A Service-oriented Architecture for Student-Oriented Courses

Naseem Ibrahim[1]

Abstract— In the last few years, the popularity of online degrees has dramatically increased due to the higher cost of education and the increased number of adult students. In the current online education model, the school specifies the courses required to obtain a degree. For each course, the instructor specifies the course elements including teaching method and assessments. But different students have different capabilities and constraints. Most institutions provide the same courses. A student should be able to select the course that best matches his capabilities and constraints as long as it satisfies the required course outcomes. To achieve this goal, we propose the use of Service-oriented Architecture (SOA). This paper introduces an extended service-oriented architecture. The elements of the architecture and an extended service definition model is discussed in this paper.

Keywords— Student-oriented, SOA, Context, Service Model.

I. Introduction

THE last few years have seen dramatic changes in higher education. The cost of edu- cation has been increasing. The demand for college graduates has also been increasing which resulted in larger number of adult students. All of this led to a dramatic increase in online education. Many institutions are providing online degrees and the number is increasing every semester. A lot of these institutions provide similar degrees including similar courses.

With current online degrees, a school defines the classes a student needs to take to complete a degree. For each course, the instructor defines the elements of the teaching process with respect to instruction method, assessment types and schedule.

But research in education has long proved that different students have different capabilities and needs. Students have different learning pace and styles. With the current online learning model this is not considered. A student should be able to choose a course from any institution. The course that best matches its capabilities and needs. The only constraint is that the course should satisfy the required outcomes. We call this model *student-oriented learning*.

To achieve the student-oriented learning model, this research suggests the uses of *Service-oriented Computing (SOC)* [1]. SOC is a computing paradigm that uses *service* as the fundamental element for application development processes. An architectural model of SOC in which *service* is a first class element is called *Service-oriented Architecture (SOA)* [2]. We believe a course can be represented as a service and can be provided by a service-oriented architecture.

Current service-oriented architectures in its current state are not sufficient for achieving this goal. The provision of services in such architectures depends on the functionality of the service which is not enough. Hence, in Section 2, we propose an *extended service-oriented architecture (ESOA)*.

In the newly introduced ESOA, a course is specified as a service. But current service definitions are not rich enough. Hence, in Section 3, we extended the definition of a service by including the concept *Context* to support the rich definition of courses. We represent student capabilities and constraints using the concept "Context". Context has been defined [3] as the information used to characterize the situation of an entity. This entity can be a person, a place, or an object. The context representation and the logic of context proposed by Wan [4] for reasoning about context-awareness are suitable formalisms for enriching SOA modeling. The extended service is defined formally to support formal verification.

In ESOA, a course requester can specify his requirements using a rich definition. Section 4 introduces a Student-oriented course definition. It gives a brief overview of how a course requester can specify all his requirements and needs.

In ESOA, there is a Unit that is responsible for mapping the course requester needs with available courses. It is also responsible for ranking courses and composing them if necessary. This unit is called the Course Mapping Unit. Section 5 provides a discussion of the unit and its functionality.

Section 6 presents a brief study of related service definition approaches and compositions. Finally, Section 7 presents some concluding remarks and future work..

II. Extended SOA

Traditional SOA model consists of three main modules, the service provider, the service requester and the service registry. The service provider publishes a service definition in the service registry. The service requester searches the service registry and selects from the published services. After selecting a service, the service requester interacts with the service provider by sending requests and receiving responses.

In traditional SOA, the publication, discovery and execution of services are heavily based on the functionality of the services. The service provider publishes the functionality of the service in the registry. The service requester searches the registry looking for services that matches its requirements in

[1]Department of Computer Science and Software Engineering, School of Engineering, Penn State Behrend, Erie, PA, USA

terms of functionalities. But such architectures are not sufficient for the publication of our student-oriented courses. Hence, this section introduces an Extended SOA (ESOA) that supports student-oriented courses.

ESOA enables course providers to define rich courses for the provision of student- oriented courses. It also enables course requesters to obtain courses that best match their requirements while considering their capabilities and constraints. Fig 1 shows the architecture of ESOA. ESOA consists of the following elements:

1. **Course Requester**: It is the entity that is requiring a course. It represents the client side of the interaction. It is usually a student who is looking for a course that best meets his requirements while respecting the student's capabilities and constraints.
2. **Course Provider**: It is the entity that provides a course. Course providers publish course descriptions on registries to enable automated discovery and invocation.
3. **Student-oriented Course Requirement Definition**: In ESOA, a course requester is able to specify and list all the course requirements, his capabilities and constrains in a rich definition. This entity enables this rich definition.
4. **Student-oriented Course Definition**: To enable the best possible matching and discovery of courses, course providers has to publish a rich definition of a course. Traditional definition of services that relay on service functionality is not sufficient. Hence, a rich course definition is required. This entity is responsible for achieving this.

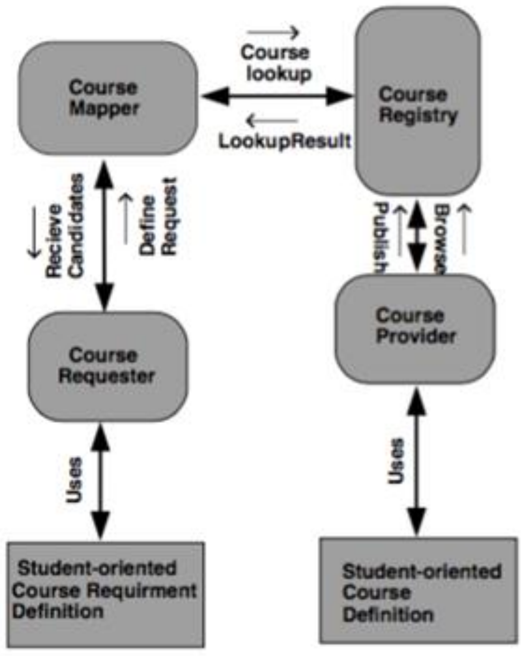


Fig. 1: Extended SOA Architecture

5. **Course Registry:** This entity is responsible for enabling the discovery of student- oriented courses. Course providers publish their rich course definition in the course registry. The course mapper searches the course registry looking for courses that matches the course requester requirements while respecting the course requester capabilities and constraints.
6. **Course Mapper:** This unit is responsible for three main roles. First, it matches the requirements of the course requester to the available course in the course registry. The novelty in the matching process is that is does not only focus on the functional requirements, it takes into consideration the capabilities and constraints of the course requester. Second, it ranks available courses is case of multiple matches. Third, it enables the composition of courses.

III. Student-Oriented Course Definition

As presented in the previous section, a course provider describes a course using a student-oriented course definition. This section extends traditional services to specify student-oriented courses.

In a traditional service the main component of a service is the functionality. But this is not sufficient for the specification of our rich student-oriented courses. Hence, we extend traditional services by adding nonfunctional properties, attributes, context, and legal rules. These elements are encapsulated in what is called ExtendedService. An *ExtendedService* is divided into the following parts, as shown in Fig 2.

1. **Functionality:** Its definition includes the function *signature*, *result*, *precondition* and *postcondition*. The *signature* part defines the function *identifier*, the invocation *address*, and the *parameters* of the function. The function invocation has the same effect as in a programming environment, since service function is an autonomous program. Each parameter has an *identifier* and a *type*. The *result* part defines the returned data of the service function. The *precondition* should be made true, either by the service provider or the consumer, in order to make the function available. The *postcondition* is guaranteed by the service provider to be true after service execution

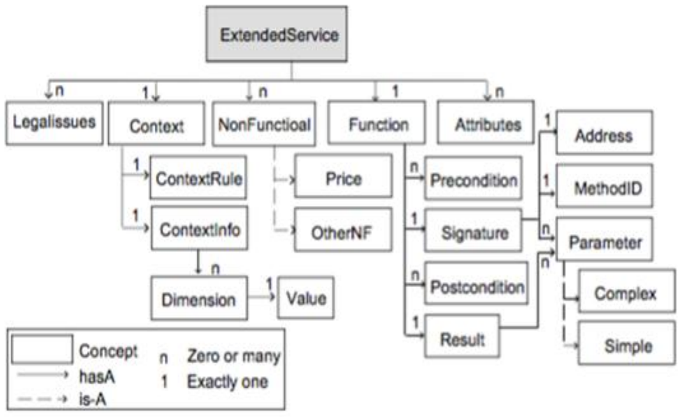


Fig. 2: ExtendedService Structure

2. Nonfunctional properties: The nonfunctional properties associated with the service are listed in this section.

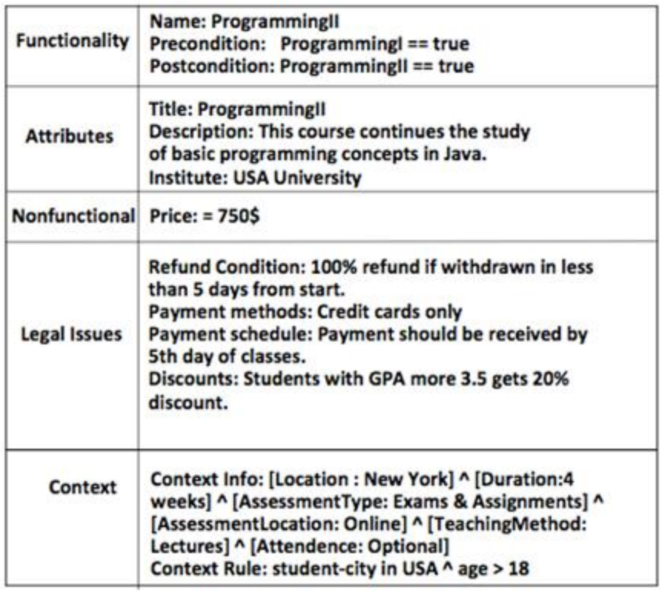
Pricing information, which can itself be a complex property expressing different prices for different amount of buying, is an example of nonfunctional property.

3. Attributes: Every attribute is a type-value pair. Attributes provide sufficient information that is unique to a service. As an example, for providing a course the appropriate attributes may include title of course and institution name.

4. Legal issues*:* Business rules and trade laws that are enforced at the locations of service provision and service delivery are included in this section. Example policies govern *refund*, *administrative charges*, *penalties*, and *service requesters rights*. Such rules are expressible as logical expressions in predicate logic.

5. Context: The context part of the contract is divided into *context info* and *context rules*. The contextual information of the service provider is specified in the *context info* section. The situation or context rule that should be true for service delivery is specified in *context rules* section. It is the responsibility of the service requester to validate the *context info* for obtaining the service, and it is the responsibility of the service provider to validate the *context rules* at service delivery time.

Example 1. Fig 3 illustrates an example of a service that was modeled using the novel ExtendedService definition. The service is for a Programming II course that is being provided by USA University.



Functionality	Name: ProgrammingII Precondition: ProgrammingI == true Postcondition: ProgrammingII == true
Attributes	Title: ProgrammingII Description: This course continues the study of basic programming concepts in Java. Institute: USA University
Nonfunctional	Price: = 750$
Legal Issues	Refund Condition: 100% refund if withdrawn in less than 5 days from start. Payment methods: Credit cards only Payment schedule: Payment should be received by 5th day of classes. Discounts: Students with GPA more 3.5 gets 20% discount.
Context	Context Info: [Location : New York] ^ [Duration:4 weeks] ^ [AssessmentType: Exams & Assignments] ^ [AssessmentLocation: Online] ^ [TeachingMethod: Lectures] ^ [Attendence: Optional] Context Rule: student-city in USA ^ age > 18

Fig. 3: Programming II Extended Service

The service requester can specify an *Exact* match for all his requirements. In this case, the mapping unit will match the requirements with the candidate courses and filter the courses that provide the exact match. It then passes the candidate courses to the course requester

IV. Course Mapping Unit

The course mapping unit is responsible for three main roles. First, it matches course re- quests with available courses in the course registry. Second, it ranks candidate services. Third, it composes multiple courses if necessary. Below is a brief discussion of these roles.

4.1 Course Matching

The Mapping Unit received course requests from the course requester. It will then con- tact the course registry looking for services that provide the same course. The registry will respond with all services that provide the required course.

4.2 Course Ranking

In many cases, no exact match is possible or the course requester set weights that are not *Exact* for all the requirements. In such cases the mapping unit ranks candidate services. The ranking algorithm takes into consideration all the requirements of the course requester and the weights assigned. It will then pass the set of ranked courses to the course requester. A course that provide a closer match to the requirements will be listed first while the course that provide the least match to the requirements will be ranked last

4.3 Course Composition

In some cases, no single course can meet the requirements of the course requester. Hence, a composition of multiple courses is necessary. This is performed by the Map- ping Unit. The Course Mapping unit creates a *course expression* involving the names of *ExtndedServices* and composition constructs. All composition constructs in a course expression have the same precedence, and hence a course expression is evaluated from left to right. To enforce a particular order of evaluation, parenthesis may be used. The result of evaluating a course expression is a *ExtendedService*.

In the context of courses, two types of compositions are necessary sequential and parallel. In sequential compositions one course is a prerequisite for another course. In a parallel composition two courses can be completed concurrently.

1. Sequential Construct $\gg$: Given two *ExtendedServices* A and B, the service expression A $\gg$ B defines an *ExtendedService* C which is the sequential composition of A and B. The intended execution behavior of the

ExtendedService C is the execution behavior of B after the execution of A.

2. Parallel Construct ||: Given two *ExtendedServices* A and B, the service expression A||B defines the parallel composition of A and B. The parallel composition A||B service models the concurrent executions of *ExtendedServices* A and B. Therefore, the resulting behaviour of this composite service should be the merging of their individual behaviours in time order

V. Student-Oriented Course Requirement Definition

In ESOA, a course requester specifies his requirements and constraints and passes them to the Course Mapping unit. The specification of the requirements and constrains is done using the *Student-oriented Course Requirement Definition (SOCRD)*.

Fig 4 shows the structure of a course request defined using SOCRD. Each course request will consist of the four parts *required function*, *required legal issues*, *required nonfunctional properties*, and *requester and consumer context*. The course requester is responsible for defining these requirements.

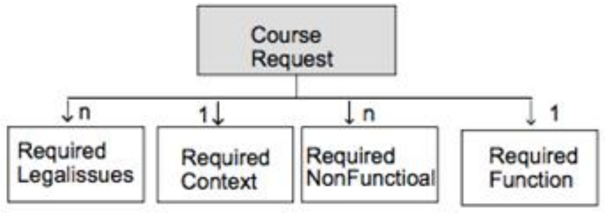


Fig. 4: Course Request Structure

The course requester can also assign a weight to each requirement. This weight defines the priority of each requirement and is used in ranking the set of candidate course when the Course Mapping unit performs matching.

1. *Required Function*: The required functional properties define the functionality required by the course requester and is defined in terms of the functionality name, preconditions and postconditions. For each element the service requester can as- sign a weight.
2. *Required Nonfunctional Properties*: The required nonfunctional properties defines the nonfunctional properties required by the course requester. The definition of the nonfunctional properties in SOCRD is identical to the definition of the non- functional properties in ExtendedService. The only exception is the addition of the weights.
3. *Required Legal Issues*: This section contains the required legal rules specified by the course requester. Its definition is also identical to the definition in ExtendedService with the addition of the weights.
4. *Required Context*: This section includes the contextual information of the course requester and provider. It also uses the same definition of the contextual information in ExtendedService. A weight value can also be added to each requirement.

VI. Related Work

Related SOA, such as *eFlow* [5], *SELF-SERV* [6] and *SWORD* [7] do not provide support for including contextual information. On the other hand, frameworks such as SeGSeC [8], SHOP2 [9] and Argos [10] do provide some support to include contextual information but context is not formally represented and the relationship between the service elements is never considered. To our knowledge, no published framework supports all the features of *ESOA*. Hence, ESOA is novel in its ability to support the provision and discovery of student-oriented courses

VII. Conclusion and Future Work

To support the publication, discovery and provision of student-oriented courses, this paper has presented an extended service-oriented architecture (ESOA). It has also presented an ExtendedService model for the specification of student-oriented courses. The other elements of the ESOA has also been discussed including the Course Mapping Unit and the Course Requirements Definition. We are currently working on a complete implementation of the newly introduced architecture and associated tools.

References

[1] D. Georgakopoulos and M. P. Papazoglou, *Service-Oriented Computing*. The MIT Press, 2008.

[2] T. Erl, SOA *Principles of Service Design*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2007.

[3] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, 2001. http://dx.doi.org/10.1007/s007790170019

[4] K. Wan, "Lucx: Lucid enriched with context," Phd Thesis, Concordia University, Montreal, Canada, January 2006.

[5] F. Casati, S. Ilnicki, L.-j. Jin, V. Krishnamoorthy, and M.-C. Shan, "Adaptive and dynamic service composition in eflow," in *Proceedings of the 12th Int'l Conference on Advanced Info. Systems Engineering*. Springer-Verlag, 2000, pp. 13–31. http://dx.doi.org/10.1007/3-540-45140-4_3

[6] Q. Z. Sheng, B. Benatallah, M. Dumas, and E. O.-Y. Mak, "Self-serv: a platform for rapid composition of web services in a peer-to-peer environment," in *Proceedings of the 28th inter- national conference on Very Large Data Bases*. VLDB Endowment, 2002, pp. 1051–1054. http://dx.doi.org/10.1016/b978-155860869-6/50106-2

[7] S. R. Ponnekanti and A. Fox, "Sword: A developer toolkit for web service composition," in *Proceedings of the 11th International WWW Conference*, 2002.

[8] K. Fujii and T. Suda, "Semantics-based context-aware dynamic service composition," *ACM Trans. on Autonomous and Adaptive Systems*, vol. 4, no. 2, pp. 1–31, 2009. http://dx.doi.org/10.1145/1516533.1516536

[9] D.Wu, B.Parsia, E.Sirin, J.Hendler, D.Nau, and D.Nau,"Automatingdaml-swebservices composition using shop2," in *Proceedings of 2nd International Semantic Web Conference*, 2003.

[10] J. L. Ambite and M. Weathers, "Automatic composition of aggregation workflows for transportation modeling," in *Proceedings of the 2005 national conference on Digital government research*. Digital Government Society of North America, 2005, pp. 41–49.