UDC 621.377

AN ENVIRONMENT FOR EDUCATIONAL SERVICE COMMUNITIES

Josef Spillner

Technische Universität Dresden, Germany

In most global economies, there is a strong trend from agriculture and manufacturing towards service-orientation and tertiarisation: Services, products with value-added service solutions and, more recently, automated Internet service offerings seamlessly delivered through on-demand elastic cloud computing resources. In the affected societies, education is recognised to be a key factor for maintaining the competitiveness. Specialised education about services is broadly available, but tool support for hands-on learning and testing of how services can be produced, offered, delivered and improved is missing. We aim to close this gap between theory and application by proposing an integrated environment for educational service communities such as service engineering classes. The initial results of our work show that the environment, which supports both auto-didactic learning and team-based competitive and collaborative learning-by-doing throughout the service lifecycle, motivates students and increases their practical knowledge about services.

Introduction

Two lines of development are pushing for a new paradigm of globally tradeable services: One the one hand, economic tendencies point towards the bundling of products and services into integrated solutions [1], on the other hand, advanced techniques emerge for the creation, description, composition and delivery of automated service offerings such as web services [2]. Such services are completely self-descriptive and therefore interoperably tradeable on different marketplaces, while the provider retains the souvereignty about the descriptions, usage agreement templates and other service artefacts. These developments supports the migration of services to more suitable hosting and cloud environments and the customisation and bundling of services towards value-added solutions, including seamlessly woven manual and automated service executions. To realise this vision, a profound education of service engineers and service providers is essential. Looking from the angle of technically executable services in service-oriented architectures, this requirement is narrowed down to web service engineers who understand the whole service lifecycle. Only through the availability of vast high-quality offers, the critical mass of tradeable services to turn the vision broadly into reality can be reached.

We contribute to this techno-economic transformation by proposing, to our knowledge, the first integrated environment for educational service communities. It combines e-learning techniques with state-of-the-art service platforms and frameworks. The environment guides future web service engineers through the processes of collaborative development, testing, provisioning, search, preparation, usage and rating of services. To complete the service lifecycle, the environment can be connected with the work and development environments on the client computers of students.

There are, of course, alternative e-learning approaches to software and service engineering available, but none of them provides an integrated learning-by-doing experience [3]. An established system for educational software engineering tasks is Praktomat [4]. We have borrowed ideas from it, however, its focus on object-oriented development positions it as rather complementary to service engineering. Furthermore, it is limited to Java artefacts, which according to our experience is unsuitable for the language and framework curiosity of future service developers. A recent system for collaboration and e-learning is CloudIA [5]. It fits well into modern XaaS software stacks and is therefore suitable for being operated in intranets or private clouds. Our proposed interactive learning environment offers similar characteristics, but is optimised for hands-on service learning, development and testing along a well-defined lifecycle.

The article is structured as follows: First, we derive requirements on the amount of information technology support for the environment itself and for each phase of the lifecycle of services. Then, we design an appropriate environment with learning and testing capabilities according to these requirements. Finally, we discuss our experience with the actual use of the environment in the context of a university course about web services and conclude the article with suggestions for future work.

Requirements Analysis

The design and realisation of an environment for educational service communities is primarily driven by the needs of the students and the investable efforts of the teachers on the requirements side, and the technical capabilities of available implementation components on the offers side.

We have determined the following six generic requirements which apply to any domain-agnostic learning-by-doing environments:

1. Multi-instantiation and Multi-tenancy: The environment should be easy to set up for both public and private use. Each installation should support multiple virtual environment, for instance targeting parallel classes of students or periodically repeated courses.
2. Guided collaboration: The environment should support teamwork, selective cooperation among team members and between teams, communication along the cooperation axes and with the teachers, as well as competitive elements such as result ratings and incentives with virtual currencies. Collaborative tools should cover all phases of the service lifecycle, including the design and development of description and implementation artefacts, and the provisioning on marketplaces.
3. Transparent Safety: Students should be given the possibility to make mistakes in a safe sandboxed environment and learn from them. Consistent versioning of contributed artefacts with the possibility to backtrack to a previous version, isolated testbeds for the results and intuitive status and progress information should therefore be supported by the environment.
4. Alignment: The environment functions should relate to the tasks which depend on the learning progress according to the course contents.
5. Automation: In order to reduce the efforts by the teachers, the degree of automation should be as high as possible. This includes checks for obvious mistakes and success notifications, as well as support material and documentation for auto-didactic learning by doing.
6. Integration: The learning environment should integrate into the students' typical digital client environments such as desktops, applications and smartphones to minimise barriers.

Further requirements result from the domain-dependent focus on service engineering and brokerage, following the research and teaching vocabulary of Service-oriented Architectures (SOA), Internet of Services (IoS) for the vision of tradeable services and more dynamic SOAs, Everything-as-a-Service (XaaS) popular with cloud computing researchers, and service science for the relevant intersections with business procedures. According to the IoS notion [6], we distinguish four main phases in the service lifecycle, each with their own phase-specific requirements:

1. Creation: Students need tools to develop generic service descriptions (for functional and non-functional aspects such as pricing and location), web service implementations, and service interaction frontends (clients). Furthermore, they need tools to package, test and offer the services to the appropriate service community in the environment. Finally, resulting from the feedback phase described below, they need tools to update and maintain their contributions to be able to react on feedback from the community.
2. Preparation: Students need a way to test and learn from services of their peers. Therefore, they need ways to find and select services, negotiate usage terms and find suitable interaction frontends.
3. Usage: Students and teachers need to be able to use the selected services easily, without any installation effort, through an assistive service platform.
4. Feedback: After using any service, students need to be able to comment on and to criticise what they have experienced. Likewise, they need to gather the comments from other students and from teachers.

The six generic requirements and the four domain-specific requirements deeply influence the design and realisation of the environment.

Environment Design

Following the rough categorisation of the generic and phase-specific requirements on environments for educational service communities, we structure and subdivide the design of the environment into its software architecture (automation, integration, transparent safety, multi-instantiation, multi-tenancy), the workflows associated with the service lifecycle, and educational concerns targeting students as future producers and consumers of services (alignment, guided collaboration).

Architecture Concept

We propose to connect the ubiquitous communication structures of digital social networks with the technical characteristics of service packages according to the Tradeable Services Model [7]. This means that the executable and descriptive artefacts of a service are rep-

resented as commercial service portfolio of an enterprise in a social business network, and as free service portfolio of a group in a community-focused social network. Service frontends (clients) and other applications logically belonging to a service are, in analogy, offered as downloads and represented as products in the network. Fig. 1 shows this mapping. It generalises enterprises and groups into virtual organisations with service offerings being mapped to tradeable web services. A social service network is created this way, which is applicable beyond educational service communities, for instance for service science experiments [8]. The specialisation on educational social service communities happens by limiting the users to students and teaching staff and the addition of learning support tools which will be described in the next section.

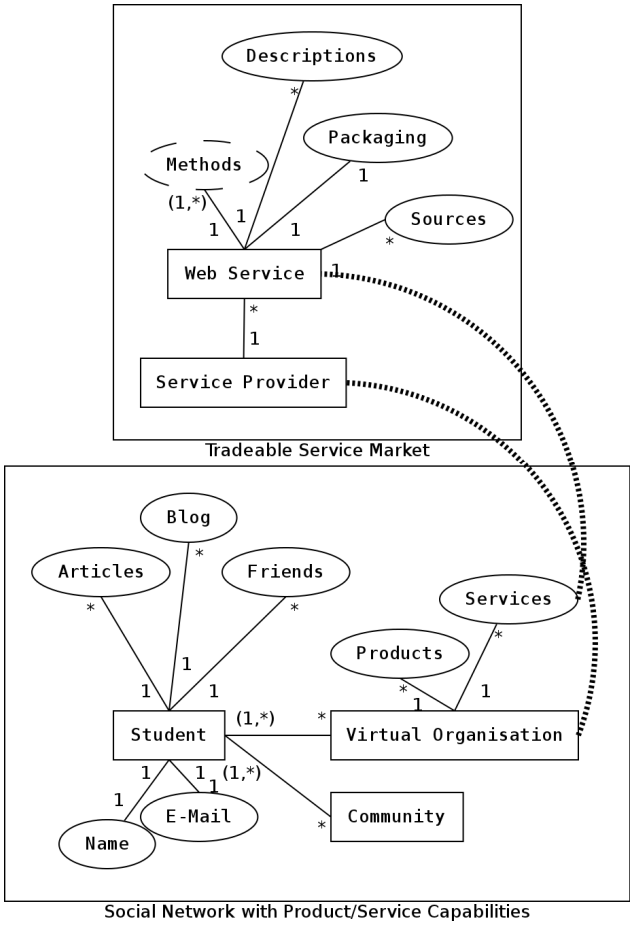

Fig. 1. Mapping of service structures (entities and relations) onto social network content structures (following [7]).

Social service networks encourage students to navigate from the very beginning in a software landscape to which they will eventually contribute by themselves with their service offers. Various service technologies can be tried by accessing the web service interfaces offered by the social network and the service platform handling the tradeable services. By carefully choosing software components for its implementation, the environment is conceived to operate in autarky as self-sufficient system. Thus, it can be used in universities, business training and continuous education institutes as dedicated instances.

On the functional side, the environment shall consist of multiple distributable platform services, each with its proper web service interface.

Workflow Concept

As discussed before, environments for educational service communities should be aligned with course tasks which are attuned to one another corresponding to the lifecycle of services. Our concept relates this requirement to workflows within the environment. We distinguish between three workflows for the service creation phase, namely development, testing and provisioning, two workflows for the service preparation phase, namely search and configuration, and one workflow for each of the remaining phases of usage and feedback.

Service development workflow: Students develop their service artefacts through a version control system which serves as both a backup facility and a way to revert to previous versions of artefacts in the case of detected problems. Artefacts include web service implementation source code, service description files, data models and associated service frontends. Artefact-specific tooling is provided, for instance, Java and WSDL editors. Students also review their software through code review tools. This strengthens the collaborative approach and reduces the lifetime of manifested mistakes. The development of features is driven by a set of tasks defined by the teachers. Hence, students have an incentive to fulfil the functional and non-functional requirements as described by the tasks.

Service testing workflow: In order to test a service within the environment, apart from local testing on the students' development computers, each virtual service-offering organisation receives an isolated and volatile sandbox within the environment. This workflow is otherwise identical to the service provisioning.

Service provisioning workflow: Once students are confident that their service is reasonably complete to be evaluated and tested, they bundle it into a service package and submit it to the repository of the environment. This workflow follows the Internet of Services toolchain approach [9]. The service platform connected to the social network takes over both the service implementation, which is installed into an appropriate runtime environment, and the descriptive service artefacts, which are installed into a publicly accessible ser-

vice registry. Furthermore, the service is added to the virtual organisation's portfolio, and as such is subject to portfolio management such as updates and deletions. The portfolio is mirrored back to the social network. Each service appears with its most important metadata as an offer in the network and can be found and selected through its user interface.

Service search workflow: Finding a suitable service for a given problem always requires a precise problem specification first. Both the interactive graphical user interface of the social network and the programmatic interface of the service platform are supposed to support specifying the problem through full-text search, category browsing and filters over functional and non-functional service properties. Once a service or set of services has been found, the result set is marked for selection and the configuration of the services starts.

Service configuration workflow: Before a service can be used, a configuration is required in many cases. This includes the negotiation of service level agreements, the definition of account data and service-specific parameters, and, depending on the technology, a configuration or allocation run inside the runtime environment of the service. For example, a photo storage service hosted in the cloud will need a certain amount of storage resources according to the maximum amount of photos the consumer has configured.

Service usage workflow: The outcome of the service configuration is a permission to use the service, and a set of choices on how to use the service. These include a web service endpoint for the programmatic invocation, for instance as part of a service composition, a link to dynamic user interface generators, and custom frontends suitable for the consumer's client environment. Each service invocation takes place through a proxy running at the service platform. This way, the agreements can be checked and the virtual currency accounts of the invoking student and the invoked student organisation can be adjusted with the service invocation price. Furthermore, the service execution can be monitored to find faults in the usage agreement or the templates which led to it during the negotiation. A so-called cross-testing can be introduced for educational reasons in which student teams of functionally equivalent services test the respective other services and rate them based on compatibility and other experience.

Service feedback workflow: After having used a service, a student rates it on a linear scale. The service platform ensures that the average rating is represented as a non-functional service property so that suboptimal services can automatically be excluded during the successive searches. Furthermore, bad ratings can also influence the reengineering of services in an iterative development setting.

Educational Concept

The educational aspects are introduced into the environment through the communication channels of the social network and the task management with partially automated solution checks.

Educational communication functions of social networks: Beside the service trading capabilities, the environment offers all communication channels known from typical social networks. The web interface offers various ways for creating personal, organisational or community-bound articles and blogs. Furthermore, galleries can be created, for instance to show detailed screenshots of realised service frontends. In addition, forums can be created for the knowledge exchange among participating students. Beside these web-based channels, real-time channels such as instant messaging for direct support queries to other students and teachers can be integrated.

Partially automated solution checks: The manual check of all delivered student solutions is a time-consuming effort. Web services need to be checked for functional correctness, robustness, matching service description, conforming packaging and overall tradeability. Therefore, a technique for automating at least some of these teacher duties is sought. The environment therefore offers at least rule-based completeness checks over the submitted service packages, and integrates existing web-based tools such as WS-Interoperability checks over the WSDL service descriptions.

Experience Discussion

The realisation of an environment for educational service communities depends heavily on the technologies en-vogue at the time of the realisation. We introduce Servomat, a mixture of both custom and slightly customised off-the-shelf software frameworks, as versatile and yet easily installable and maintainable environment, to validate our approach. Furthermore, we discuss our experience with using Servomat in an actual university service engineering course.

Servomat, a social environment for educational service communities

Servomat is a learning-by-doing environment for all matters of the service lifecycle. It lets its users offer. select and execute services. Servomat is designed to fulfil most of the requirements on information technology support for service communities. Its primary user interface consists of a typical web-based social network, for which the Noosfero framework [8] has been

chosen due to its pre-existing support for product offers in virtual organisations and due to its rich customisation and communication features, including instant messaging through Jabber. Several extensions result from the requirements, including a virtual currency account for each student and a centralised account management attached to the university information system (jExam) which synchronises login information between the heterogeneous components.

The service lifecycle is handled by the Service Platform Architecture for Contracting and Execution (SPACE) [10], an open-source, modular service middleware with brokering and unified hosting capabilities. The SPACE platform services are attached to the appropriate actions in the social network. The provider profile modification and service offers happen through the Provider Wizard platform service, tradeable service and product search through the ConQo discovery platform service, and the billing of service invocations through the Access Gate proxy platform service which is in turn integrated with the ServBank component. With optimisations such as incremental paginated web service synchronisation of the databases between SPACE and Noosfero applied to reduce the on-demand access waiting times [11], the integrated architecture shown in Fig. 2 is derived.

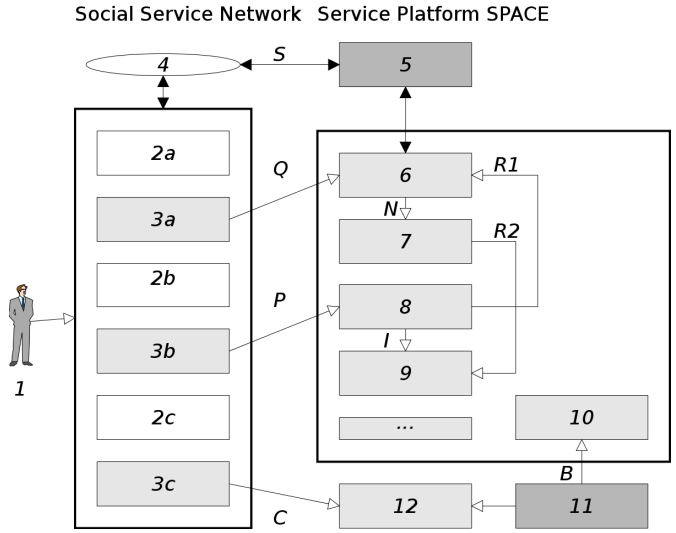

Fig. 2. Integration between the service platform SPACE and the social network framework Noosfero, representing a concrete Social Service Network. Users (1) interact with existing social network functions like people (2a), communities (2b) and content (2c). These are blended with platform features like services (3a), own service offers (3b) and finances (3c). The service database (4) is synchronized (S) with an update tool (5) from the platform's registry (6) which can be queried (Q). Contracts are negotiated (N) in a negotiation wizard (7) based on SLA templates registered by a provisioning tool (8). This tool also installs (I) executable artefacts into a runtime container (9). Finally, a service invocation proxy (10) checks the caller's contract-based authorization and a transaction tool (11) bills (B) the user's financial account (12) accordingly.

For the collaborative development, the version control system Subversion has been chosen in combination with the web-based ReviewBoard software. Finally, tasks and checks are implemented as a custom software tool. The overall environment architecture can be seen in Fig. 3. It highlights the direct involvement of students through SPACEflight [10], a virtualisable live demonstrator of the SPACE service platform and, among other tools, an Internet of Service service engineering and provisioning toolchain, as alternative to manual installation of tools on conventional student desktops [9].

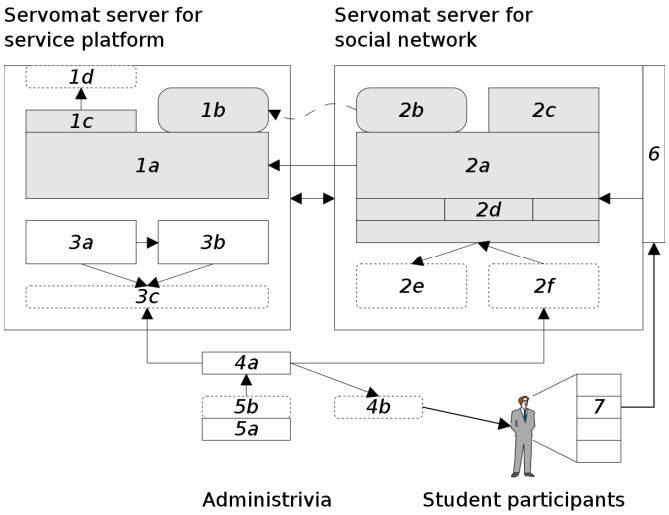

Fig. 3. Architecture of the Servomat environment, consisting of a service platform (1a) with database (1b) and hosting containers (1c), a Social Service Network (2a) with its own database (2b), network engineering tools, communication services (2c) and attached student clients (7). Student accounts and repositories (2f, 3c) are created by a tool (4a) by querying a central student registry (5) and notifying the students by e-mail (4b). Task and review tools (3a, 3b) operate directly on the repositories. Students generate content interactively in the social network (2e) or programmatically by providing service packages (1d). The Social Service Network (2a) contains all the extensions from Fig. 2 as well as an API and a custom theme and site-specific configuration. A firewall and proxy subsystem (6) ensures that only authorized students can participate.

The Servomat environment is itself service-oriented. The SPACE platform services, the Noosfero API and other remote interfaces allow students to tinker not just with their peers' services, but also with the environment itself in order to gain experience and knowledge on the advantages and disadvantages of different web service

protocols, description formats and tool support.A detailed manual containing the Servomat background, functionality description, installation instructions and operation guidelines in German language is available for interested parties [12] in order to increase the likeliyhood of adoption by other course or lab exercise designers.

Experiences from using Servomat

The Servomat software has been developed and experimentally used for two consecutive semesters in a lecture on the design of distributed systems based on service-oriented architectures at Technische Universität Dresden in Germany. The environment runs at `http://servomat.inf.tu-dresden.de`. The first time, domain-specific flight booking services were created, whereas the second time this limit was lifted. Both in the lecture and in other experimental setups with other user groups, a number of user surveys were performed to get information about the acceptance, the suitability and the general impression of such supportive environments. The majority of students finds Servomat helpful and makes use of the provided documentation and communication channels. The choice of letting students navigate in a social network considerably lowers the barrier between their leisure time activities and study-related obligations.

In one particular survey, 44 students were invited to enter into the service lifecycle with the search workflow. 13 students immediately managed to find and configure the service. Of these, 5 students then transitioned through the usage phase by invoking the service successfully. Nevertheless, these results suggest future improvements.

Conclusions and Future Work

The article has motivated the need of introducing assistive environments for educational service communities in order to advance the techno-economic transformation of societies. Both generic and service lifecycle-specific requirements have been extracted and used for the proposal of an architecture of a social and collaborative environment and seven distinct workflows therein. Furthermore, the proposals were validated with an environment implementation and its usage in a university course setting. In order to maintain the alignment of the learning-by-doing effect of the environment, future work will have to introduce new technologies such as dynamic platform-as-a-service coupling for the service hosting. To increase the digital inclusion of students, a global environment acting as distributed social networking aggregator for multiple environment installations would be useful.

References

1. S. Montresor, G. V. Marzetti: The deindustrialisation/tertiarisation hypothesis reconsidered: a subsystem application to the OECD7. Cambridge Journal of Economics, Oxford University Press, vol. 35(2), pp. 401-421, 2011.

2. A. Barros, M. Allgaier, A. Charfi, M. Heller, U. Kylau, B. Schmeling, M. Stollberg: Diversified Service Provisioning in Global Business Networks. Proceedings of the SRII Global Conference, pp. 716-728, April 2011.

3. F. Chesani, A. Ciampolini, P. Mello: E-Learning by Doing with Computational Logic. Proceedings of Knowledge Construction in E-Learning Context, Cesena, Italy, September 1-2, 2008.

4. A. Zeller: Making Students Read and Review Code. Proceedings of the 5th ACM SIGCSE/SIGCUE Annual Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland, July 11-13, 2000.

5. F. Doelitzscher, A. Sulistio, C. Reich, H. Kuijs, D. Wolf: Private cloud for collaboration and e-Learning services:from IaaS to SaaS. Computing vol. 91, pp. 23-42, 2011.

6. A. Kabzeva, M. Hillenbrand, P. Müller, R. Steinmetz: Towards an Architecture for the Internet of Services. Proceedings of the 35th EUROMICRO Conference on Software Engineering and Advanced Applications, Greece, 2009.

7. J. Spillner: Methodik und Referenzarchitektur zur inkrementellen Verbesserung der Metaqualität einer vertragsgebundenen, heterogenen und verteilten Dienstausführung. Dissertation, Technische Universität Dresden, September 2010.

8. J. Spillner, A. Caceres, B. Buder, R. Kursawe, L.S. Globa, A. Schill: Extending Social Networks with Service Delivery Capabilities for User-Centric Service Trading. In Proceedings of the 10th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), Lviv-Slavske, Ukraine, February 2010.

9. J. Spillner, A. Kümpel, S. Uhlig, I. Braun, A. Schill: An Integrated Provisioning Toolchain for the Internet of Services. 10th IADIS International Conference WWW/Internet, Rio de Janeiro, Brazil, November 2011.

10. J. Spillner: SPACEflight - A Versatile Live Demonstrator and Teaching System for Advanced Service-Oriented Technologies. In Proceedings of the 21st International Crimean Conference on Microwave and Telecommunication Technology (CriMiCo/КрыМиКо), vol. 1, p. 455-456, Sevastopol, Crimea - Ukraine, September 2011.

11. T. Muckwar: Optimierung der Skalierbarkeit einer Dienstplattform durch Einsatz performanter Web-Service-Techniken. Diploma thesis, Technische Universität Dresden, August 2011.

12. J. Spillner: Servomat - Soziale Dienstleistungsnetzwerke für angehende Web-Service-Ingenieure und -Anbieter. Manual, online: `http://servomat.inf.tu-dresden.de/`, November 2011.

Received in final form May 10, 2012