://www.uddi.org.

[14] Vossen, G., P. Westerkamp (2003). UDDI for E-Learning: A Repository for Distributed Learning Objects. In Proc. 2nd International Conference on Information and Knowledge Sharing (IKS2003), Scottsdale, AZ, USA, November 2003, pp. 101-106.

[15] Kropp, A., C. Leue, R. Thompson (2003). Web Services for Remote Portlets Specification, OASIS Standard Version 1.0, August 2003.

[16] Vossen, G., P. Westerkamp (2003). E-learning as a Web service (extended abstract). In Proc. 7th International Conference on Database Engineering and Applications (IDEAS), Hong Kong, China, IEEE Computer Society Press, pp. 242-249.