Magoutis *et al. Journal of Internet Services and Applications* (2015) 6:19
DOI 10.1186/s13174-015-0033-5

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

Design and implementation of a social networking platform for cloud deployment specialists

Kostas Magoutis[1,2*], Christos Papoulas[1,3], Antonis Papaioannou[1,3], Flora Karniavoura[1,3], Dimitrios-Georgios Akestoridis[2], Nikos Parotsidis[2], Maria Korozi[1,3], Asterios Leonidis[1,3], Stavroula Ntoa[1] and Constantine Stephanidis[1,3]

Abstract

A new discipline at the intersection of the development and operation of software systems known as *DevOps* has seen significant growth recently. Among the wide range of tasks of DevOps professionals, we focus on that of selecting appropriate cloud deployments for distributed applications. Despite the advent of automated software deployment and management frameworks, reasoning about good deployments still requires interaction with experts, often through discussions on online technical forums and social networks.

Current social networking technologies offer basic ways to communicate. Within the *DevOps* community, communication on application structure and cloud deployment tradeoffs could become more effective by using knowledge present in global community-sourced information repositories. In this paper we argue for the benefits of tapping into such knowledge and for seamlessly feeding it back into the social networking platform.

The social networking platform presented in this paper integrates social networking with automated deployment of applications on multi-clouds and with knowledge drawn from community-sourced information repositories. The implementation leverages two such repositories, the PaaSage repository and Chef Supermarket. Our user evaluation experiments demonstrate the value created for *DevOps* professionals.

1 Introduction

In an information technology (IT) world that requires shorter development cycles, excellent software reliability and delivery, and a service-oriented perspective on all aspects of software, the fields of software development and IT operations are drawn closer together in a new field called *DevOps* [1] (short for "development and operations"). DevOps encompasses a rapidly growing class of IT professionals whose interests span the fields of software development and deployment, infrastructure management, system administration, cloud computing operations, and IT optimization.

In recent years there have been several efforts to provide DevOps professionals with the tools that they need to address challenges in developing, deploying, and managing large-scale applications. To bridge across different development and deployment environments, especially in the cloud computing space, configuration management systems such as Chef [2] and Puppet [3] have emerged as solutions to codifying and executing management procedures (installation, deployment, etc.) around software components. DevOps tools aiming to simplify application management increasingly express application structure, requirements, and application deployments in a cross-platform manner using models[1]. TOSCA [4] and CloudML [5] are two such frameworks with associated runtimes [6, 7].

Increased adoption of these frameworks by DevOps professionals has spawned active communities of users

*Correspondence: magoutis@ics.forth.gr
[1]Institute of Computer Science (ICS), Foundation for Research and Technology – Hellas (FORTH), GR-70013 Heraklion, Greece
[2]Department of Computer Science and Engineering, University of Ioannina, GR-45100 Ioannina, Greece
Full list of author information is available at the end of the article

with an interest in exchanging know-how and in better understanding the DevOps field. A recent trend in DevOps communities is the creation of *repositories of information* where users can jointly contribute knowledge –via *crowdsourcing* [8]– creating shared value. These include repositories of software, such as GitHub [9], Sourceforge [10], GoogleCode [11], CodePlex [12], and repositories of software component descriptions and associated configuration and management procedures, such as Chef Supermarket [13].

Among the range of possible DevOps tasks, in this paper we focus on selecting the most appropriate deployment configuration for an application. This task is especially challenging in a *multi-cloud* setting due to the large diversity of deployment possibilities and tradeoffs. It is of particular interest to *cloud deployment specialists*, DevOps engineers whose main role is to perform distributed software deployment to various testing and production environments. Since DevOps engineers usually form well-integrated teams, we believe that the theme of this paper is more broadly relevant to the entire DevOps community.

Currently DevOps engineers work with a small set of well-understood deployment options, missing opportunities for improving performance, reliability and/or lower cost. Investigating new options involves time-consuming testing over new infrastructures. Discussing with the community in online social or technical forums may provide insight over deployment options; however the answer to a hard question often needs to be backed by experimental data that is not readily available.

In this paper we argue that the state of things can be improved by providing DevOps engineers with analyses of execution data from a large set of cloud applications over a variety of cloud infrastructures. To store such execution data, we use a new repository of information developed by the PaaSage EU project [14] based on the CAMEL modeling language. In CAMEL, application models are associated with *execution histories* collected from application deployments over different cloud infrastructures. Analysis of data (collected in a crowdsourced manner) can reveal important knowledge about application deployments, such as performance and availability tradeoffs, cost-effectiveness, etc. [15].

In line with the DevOps principles of rapid and continuous deployment, we require that CAMEL application models be automatically deployable. We achieve this by drawing Chef management procedures for each software component of the CAMEL application model from Chef Supermarket and orchestrating them to achieve full application deployment. The use of Chef Supermarket adds important new value: through analysis of information mined from Chef Supermarket on components and their interrelationships, we can aid DevOps engineers in understanding and adapting the structure of their applications and in answering questions such as *"which relational databases are alternatives and can replace MySQL?"* or *"what other components MySQL depends on?"*.

Motivated by the fact that information sources such as the CAMEL repository and Chef Supermarket contain important knowledge that can aid DevOps engineers but is currently unexploited, we embarked on a project to bring this knowledge closer to them. Current general-purpose forums, message boards, or generic social networks however support only basic social interaction between professionals. We thus decided to design and prototype a social networking platform that leverages these information repositories.

The social networking platform (developed in the context of the PaaSage project and named after it) enriches user interactions with structured references to applications and their components, and mined information from execution data of real deployments. Mined knowledge is combined with user activity and profiles to provide personalized suggestions and hints. The main research question addressed in this paper is whether DevOps professionals perceive value in a social networking platform designed to leverage the aforementioned information repositories and thus whether they have incentives to use it. We address this question through three user evaluation studies.

The architecture of the PaaSage social networking platform is shown in Fig. 1, highlighting the integration of a collaborative social platform (left) with the repositories of information (right). It shares certain principles with standard question-and-answer (Q&A) sites where users can create sub-communities relative to specific topics of interest (groups) and use features such as following other users or news feeds, facilitating stronger interaction between users.

The PaaSage social networking platform advances the state of the art in several ways. First, it encompasses the ability to represent applications and infrastructure as *models*, and to use them in automating deployment on cloud platforms. Second, it incorporates two crowdsourced repositories: a repository of application models, including software components and configuration properties, along with a collection of their execution histories (denoted "CAMEL repository" in Fig. 1); and a repository of management processes of individual software components (denoted "Chef repository" [13] in Fig. 1). Another distinctive feature of our system is the mining of information (such as application behavior under different cloud environments in the former repository, and component information and inter-relationships in the latter) and its use in providing users with suggestions and hints while browsing or in Q&A.

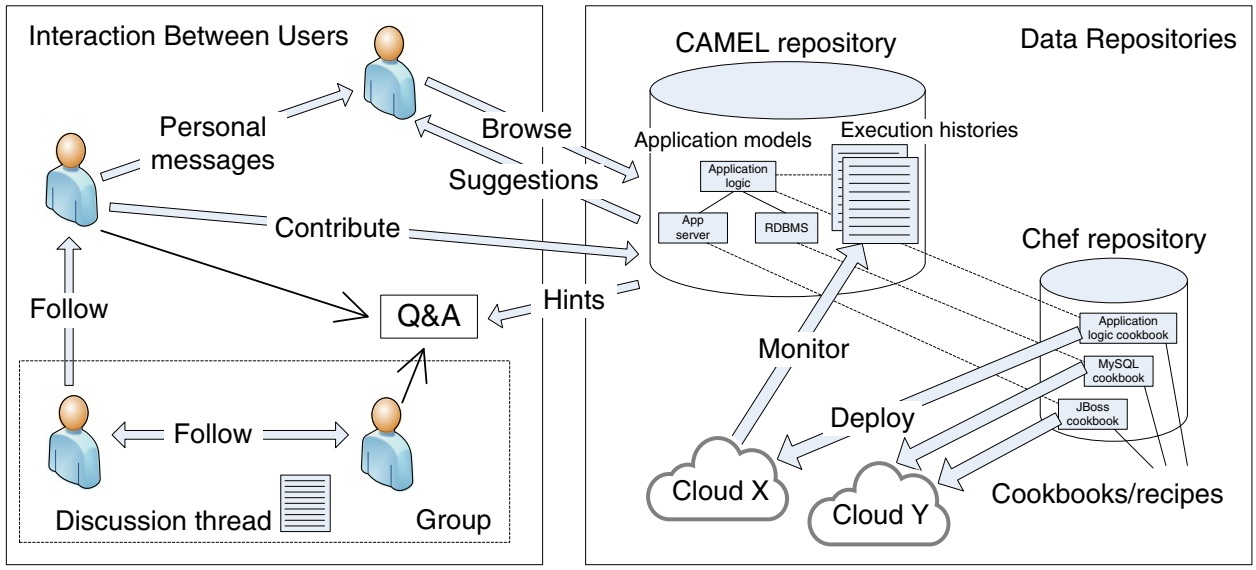

Fig. 1 Integration of a collaborative social platform with repositories of information

Our contributions in this paper are:

- The design and implementation of the PaaSage social networking platform (user interface and back-end infrastructure) for the DevOps community, integrated with two information repositories.
- Use of advanced features such as pointers to structured descriptions of applications, components, executions, etc., and mined knowledge (statistics, cost-benefit analyses) to improve the level of technical discussion between DevOps users
- Three user evaluation studies and a demonstration of offering users with cost-benefit analyses of application deployments on multi-cloud setups.

The paper is structured as follows. In Section 2 we describe related work split into two major areas: professional (social) networks and systems automating configuration management and deployment. In Section 3 we provide background on modeling distributed applications and the Chef configuration management framework. In Section 4 we describe the requirements and design principles of the social networking platform and in Section 5 we describe our implementation. In Section 6 we describe our user evaluations and experience with our prototype (accessible online) and in Section 7 we present our conclusions.

2 Related work

In the following subsection we review related work on professional social networks. In Table 1 we summarize and categorize the characteristics of the most important related approaches along the following dimensions (depicted as columns of Table 1): (1) which of the following key social features are supported by the platform: follow users; news feeds; groups; Q&A; personal messages; (2) does the platform rely on one or more repositories to store the following type of information: software code, software models, configuration information, execution histories; and whether these repositories are community-sourced; (3) does the social networking platform leverage the repositories to provide users with specific suggestions and hints; (4) does it support application deployment? In Table 1 we compare related approaches to ours along these dimensions and provide further details in the following section.

2.1 Professional networks

During the last few years social networking has evolved into a fundamental daily activity for many individuals and a new frontier for business marketing. Besides "traditional" social networking, a recent trend is professional and domain-specific networking services focused on interactions and relationships of a business nature around a specific target domain. These online communities have the potential to become a platform for collective intelligence and open innovation [16], a medium for knowledge-centered collaboration [17], and a trustworthy decision-support tool [18].

IT professionals use a variety of online sources as aids in their daily tasks. Developers typically prefer community-moderated forums over vendor-moderated sites [19]. Professional networks focusing on software technology in particular provide developers with the opportunity to leverage the knowledge and expertise of their peers. One of the most popular such platforms is GitHub [9],

Table 1 Feature comparison

	Interaction Between Users				Repository						
	Social features[a]	Groups	Q & A	Personal messaging	Software code	Software models	Software config	Execution histories	Crowd sourced	Repo assisted hints[b]	Application deployment
GitHub	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
Sourceforge	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗
GoogleCode	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗
CodePlex	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗
StackOverflow	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Bluemix	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓
Chef Supermarket	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
LinkedIn	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
PaaSage SN	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓

[a]Features: follow and news feed
[b]User assistance based on data analysis of the repository

a collaborative revision control platform for developers launched in April 2008, and arguably the largest code-hosting site in the world. GitHub provides social networking functionality such as feeds, followers, wikis and a social network graph that captures how developers work on their versions ("forks") of a repository, which version is newest, etc. Gitter [20] is a related service that facilitates discussions between members of GitHub communities by providing a long-term chat integrated with code and issues. Neither GitHub nor Gitter collect, analyze, or use information from executions of application deployments to improve the level of technical discussion between users.

Sourceforge [10] was the first code-hosting platform offered to open-source projects. It was launched in 1999 and offered IT professionals the ability to develop, download, review, and publish open-source software. Sourceforge is similar to GitHub in its support for social features. In 2006 Google offered publicly their own hosting platform, Google Code [11], which is planned to shut down by January 2016 as "*the service simply isn't needed anymore*" [21]. In June 2006 Microsoft officially released their own open source project hosting website, CodePlex [12]. Similar to GitHub it allows shared development of open source projects. CodePlex features include wiki pages, source control systems (such as git, mercurial, etc.), discussion forums, issue tracking, project tagging, RSS support, statistics, and releases. Microsoft is currently moving many of its premier projects from CodePlex to GitHub [22]. None of these platforms abstract code structure through modeling or enhance user interactions through the use of analytics over application execution histories.

StackOverflow [23] advances on earlier Q&A sites in which users ask and answer questions. Users can vote up or down questions and answers and earn *reputation points* and *badges* in return for their active participation. Although StackOverflow and GitHub address different aspects of software development (StackOverflow is not a code-hosting platform) there is a synergy and correlation between the two [24]. Our system extends StackOverflow through the use of social networking features that enable users interested in reasoning about application deployments to use and share knowledge drawn from analyses of information repositories.

IBM's Bluemix [25, 26] is a development and support platform for communities of DevOps users wishing to compose distributed applications out of components drawn from libraries and deploy them at IBM-provided and supported cloud infrastructure. Bluemix is a key component of IBM's DevOps best practices [27] for achieving rapid prototyping, automated deployment, and continuous testing of software. Bluemix relies on StackOverflow to support community discussions. It uses the StackOverflow API [28] to search and retrieve Q&A threads and display them within the Bluemix platform [26]. Our system differs from Bluemix in its support for expressing applications as models (CloudML, CAMEL), its use of two information repositories, the PaaSage repository of models and execution histories and Chef supermarket, and the use of analytics over past executions to enable users to reason about application deployments. A common feature between our system and Bluemix (shared by no other related system in Table 1) is support for deployment of distributed applications, although we use a different deployment mechanism (Section 5.2). This is a key feature required by DevOps users [27] and thus included it in our professional networking platform.

Perhaps the best known social networking platform for professionals is LinkedIn. Widely adopted across a range of professional communities due to its robust set of social features (and to some extent due to its use of extensive analytics over collected information [29]),

LinkedIn provides no specific support for software engineering activities and thus more closely resembles traditional social networking platforms such as Facebook.

The above systems can be further classified based on whether they use a repository to store software-related information (code, models, configuration, or execution histories) and whether this information is shared and raised through crowdsourcing [8]. As seen from Table 1, GitHub, GoogleCode, CodePlex, SourceForge, Bluemix, Chef Supermarket (a community site and repository described in Section 3.2), and our platform store at least one type of software-related information and all systems but Bluemix are raising shared content in their software-related repositories via crowdsourcing. Our professional network is the only solution that analyzes information in its software-related repositories to assist users with suggestions and hints.

On the other hand, an important concern when designing an online community portal is to stimulate regular contributions and effective collaborations. The willingness to share knowledge with community members is influenced by social interaction ties, trust, norm of reciprocity, identification, shared vision and shared language [30]. In addition, the willingness of professional community members to continue being engaged with the network was found to be affected by social interaction ties and by their satisfaction in relation to their pre-usage expectations [31]. Following the above principles, as well as literature reported usability and sociability factors to reading, contributing, collaborating and leading in an online community [32] the PaaSage professional network aims to provide high quality services to software engineers, actively engaging them in collaborative and community-building activities.

Next we review related work in the areas of configuration management and application deployment.

2.2 Configuration management and deployment

A key component in a portfolio of DevOps tools is *configuration management* (CM) [33], the process of maintaining a detailed recording of software and hardware components in an infrastructure. An effective CM process provides significant benefits including reduced complexity through abstraction, greater flexibility, faster machine deployment, faster disaster recovery, etc. There are numerous configuration management tools from which a system administrator can choose, however the most widely known are: Bcfg2 [34], CFEngine [35], Chef [2], and Puppet [3]. Each of these tools has its strengths and weaknesses [36, 37]. In a DevOps environment, a CM solution is often combined with provisioning and deployment tooling [27]. In this work, we use Chef as a CM and deployment automation tool to support professional network users.

A recent trend in DevOps software development is *continuous integration* (CI) [38] and automated code deployment and testing off of online code repositories. Travis [39] is a CI tool that automatically detects when a commit has been made and pushed to a GitHub repository, subsequently tries to build the project, deploy and run tests, and notify the user of the status. Another popular CI tool is Jenkins [40], an open-source software tool for testing and reporting on isolated code changes in real time. Similar to Travis, Jenkins, enables developers to find and solve defects in their code rapidly and automates the testing of their builds. Although the PaaSage social networking platform does not provide a complete CI solution, it automates the deployment of complex applications through a model-driven process (C2C) described in Section 5.2.

Previous research on automatically finding optimal deployments of distributed applications on multi-clouds has explored mathematical optimization techniques with main objectives being performance and cost [41–43]. QoS-aware deployment and management of applications on cloud infrastructures using workload characterization and system modeling techniques offer another approach to this problem [44]. While such tools are useful in reducing the range of deployment choices, their simplifying assumptions and inability to capture the full set of user requirements means that they usually provide only approximate solutions to deployment problems. Our work is complementary in that it helps DevOps engineers browse over and analyze results of past deployment executions and communicate and exchange ideas to better understand tradeoffs in the space of deployment solutions.

3 Background

In the following subsections we briefly describe the modeling concepts used in our social networking platform as well as the key concepts behind the Chef configuration management framework. Additionally, we describe the characteristics of the community of users that contribute to the Chef Supermarket repository.

3.1 Application modeling

A variety of approaches have been used in the past to model applications and their deployment characteristics [4, 5]. CloudML [5] is a recent approach that focuses on the provisioning and deployment of multi-cloud applications and also at the center of our modeling activities. The universe of modeled concepts in our work extends beyond CloudML and into the Cloud Application Modeling and Execution Language (CAMEL), a family of domain-specific languages (DSLs) currently under development in the PaaSage EU project [14]. CAMEL DSLs cover a wealth of aspects of specification and execution of multi-cloud applications (Fig. 2). With CloudML at

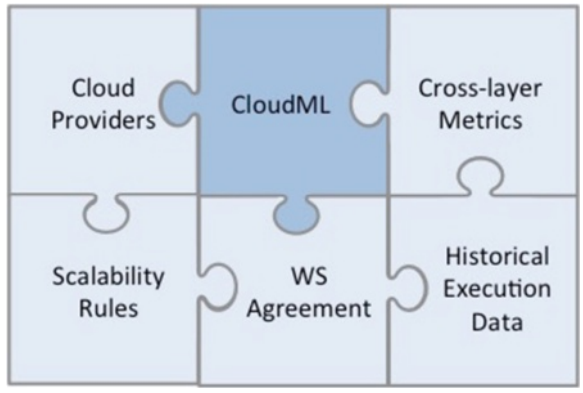

Fig. 2 DSLs comprising CAMEL

the core of application modeling CAMEL (and thus in our professional network) it is worthwhile describing its key modeling concepts:

- *Internal component*: a reusable type of application component, whereas an *internal component instance* represents an instance of an application component. The description of an application component stays at a generic level while the specification of its respective instances involves particular configuration information.
- *Communication*, *Communication Instance*: a relationship between two application components or component instances respectively. This concept is used to describe communication or containment relationship between components.
- *Cloud*: a collection of virtual machines (VMs) offered by a cloud provider.
- *VM type*, *VM instance*: a VM type refers to a generic description of a VM, while an instance of a VM type maps to a specific instantiation of a VM including specific configuration information.

We have implemented a CAMEL information repository using the Eclipse Connected Data Objects (CDO) [45–47] technology. The repository is currently being populated with a wealth of information from multi-cloud deployments of various distributed applications [15]. We are also in the process of building an analytics engine and knowledge base over the CAMEL repository to extract knowledge about deployments characteristics that work best for certain applications and use it in the context of the professional network. While we refer to this knowledge base in the context of our description of the professional network, its implementation is beyond the scope of this paper and subject of future work.

3.2 The Chef configuration management framework

The Chef CM framework automates complex configuration management tasks through an *infrastructure-as-code* approach. Packages, services, and other pieces of a system are represented as *resources*. Configuration files that describe resources and their desired state, called *recipes*, provide operations such as package installment, software configuration, and application deployment. Recipes are stored in *cookbooks*, which also contain other related components including templates, file distributions, and metadata.

One of the key strengths of Chef is its community. The Opscode Community site (or Chef Supermarket) [13] is a site where people from a wide range of backgrounds can contribute and share a plethora of cookbooks that are available to the community. This community seems to evolve and grow over time. As seen from our analysis of contributions to Chef's public repository (Fig. 3) its contents started from an initial state of 72 cookbooks (October 2009) and grew slowly over its first year of operation. After two years, the rate of new cookbooks started to increase. In particular, each month in 2014 about 58 new cookbooks were shared in the Opscode community site. As of June 2015, there are 2303 cookbooks in Chef's public repository and their number is growing at an even higher rate. This highlights Chef's popularity and its emergence as a de-facto standard for systems management and application deployment tasks.

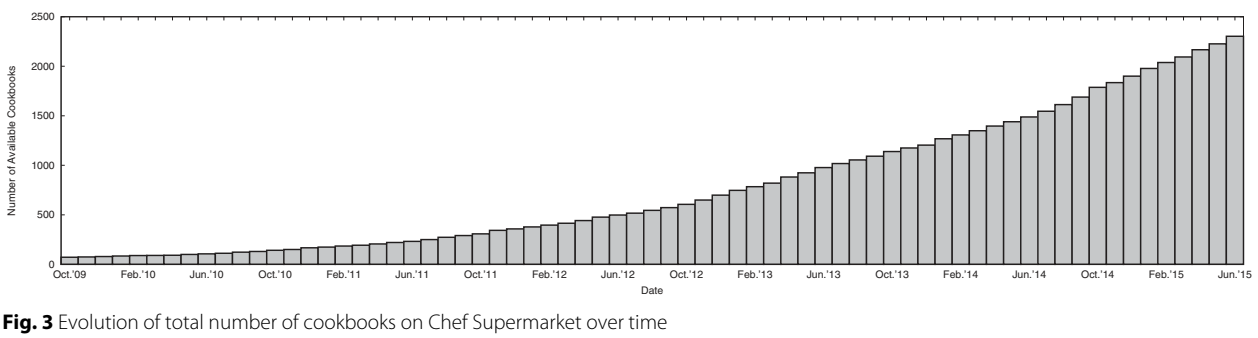

Fig. 3 Evolution of total number of cookbooks on Chef Supermarket over time

4 The PaaSage social networking platform

A key design objective of the professional social networking platform is to create a strong bond between (i) software engineering services for managing and deploying cloud-targeted application models; and (ii) community-oriented facilities for communication and collaboration between users. The interconnections between the two in the design of the user interface are depicted in Fig. 4 and further explained in Section 4.3.

The architecture of the professional networking platform and the supporting infrastructure (including the two information repositories and facilities for application modeling and deployment on multi-cloud infrastructures) are shown in Fig. 1. The requirements that drove the design of the overall platform are summarized in the last row of Table 1 and details on the implementation are provided in Section 5. The prototype implementation is publicly accessible online at http://socialnetwork.paasage.eu.

Users of the professional network are expected to be familiar with basic principles of software modeling and distributed cloud-based application design and deployment. Their roles will range from DevOps engineers and cloud deployment specialists to IT architects, cloud developers, software engineers, application designers, and system administrators. Section 4.1 provides an overview of support around application modeling. Section 4.2 focuses on community-building aspects. Section 4.3 analyzes the process and practices applied to the design of the user interface.

4.1 Application modeling

The PaaSage professional networking platform aims to support users that wish to explore, deploy, and optimize application models and components.

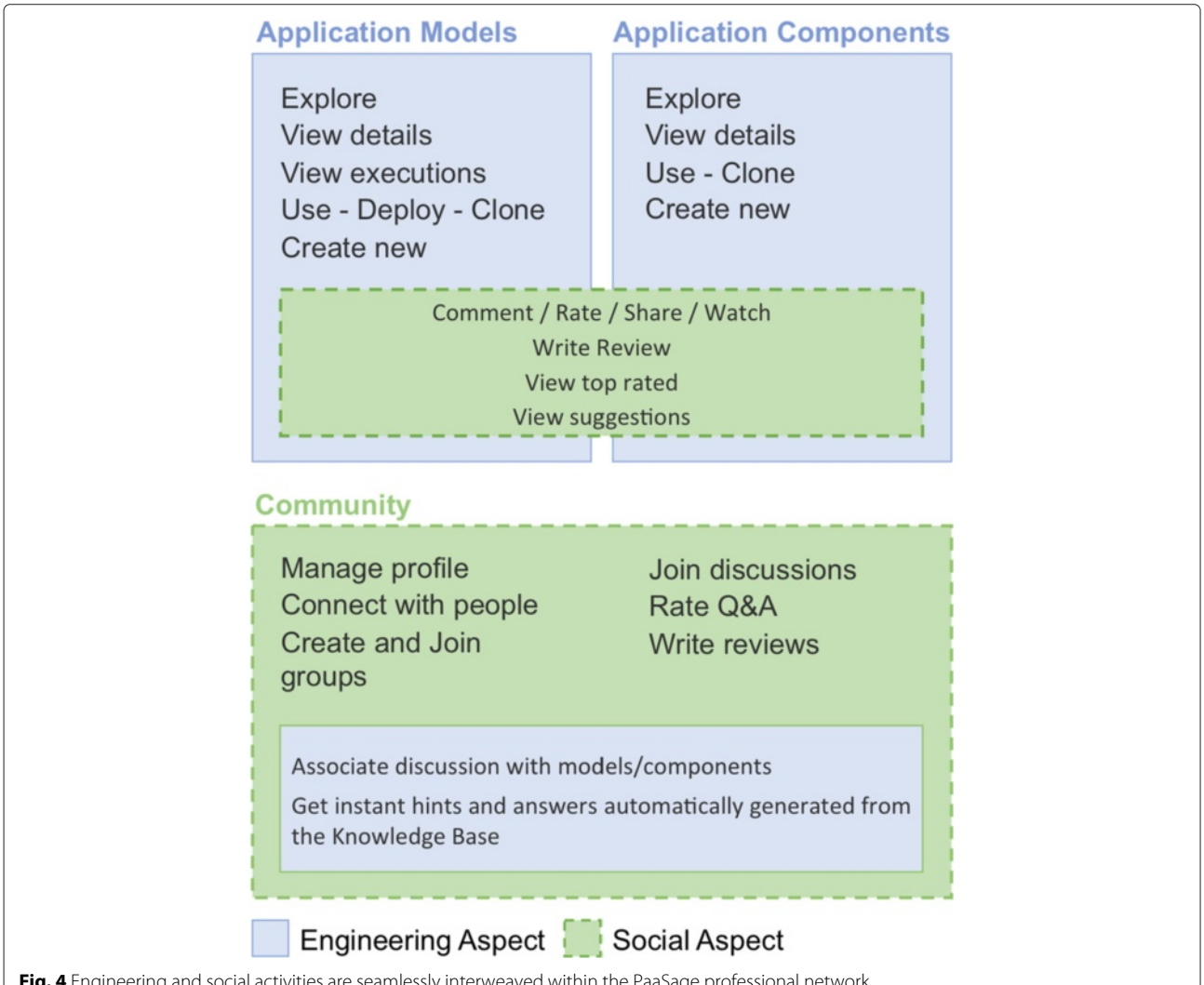

Fig. 4 Engineering and social activities are seamlessly interweaved within the PaaSage professional network

4.1.1 Model exploration

The platform offers personalized and universal exploration facilities to accelerate model discovery. The *models page* depicted in Fig. 5 concentrates information on application models available in the network. The level of personalization on the displayed content is known to strongly influence overall user experience [48]. Based on this principle, our platform delivers (i) personalized recommendations based on each user's areas of interest (Fig. 5a) and (ii) context-sensitive lists of recently-viewed models and components for quicker reference (Fig. 5b).

Among universal features, the most commonly used yet highly efficient one is *content classification based on explicit categories*. Previous studies have shown it to simplify browsing [49] for almost 50 % of targeted users [50, 51]. Content organization is further enhanced with *tag annotations* that can be set in a category-independent manner, motivating a *classification-by-use* basis [52]. To leverage the collective experience of professional network members, we facilitate the discovery of new and noteworthy content by promoting featured, popular and trending application models based on their community impact [53].

To assist users that prefer searching over browsing [50, 51] we provide an advanced filtering mechanism and a faceted-search facility to enable that type of content discovery. Filters are differentiated based on context. Filters on the initial models' category page include model categories and recently-used models. When viewing models under a specific category, filters narrow down results based on: model status, deployment platform, modeling framework, model cost, minimum uptime, maximum response time, minimum throughput, geographical distribution of the model's executions, and specific tags related to the model. When viewing the page of an application model, filters narrow down the displayed executions based on execution start and end date, cost, uptime, response time, throughput, geographical distribution, cloud provider, and execution owner (user, user's network, all users).

4.1.2 Focusing on a model

An application model is a discrete entity with a wealth of heterogenous information that must be delivered in a well-structured manner. Combining this requirement with the fact that the popularity of a model is based on

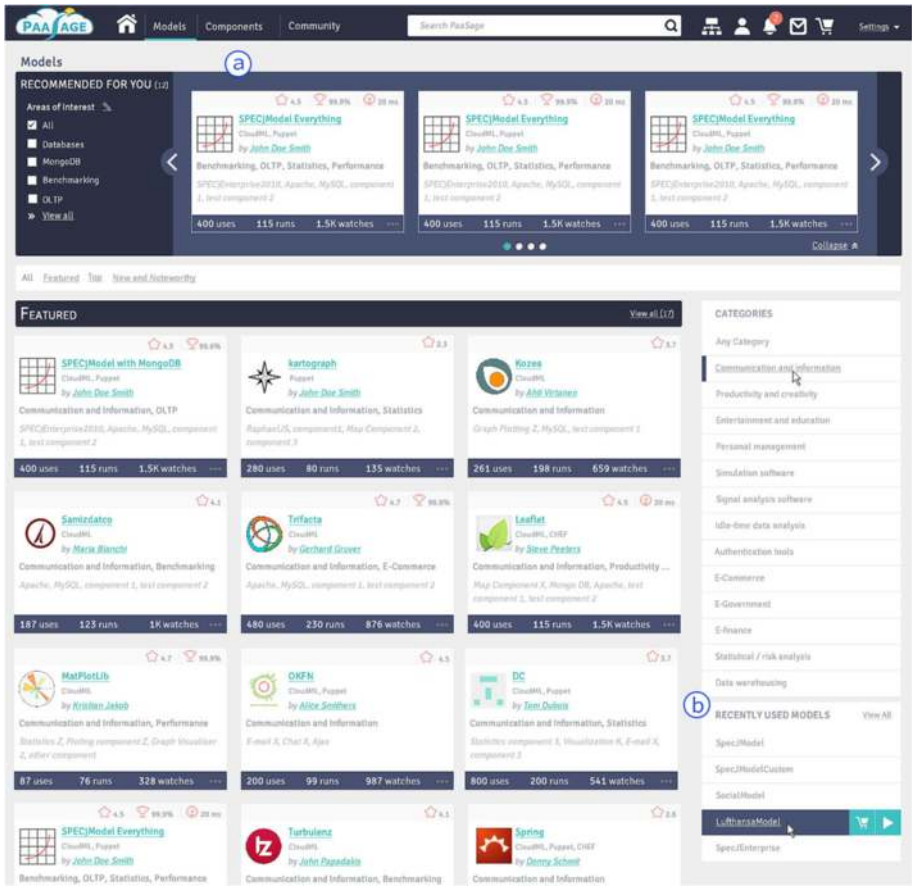

Fig. 5 Personalized models page

efficient broadcasting of its strong features, the PaaSage professional networking platform employs the concept of Pages typically used to promote organizations/businesses in social networks (e.g., Facebook fan Pages, Google+ Pages, LinkedIn Company Pages, etc.) [54].

At the top of an application model Page (Fig. 6a) a user can find essential information regarding that model, including its identity and impact in the community. Apart from the basic description and contributors, a mix of social information (e.g., rating, top review, number of watches and shares) and engineering information (e.g., version, number of deployments and uses) allows the user to form a clear picture about the model's capabilities. Interactive controls permit rapid action from an engineering (e.g., cloning, deploying, use) and social perspective (e.g., watching or sharing a model).

Besides the model overview, a secondary menu (Fig. 6b) allows users to get additional information through separate screens that present: the hierarchical structure of its components; any related discussions and reviews; similar models; and lists with extended execution statistics (runs). The past-executions list stored in the CAMEL information repository offers the ability to compare execution statistics and identify the most successful configurations (i.e., identify best practices). Due to the large amount of information, we use a rich filtering mechanism to narrow down the displayed data, while graphical visualizations facilitate quicker analysis (e.g., uptime, response time, throughput, cost, etc.). The same design approach was used on *component Pages*; the only exception is the fact that the execution statistics screen is missing as such data are not available for individual components.

Although the PaaSage professional network does not provide an integrated model editor, it does provide some support towards creation of application models. Members can import new models or reuse (clone and modify) existing models to benefit from the design principle of service composability. We have applied the *shopping cart metaphor* (Fig. 7) so that users, while browsing, can save

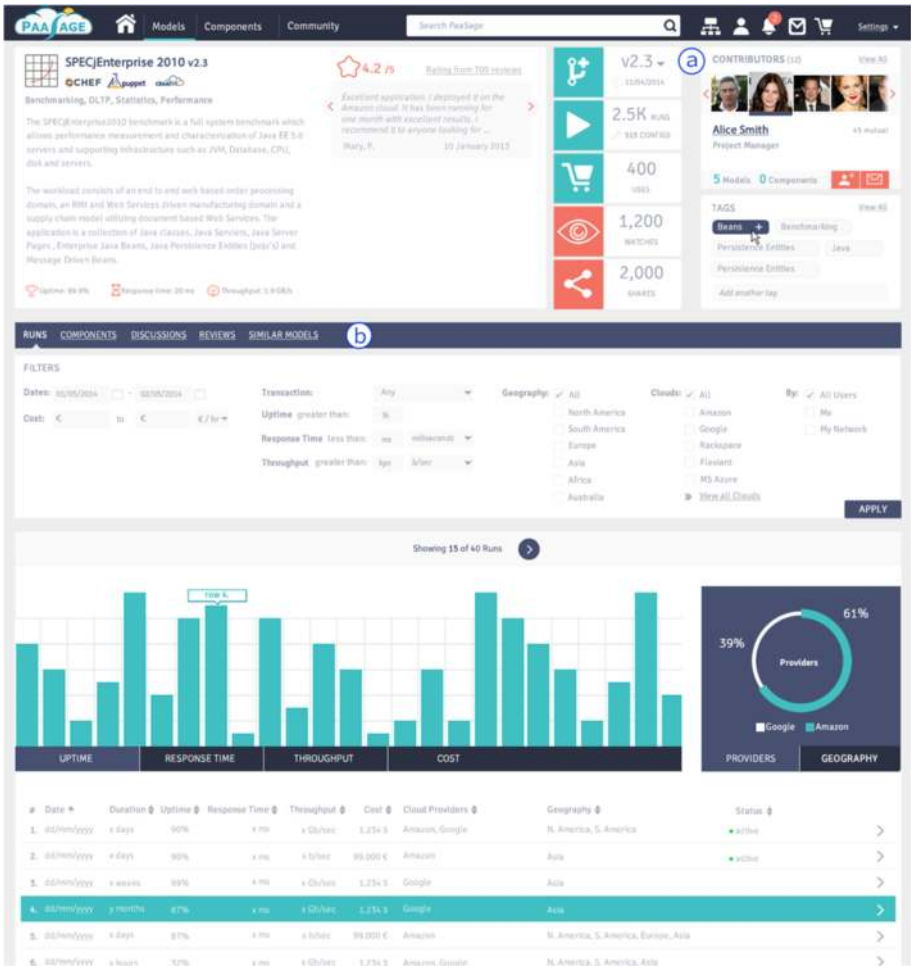

Fig. 6 Application model page

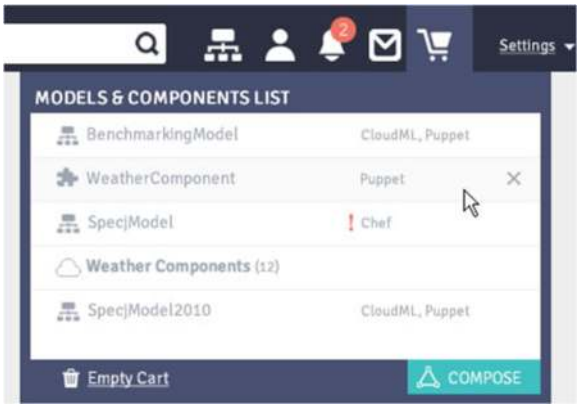

Fig. 7 Models and components list

application and/or component models of interest for later use.

Among cart features, the most advanced is the ability to collect entire categories of components (e.g., Web Servers). The PaaSage professional network relies on proven external model editors for creating or modifying models. The cart can deliver components to such an editor, in which model composition and further editing would take place, prior to importing the new model back into the platform.

To further support engineers, new models are initially marked as *drafts* and their visibility limited to the model owner and any contributors explicitly invited. At any time the owner is able to *publish* a model to the community by marking it as *published*. From that point on the model becomes public and every user can use, clone, or deploy it. Prior to publishing, the owner is also able to determine whether the model is a self-contained element that can be deployed as-is (i.e., application model) or it is a composite component that must be embedded in other application models.

4.1.3 User's personal area

An important component of the network is the *user's personal area*, which reflects the user's activity towards building and deploying models and components. It is also an information retrieval point on community activities related to the user's models. The PaaSage professional network introduces various tools in a user's personal area to facilitate management of model-based distributed applications. Users get live statistics of their active executions and can easily control them (e.g., stop, undeploy one or more instances) (Fig. 8). They can modify the details of individual configuration schemes (e.g., VM properties, resource allocation, distribution, etc.) on a model-by-model basis. For convenience, context-sensitive actions and aggregated statistics regarding engineering and social performance (e.g., number of uses, rating, average execution time, etc.) are visualized adjacently.

4.2 Community

Apart from social features that facilitate distribution and promotion of application models and components

Fig. 8 List of currently running applications

(e.g., share, rate, review, watch), the professional network encourages the creation of an active organized community that supports collaboration (Fig. 9).

People with similar interests can connect and follow each others' updates. The formation of groups promotes a team spirit, exchange of opinions, and knowledge creation through the accumulation of experiences. Members are encouraged to participate in discussions by posting their questions and answers on topics of interest and rate or review the answers of others. To further contextualize discussions, the system offers the ability to associate posts with references to models and components (similar to attaching a file in an e-mail), while instant and automatically-generated hints are provided when possible by the network's knowledge base. Figure 10 depicts an example where two hints are automatically generated via simple queries (constructed based on the keywords "Amazon" and "SpecJEnterprise" found in the question body) to the CAMEL repository. They are meant to inform the user about the quantity of information that is possibly relevant to their question. We envision a scheme where a question is associated with higher-level knowledge drawn from the knowledge base, this is however subject of ongoing research and out of scope of this paper.

Community growth is stimulated by intelligent suggestions on connections or groups that may interest a user. Besides invitations to connect with other members or to join new groups, sugggestions may also encourage individuals to endorse their connections for skills or even invite members to groups to strengthen community bonds. An adaptive approach has been employed to deliver and present the ideas: prompts are either displayed on dedicated areas (e.g., sidebars) or embedded in the main content area, mixed with existing notifications (e.g., updates in models in watchlist, new answers on topics that the user currently follows, etc.). To support new members, the system integrates advanced help facilities, including FAQ and Q&A sections, with intelligent mechanisms that discover and suggest similar questions or relevant answers.

4.3 User interface design process and practices

The user-facing interface of the professional network was designed following an *iterative approach* [55, 56], alternating design with expert-based evaluation involving both usability and domain experts. The main benefit of this approach is that problems are identified early in the development lifecycle and can be corrected before the implementation, even in cases where the agile programming method is applied [57], ensuring high quality of user experience [58]. Discussion of user requirements and evaluation of the designed mockups were carried out in focus groups involving users of social and professional networks, software engineers, cloud computing experts and usability experts.

The focus groups involved seven participants with the following characteristics: two cloud computing experts with limited experience in social and professional networks, three software engineers –all of which were regular users of social networks and one was also experienced in using professional networks– and two usability experts, one of whom had some experience in cloud computing platforms and DevOps environments. During the iterative design process the same group of experts and users was engaged in discussions regarding requirements elicitation

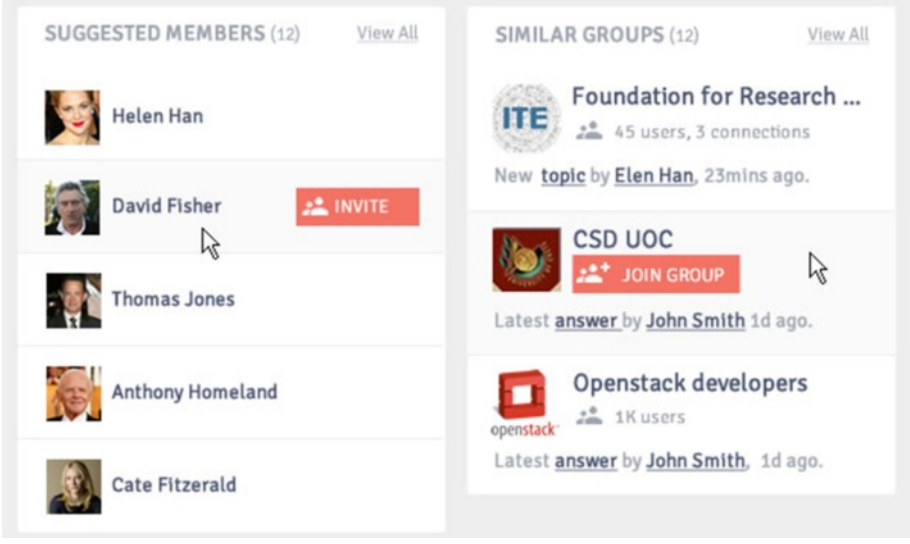

Fig. 9 Users are constantly motivated to participate in social activities

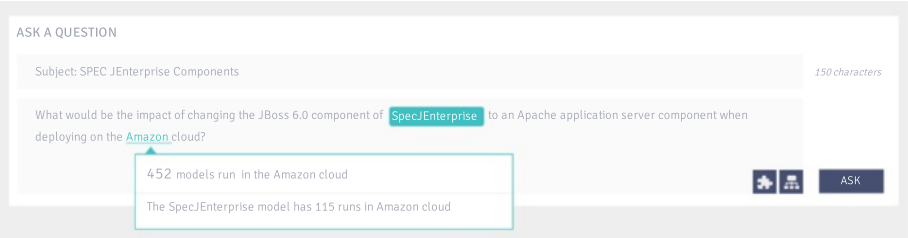

Fig. 10 Automatic provision of context-sensitive assistance when available

and evaluation of the mockups. Given the extent of the design (100 mockups were produced) the process required 13 sessions to evaluate the mockups. The discussions carried out within the focus group led to thorough redesigning and usability fixes for several parts of the professional network. An example of how the design of a system component (the Application Model Overview Panel) evolved over time using our approach is illustrated in Fig. 11.

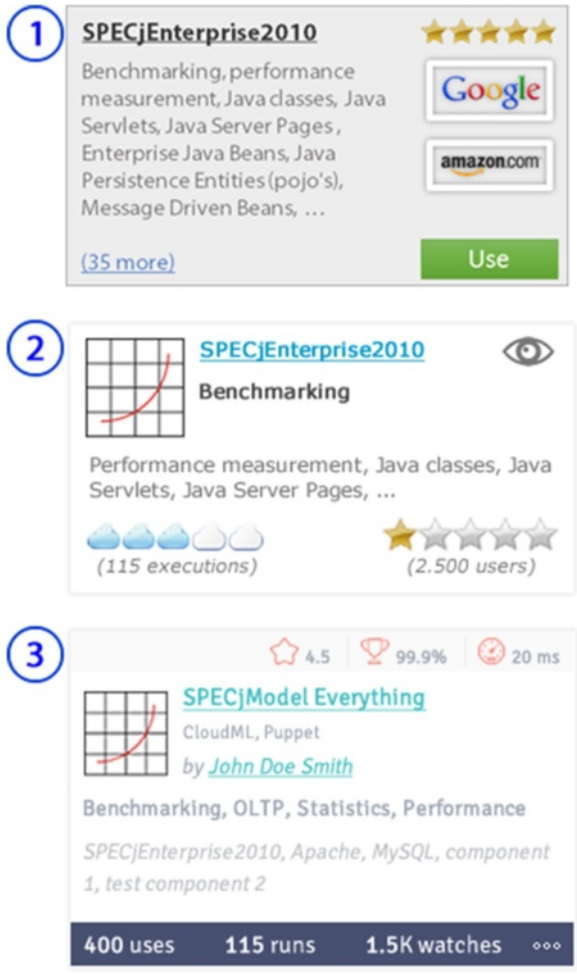

Fig. 11 Evolutionary design of the application model overview panel

Overall, the design of the PaaSage professional network had five objectives:

4.3.1 *Strong bond between software engineering and community-oriented activities*

The design ensures that pages oriented towards software engineering information include details and prompts for community-building and vice versa in order to exploit the generational experience [59] and enable new ways for software developers to work together [60] and build stronger bonds [61]. For instance, in the Application Model page (Fig. 6) users can view information about the model and *also*: (i) information on model contributors, enhanced with direct options for sending them a message or asking them to be connected; (ii) the most popular tags that describe the model, with facilities for adding tags; (iii) overall rating of the model and most popular reviews. Similarly, in a user's Profile page, his/her top model and component contributions are displayed at a prominent area, along with ready-to-use options for using or running these models. The two types of user actions are seamlessly interweaved; being distinct perspectives however, they can be easily distinguished through design cues, such as different colors used consistently throughout to indicate different action types: *blue (turquoise)* for model-related and *red (terra cotta)* for community-related activities. Figure 4 summarizes these concepts.

4.3.2 *Prioritization of personalized information*

Personalized information [62] (relating content to the user and his activities) is included and prioritized in all relevant pages to increase persuasion [63] and facilitate participation for both novice and expert users [64]. For example, when browsing application or component models, system recommendations for models that might be of interest to the user are displayed topmost. In the Application Model page a user can select to view and manage his own configurations for the specific model. The importance of the user's personal stuff is also reflected in the design of the Main Menu (Fig. 12). The Main Navigation Menu can be divided into three sections: (i) user's homepage and navigation to network content (applications, components, community); (ii) search; and (iii) user's personal

Fig. 12 Main navigation menu

material, including his own model and component designs and deployments, profile, notifications, messages, cart and settings. Thus, the design promotes three modalities of interaction: *browsing*, *searching for*, and *personally contributing content*.

4.3.3 Rich home page

An important concern during the design of any website is its home page functionality and layout. The home page is the flagship of every site [65] and should be designed accordingly, displaying vital information both to first-time and regular users. Such information may include [66]: site identity and mission, site hierarchy, search, content and feature promotion, timely content, and shortcuts. Along these guidelines, the home page of the PaaSage professional network was designed to act as an information point for activity related to the user's models and components and to community activity that might be of interest.

The home page of a signed-in user includes: (1) Statistical information and graphics regarding the user's models and components (e.g., top rated and most-run models, number of reviews, discussions, runs, uses and watches of his models and components, etc.) as shown in Fig. 13; (2) Live feed including model- and component-related information (e.g., new models or components made public, updated configurations of models, badges received and new reviews for a model or component, etc.) and community-related facts (e.g., who has an updated profile or was endorsed for a skill, who has made new connections, who posted questions and answers regarding a specific topic, etc.); (3) Information on the user's profile, such as profile completeness, skills and areas of interest. (4) Suggested content according to the user's profile and recent activity, including potentially interesting models, groups and connections.

4.3.4 Nurture an active and sustainable community

A vital consideration for any online community is how actively its members are involved in it, which can be determined by the quality of the content itself [67] and by the design of its user interface [68, 69], which should support and facilitate communication, collaboration, and content contribution. To nurture an active and sustainable community our design applies the following practices: First, it makes content contributions (models, components, questions, answers, ratings and reviews) visible to the community. Second, it cultivates a sense of belonging by presenting activities and contributions of familiar people which may be relevant to the current context. Third, it supports both major contributions (e.g., models or components) as well as smaller contributions (e.g., ratings, reviews, or question-asking). Fourth, it recognizes high quality contributions (e.g., top rated models,

Fig. 13 Statistical information and graphics for user's models and components

mostly up-voted answers) as well as high quantity contributions: the more a user contributes, the more expertise points he gains (top contributors receive special badges as an indication of expertise). Fifth, it enhances the feeling of safety and trust, by making links to the terms of services and privacy policy easily accessible through any network page. Furthermore, users can define privacy settings regarding who can see their data, send requests to them, or look them up [64]. Finally, any user can report members or groups to network administrators if they feel that they violate intellectual rights or exhibit abusive behavior.

DevOps professional communities have indeed shown interest in supporting and using such platforms. For example, the community that has formed around Chef Supermarket is an active and sustainable one as shown from an analysis of cookbook popularity figures (Table 2). While Opscode, Inc. is the single largest cookbook maintainer in the repository, about 96 % of the cookbooks in the site are maintained by other developers, students, organizations, etc. Most of the shared cookbooks seem to be maintained by independent developers rather than companies, if we take contributor e-mail domain as an indication (gmail.com is by far more popular (507 users) than opscode.com (121 users) or getchef.io (68 users)). Finally, a majority (about 62 %) of cookbook maintainers have shared only a single cookbook. We expect that the Chef Supermarket community will share characteristics with the community to be formed around our professional network.

4.3.5 Motivate active and regular participation

Following recent trends in HCI design and with the aim to motivate active and regular participation in the PaaSage professional network, the design employs *gamification features*, namely use of video game elements to improve user experience and user engagement in non-game services and applications [70]. One gamification feature in our current design is the reward system for active community members. As users contribute content (models, components, ratings, reviews, questions, or answers) they receive experience points leading to special badges visible to all community members. Other features are the *Profile completeness bar* with suggestions on how to increase it, and a *skills' endorsement system*: a user can be endorsed by others for specific skills. Such endorsements are visible to all his friends' live feeds, increasing his popularity in the community. Finally, the concept of *Model badges* awarded to application and component models in case of excelling performance. Badges can serve among others as goal-setting devices, status symbols, and indications of reputation assessment procedures [71].

We next describe the implementation of our current professional network prototype.

5 Implementation

The social networking platform is implemented over the Elgg extensible social network framework [72]. Elgg is open-source software written in PHP, uses MySQL for data persistence and supports jQuery for client-side scripting. The Elgg framework is structured around the following key concepts:

- *Entities*, classes capturing concepts such as users, communities, application models, etc.
- *Metadata* describing and extending entities (e.g., a response to a question, a review of an application model, etc.).
- *Relationships* connecting two entities (e.g., user A is a friend of user B, user C is a contributor to an application model, etc.).

Elgg comprises a core system that can be extended through *plugins*. The core comes with a few basic entities (Object, User, Group, Site, Session, Cache) as well as

Table 2 Popularity of Chef cookbooks (as of June 30, 2015)

Rank	Total downloads	Followers	Versions
1	`mysql` (108,162,744)	`mysql` (561)	`mysql` (115)
2	`java` (64,872,884)	`nginx` (541)	`newrelic` (71)
3	`apache2` (64,149,630)	`apache2` (448)	`docker` (59)
4	`docker` (64,114,027)	`java` (301)	`apache2` (57)
5	`newrelic` (61,039,635)	`apt` (266)	`chef-client` (56)
6	`bacon` (60,405,762)	`chef-client` (253)	`java` (56)
7	`nginx` (54,673,163)	`git` (238)	`bacon` (55)
8	`windows` (54,263,225)	`postgresql` (226)	`windows` (54)
9	`chef-client` (54,073,080)	`build-essential` (194)	`nginx` (48)
10	`apt` (51,555,051)	`php` (170)	`noosfero` (47)

other classes necessary for the operation of the engine. All Elgg objects inherit from Entity, which provides the general attributes of an object. Plugins add new functionality, can customize aspects of the Elgg engine, or change the representation of pages (examples are the Cart system or the handling of Application Models). A plugin can create new objects characterized (through inheritance of Entity) by a numeric globally unique identifier (GUID), owner GUID, and Access ID. Access ID encodes permissions ensuring that a user touches only data it has permissions on.

Figure 14 shows the model, view, and control parts of Elgg's architecture. In a typical scenario, a web client requests an HTML page (e.g., the description of an application model, Fig. 6). The request arrives at the *Controller*, which confirms that the application exists and instructs *Model* to increase the view counter on the application model object. The controller dispatches the request to the appropriate handler (e.g., application model, component handler, community handler) which then turns the request to the view system. View pulls the information about the application model and creates the HTML page returned to the web client.

All plugins share a common structure of folders and php files. Folder *actions* includes the actions applied on application models (delete, save, or search). The *views* folder contains the *php* forms applied on application models and *river* events (live feeds). *Pages* overrides elements of core Elgg pages. The *js* and *lib* folder provides javascript and *php* library functions. Finally, the *vendors* folders include third-party frameworks such as Twitter's bootstrap frontend [73] , which lends its responsiveness, look and feel, and portability across Web browsers to the PaaSage professional network..

Social network relationships (friendship, group, ownership, etc.) are persisted in the Elgg back-end database. The execution history of deployments of application models and the description of those models is stored in the CAMEL information repository, which is implemented as an Eclipse CDO server. The exchange of information between the Elgg and CDO servers is implemented over network sockets.

5.1 Components and categories

In the implemented prototype, application component categories as well as individual components have been imported from Chef Supermarket (and periodically refreshed as the master copy of this content remains with Chef Supermarket). Figure 15 depicts those component categories. Each category contains all imported components and associated cookbooks. The distribution of Chef cookbooks over the existing categories and their absolute numbers are depicted in Fig. 16. The total number of cookbooks available on Chef Supermarket on June 30, 2015 was 2303. Our analysis is based on a full download using the `knife cookbook site` command [74]. We have mined additional information on the popularity of cookbooks (most downloaded, followed, and updated) shown in Table 2. We believe that this information is of interest to users (since application deployment relies on execution of cookbooks) and plan to make it available to them.

It is worthwhile mentioning that over time we observed a shift in the way cookbooks are categorized: Initially it was customary for cookbooks in the repository to be associated with more than one categories; however later this changed to just a single category per cookbook. More recently, several cookbooks have been left unclassified or re-categorized in the generic category "Other", including `mysql` (previously in "Databases"), `apache2` (previously in "Web Servers"), and `git` (previously in "Utilities"). On May 2014 about 8 % of the cookbooks were categorized as "Other", while on June 2015 this increased to about 29.5 %.

We believe that this shift away from user-provided cookbook categorization is due in part to (a) the users' lack of belief in the usefulness of such a scheme, as reported by the creators of the community site [75, 76]; and (b) the difficulty with coming up with a consistent and widely-accepted software categorization scheme in Chef Supermarket. We believe that even if users saw the benefits in achieving a universal cookbook categorization (e.g., as a

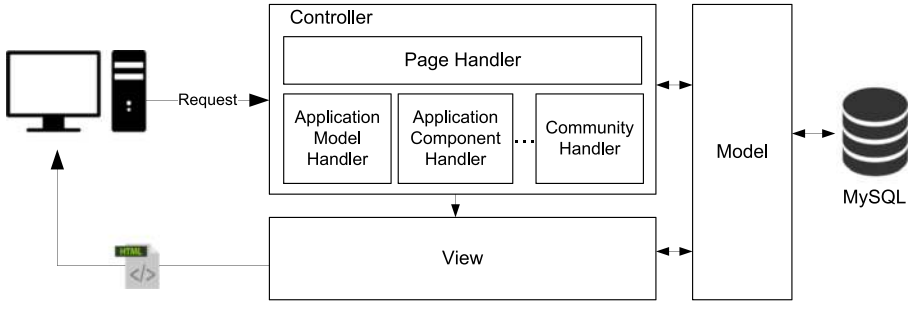

Fig. 14 Architecture of the Elgg Social Networking engine. Page handlers refer to functionality of implemented plugins

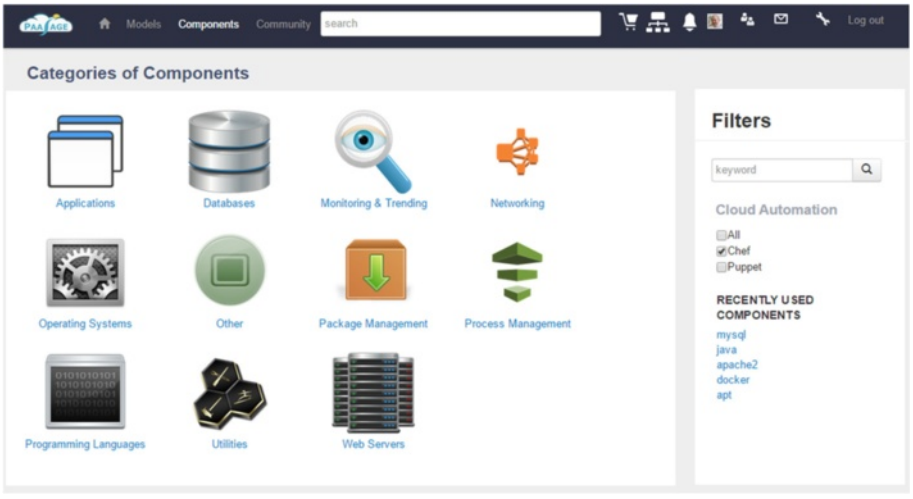

Fig. 15 Application components. Component categories and components imported from Chef Supermarket

enabler of advanced functionality, such as coming up with suggestions for alternative implementations to a specific software component as part of a "what-if" analysis), the only way to achieve it would be through a community-moderated categorization scheme (e.g., majority voting on category proposals) for software components. To be successful, such a scheme would require strong integration between technical and community aspects. Our professional network is uniquely positioned to support such as scheme.

5.2 Application deployment

A key aspect of the PaaSage professional network is its support for deployment of applications. CAMEL (through CloudML) is able to express deployable models of applications that specify the infrastructure upon which middleware and application logic will be deployed and run on.

An important feature of our system is its support for the creation of CAMEL applications through composition of components imported from Chef Supermarket (Fig. 15). We orchestrate the deployment of all application components starting from a CloudML/CAMEL model of the application using the Chef configuration management tool. In what follows we briefly describe our methodology (called *CAMEL-to-Chef* or C2C for short). The reader is referred to [77] for a complete exposition.

C2C comprises three major modules: i) A model parser that analyzes an application model and extracts a list of application components and the nodes (VMs) that the components will be deployed on; ii) the VM manager module responsible for the provisioning of the VMs (using third-party libraries such as JClouds [78] or provider-specific APIs such as Azure SDK [79]) and installs the Chef client in each one; and iii) the Chef instructor, a wrapper around the Chef runtime

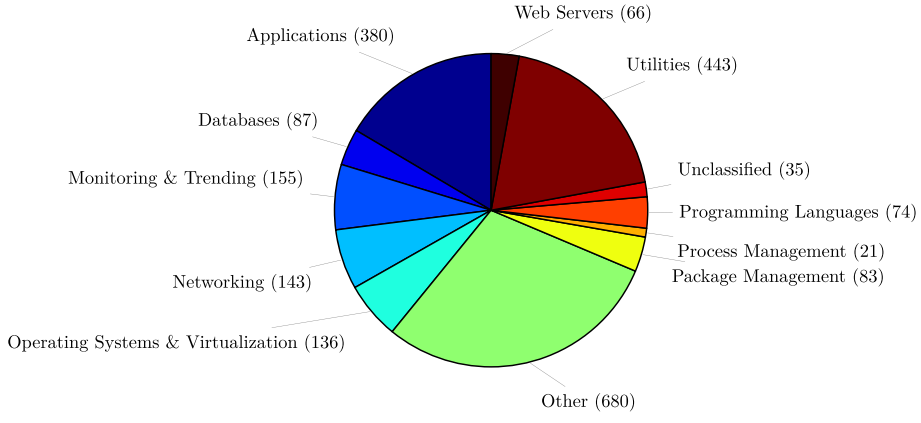

Fig. 16 Distribution of Chef cookbook categories

system. C2C identifies the corresponding cookbooks (Section 3.2) to each application component and deploys them on the appropriate VMs. We assume that cookbooks are imported from Chef Supermarket. It derives the order in which the components should be installed by analyzing the application structure and the dependencies among its components as described in CAMEL model.

Note that there are additional component dependencies (besides CAMEL component dependencies) that are at play here. Each Chef cookbook may define a number of Chef dependencies on other cookbooks describing software that must be installed, configured, and started first. The wealth of information available at Chef Supermarket gives us the opportunity to study those dependencies thoroughly on a large collection of cookbooks. This is interesting to the user of the social networking platform since it provides insight as to how many and which cookbooks are *foundational* (many others depend on them) and therefore their reliability history and other dependability aspects are important. Section 5.3 presents the results of our dependency analysis.

5.3 Chef cookbook dependency analysis

We analyzed information on dependencies between Chef cookbooks in Chef Supermarket as follows. We crafted a *dependency graph* in which each vertex v corresponds to a cookbook and each edge $e = (v, w)$ represents a dependency from cookbook v to cookbook w. For each cookbook (software component) we calculate the number of cookbooks that it supports, i.e. the number of cookbooks that depend on this cookbook, and the number of cookbooks that it depends on. To obtain their indirect dependencies we performed a depth-first search from each vertex v in the cookbook dependency graph and added an edge $e = (v, w)$ in another graph for each unique vertex w that we visited.

In Fig. 17 we observe that both direct and indirect cookbook dependencies follow power-law distributions. We found 1468 of the 2303 cookbooks to depend on at least one other cookbook. This prevalence of dependencies highlights the value of automated deployment frameworks such as Chef. Table 3 lists the top 10 cookbooks with the highest in-degree and out-degree from direct and indirect dependencies.

Our prototype is being extended to include this information (marking components as *"foundational"*, *"components with many dependencies"*, etc.) in the context of the components page.

6 Evaluation

In this section we describe the user evaluation of the PaaSage professional network, as well as the design and deployment of an application model using the platform. During the iterative process of designing the user interface several expert-based evaluations were carried out in group sessions (Section 4). To obtain additional feedback from non-experts, three additional user-based evaluation experiments were designed and carried out involving potential users. This research work does not involve identifiable human material or data and user privacy is ensured in all cases.

6.1 Preliminary evaluation of mockups

The first experiment aimed at assessing the overall look and feel of the network, the navigation mechanisms, as well as the design of fundamental functionality (e.g., home page, model categories, component categories, and model details). Six non-interactive mockups along with a documentation of the design rationale were presented to 10 experienced cloud-computing engineers, in the context of a specific use case scenario. Users were asked to view the mockups and the related rationale and provide their feedback so as to shape the requirements from an end-user perspective. The evaluation resulted in 25 distinct questions that came across from participants on more than one occasions (see Fig. 18). All the issues that were identified focused mainly in (a) functionality that was not showcased through the provided mockups, and (b) general questions regarding the modus operandi of the professional networking platform (e.g., Question 1: As a user who is writing code how do I publish either Models or Components?, Question 9: Is it possible and encouraged to make direct contact with a user who has had a similar experience to my planned model?). In summary, the raised issues reflected users'

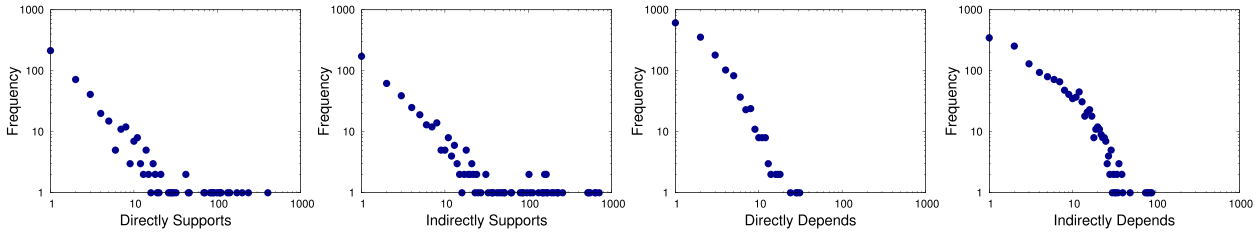

Fig. 17 Distribution of Chef cookbook dependencies in log-log scale

Table 3 Ranking of Chef cookbook dependencies (as of June 30, 2015)

Rank	Directly supports	Indirectly supports	Directly depends	Indirectly depends
1	`apt` (404)	`yum` (708)	`nodestack` (31)	`magentostack` (90)
2	`build-essential` (236)	`build-essential` (645)	`magentostack` (29)	`nodestack` (87)
3	`yum` (199)	`apt` (629)	`stack_commons` (28)	`pythonstack` (82)
4	`java` (172)	`chef_handler` (551)	`platformstack` (24)	`phpstack` (80)
5	`git` (141)	`yum-epel` (528)	`elkstack` (18)	`stack_commons` (76)
6	`apache2` (133)	`windows` (518)	`gitlab` (18)	`jenkinsstack` (50)
7	`runit` (111)	`chef-sugar` (259)	`phpstack` (17)	`rackops_rolebook` (41)
8	`windows` (107)	`packagecloud` (230)	`pythonstack` (17)	`noosfero` (40)
9	`mysql` (106)	`runit` (225)	`noosfero` (16)	`platformstack` (40)
10	`yum-epel` (99)	`dmg` (223)	`ut_workstation` (16)	`boilerplate_php` (37)

concerns regarding the final design and functionality of the professional network and did not focus on usability problems of the specific mockups that were evaluated. As a result, a detailed design of the professional network was created through 100 non-interactive mockups, taking into account all the issues raised in the first evaluation. These mockups were the basis for developing an interactive working prototype of the PaaSage professional network platform, featuring the final interface design and exposing an extensive set of the most important functionality.

6.2 Evaluation of interactive prototype through free exploration

The second evaluation experiment aimed to collect subjective results rather than performance metrics. It involved a different set of 12 users who, after a brief introduction to the available facilities, were asked to use the interactive prototype [80, 81] using the free exploration method of the Thinking Aloud protocol [82] and fill-in a questionnaire in order to rate and comment their experience. Users were recruited through the PaaSage EU project [14] consortium, based on the expertise criteria that all participants should be software developers with at least some expertise in social networking or cloud computing. The interactive prototype to be evaluated included all the necessary functionality for browsing through models and components, viewing detailed model information, and engaging in social activities, such as following and messaging a user, joining a group and posting questions to the group.

The questionnaire comprised four sections: (a) background information (sex, age, expertise), (b) overall system usability according to the System Usability Scale (SUS) questionnaire [83], (c) assessment of specific system features (registration, finding a model, model information,

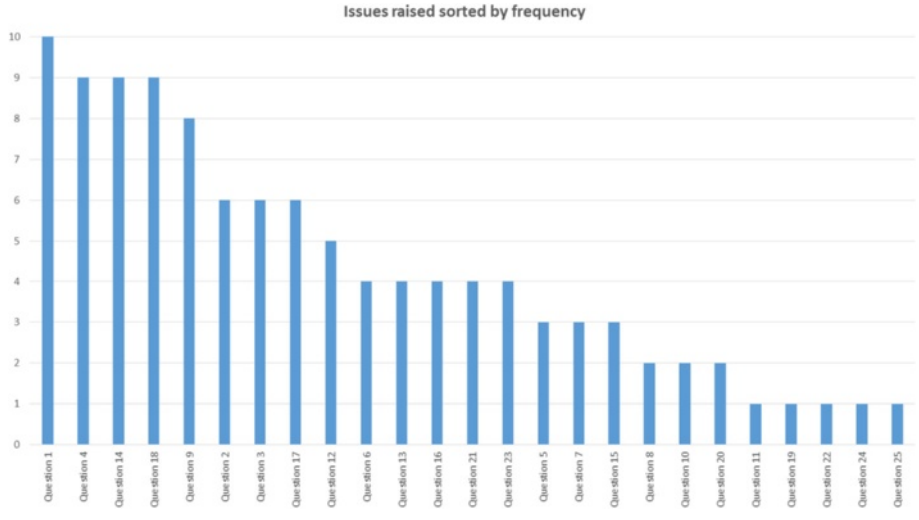

Fig. 18 Frequency of issues raised during the first evaluation iteration with users

community features) rated on a five point Likert-scale and (d) additional information, where participants were asked to identify the three features that they liked most, the three features that they disliked most, and provide additional comments. Statistics regarding the participants' gender, age, social network expertise, cloud computing expertise, and IT expertise are provided in Table 4.

As shown in Fig. 19, the overall SUS score for the professional network prototype (section B of the questionnaire) was 68.5 (a SUS score above 68 can be considered above average), while the SUS score from expert users was 77.8 and the SUS score from users with medium expertise was 55.5 (below average).

Rating of the individual network features (section C of the questionnaire), as shown in Fig. 20, indicated that users encountered difficulties in finding specific models and viewing their information. As explained by additional comments provided by some of the users, an important shortcoming was that only a small number of models were included in the prototype and only one model included rich information for users to view.

Analysis of users' responses in section D of the questionnaire (most liked and most disliked features) revealed that users most commonly favored the user interface of the network, the potential to share models and components with experts in the field and the ability to discuss with others and participate in groups of users with similar interests. On the other hand, users' responses regarding the features they disliked most indicated that medium expertise users would prefer an introductory video, a user guide, or tooltips in order to help them understand how the site works. Furthermore, most users commented on functionality that was missing or partially implemented, such as editing user profile, searching for friends, resetting password, and filtering model runs. Finally, users pointed out the lack of content as a problem that negatively affected their experience with the prototype. In summary, evaluation of the interactive prototype indicated that users liked the user interface as simple and professional and that they highly appreciated the facilities provided by the social network allowing model sharing, discussions and networking with users of similar interests.

Most of the negative comments and ratings are a result of the small size of the current content and evolving implementation. This was due to the nature of the experiment: aiming to receive as many comments and suggestions as possible for functionality that users desire to see included, they were asked to freely use the prototype, vocalize their thoughts and fill-in the evaluation questionnaire, rather than been taken through specific scenarios in an observation experiment.

Table 4 Demographic information of participants of the second evaluation experiment. Social network, cloud computing and software development expertise was provided by the participants through rating their own level of exposure (scale of 1 to 5)

Gender			Social network expertise		
Male	8	66.7 %	Very high	2	16.6 %
Female	4	33.3 %	High	6	50.0 %
TOTAL	12	100.0 %	Medium	2	16.6 %
			Low	0	0.0 %
			Very low	2	16.6 %
Age			TOTAL	12	100 %
<30	7	58.3 %			
30 - 40	5	41.7 %			
TOTAL	12	100.0 %			
Cloud computing expertise			Software development expertise		
Very high	3	25.0 %	Very high	1	8.3 %
High	4	33.3 %	High	6	50.0 %
Medium	5	41.7 %	Medium	5	41.7 %
Low	0	0.0 %	Low	0	0.0 %
Very low	0	0.0 %	Very low	0	0.0 %
TOTAL	12	100.0 %	TOTAL	12	100.0 %

6.3 Evaluation of prototypes through scenarios and interviews

6.3.1 Methodology

The third evaluation experiment involved 15 participants who were guided through the interactive prototype of the PaaSage social network using specific scenarios. They were interviewed on their requirements and feedback following a semi-structured interview approach [84] that included both open-ended and close-ended questions. The evaluation session was carried out via Skype [85]. Participants were recruited through European companies and organizations associated with the PaaSage EU project [14]. They were either developers or operations staff, thus within the target user groups of the PaaSage social network. Participants were not users of the PaaSage platform; some however were familiar with the project's goals and objectives. Before the experiment, each participant was requested to fill-in a background information form and was sent an informed consent form, explaining all the recording and anonymity-ensuring procedures. A more detailed analysis of the participants' profile is presented in Table 5.

Each interview session was structured as follows: *(i)* introduction to the purpose of the evaluation experiment and the procedures that will be followed; *(ii)* video-recorded consent to participate in the experiment; *(iii)* their requirements from DevOps platforms and social

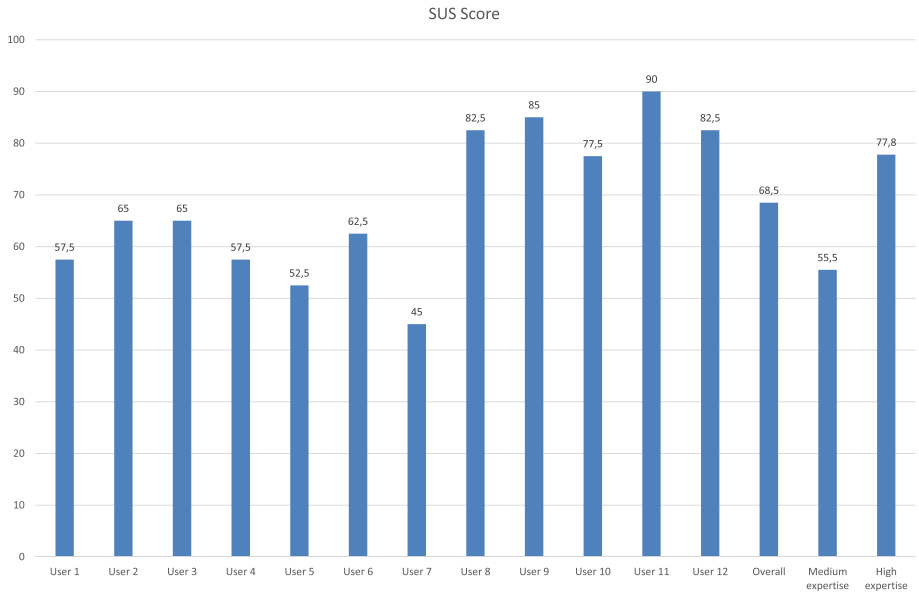

Fig. 19 SUS scores

networking sites; *(iv)* use of the PaaSage social network through given scenarios; *(v)* feedback regarding the PaaSage social network; and *(vi)* debriefing. Upon completing these sections, the interviewer read his/her notes to the participants and asked them (if needed) to clarify notes or add more comments prior to ending the session. All sessions were audio recorded. To ensure anonymity and data safety, each user received an ID with which he/she was referred to, while all documents and recordings of the participant were password protected.

6.3.2 User requirements

Questions regarding users' requirements from DevOps and social networking environments aimed at receiving information about which specific DevOps and social networking platforms each participant uses, how frequently they use them, features that they like or dislike in these environments, and features that are missing. When a participant was not a DevOps or social network user, they were asked why they were reluctant to use them. In all cases, the participants' willingness to try a new

Fig. 20 Scores of the individual PaaSage features: registration, finding a model, model information, finding a component

Table 5 Demographic information of participants of the third evaluation experiment. Professional expertise was provided by participants through an open-ended question. Several participants indicated more than one domains of expertise

Gender			Age		
Male	12	80 %	<30	6	40 %
Female	3	20 %	30–40	9	53.3 %
TOTAL	15	100 %	Undeclared	1	6.7 %
			TOTAL	15	100.0 %
DevOps expertise			Social networking expertise		
0–1 years	3	20 %	0–1 years	1	6.7 %
1–2 years	1	6.7 %	1–2 years	0	0.0 %
>2 years	11	73.3 %	>2 years	14	93.3 %
TOTAL	15	100.00 %	TOTAL	15	100.00 %
Professional expertise*			Familiarity with PaaSage objectives		
Software Engineer	13	86.7 %	Limited	5	33.3 %
Researcher	6	40 %	Medium	4	26.7 %
Solutions Architect	2	13.3 %	High	6	40.0 %
Other	1	6.7 %	TOTAL	15	100.0 %

environment was explored both in general and in particular for an environment that would combine both DevOps and social networking features.

In relation to DevOps platforms, most of the participants were GitHub (80 %), GoogleCode (33.3 %), StackOverflow (33.3 %), or BitBucket (26.7 %) users. Users reported that the features they liked most about the DevOps environments they use were: (1) related to developer activities: issue and activity tracking, bug reporting, viewing project statistics and following specific projects; and (2) relevant to community activities around the projects: user collaboration, ranking and rating, commenting. Features missing from these platforms include instant messaging, project management facilities, personalization characteristics (e.g., suggestion of possibly interesting projects according to each user's profile).

In relation to the social networking platforms, most of the participants were Facebook (80 %), LinkedIn (73.3 %) and Twitter (60 %) users. The majority of users reported that they use those platforms for personal and professional activities. Professional activities include information retrieval, connection with other professionals, seeking job opportunities, and promotion of oneself by making visible one's work. Features that users favored include communication with other people, fast information retrieval, linking with other individuals with similar interests and joining a large community of users. A major concern reported by nearly all participants is data privacy and security. Furthermore, some users reported that they were annoyed by e-mail pushing regarding activities carried out in the network.

Participants were asked if they would be willing to use a new platform that combines some DevOps features with social features, as well as what requirements they would have for such a platform in order to use it and prefer it in comparison to the other platforms that they use. All participants except one (who was not a developer) were positive to the idea of using such an environment. Several highlighted the fact that currently they have to use several different social networks to communicate with experts in their field. Features that the platform would have to deliver include privacy control, source code sharing and management options, good search and filtering mechanisms, user collaboration and communication facilities, as well as personalization features (such as the suggestion of potentially interesting people or projects according to the user's interests and current content that he is viewing).

6.3.3 Exploration through scenarios

Following the initial interview, participants were asked to explore the PaaSage network using a specific scenario that involved logging in, locating a specific application model (the SPEC jEnterprise2010 model described in Section 6.4), viewing its information, and selecting a specific execution of the model (the most cost-effective execution of the model on a specific cloud) to view its configuration. Each participant was then asked to visit the

profile page of a specific model contributor, add him to the users he/she follows and view other models that this network user has contributed. Finally, each participant was asked to find a specific group in the PaaSage social network community, become a member, and post a question to the group.

During this process, participants were instructed to share their screen through Skype so that the interviewer could guide them through the scenarios and also observe their interaction with the system. The analysis of users' interaction with the system resulted in the identification of specific problems, which in most cases were also confirmed by the participants during their feedback elicitation process. For instance, issues that were identified included the visibility of certain UI elements, incomplete implementation of specific platform features (e.g., search), lack of integrated model editor, use of inappropriate labels in some cases, and lack of instructions or help for the more complicated pages.

6.3.4 User feedback

After going through the scenario, participants were interviewed regarding their opinion for the PaaSage social network. They were asked to rate on a scale of 1–10 how easy it was to find a specific model and if the model information was satisfying for them. For rates lower than 8, they were asked to identify what posed difficulties in their interaction and what they did not like about the specific network features. Model-finding received an average rate of 9 (standard error of the mean: 0.31), while model information was rated 8.4 on average (standard error of the mean: 0.34), as shown in Fig. 21.

The lack of a fully implemented search facility was reported as the feature that prevented users from easily locating a specific model, since six of the users tried to locate the model through search and not through navigation on the available content. Additional information requested for the model page was: specific instructions on how to deploy a model, a graphical representation of the model, and to what percentage each contributor participates in the development of a model. Furthermore, some users found that due to the amount of information on the model page they needed additional time and instructions to view and fully understand it.

Next, users were asked to identify DevOps or social networking features that they would like to have seen in the network. DevOps features that were requested (Fig. 22) included code sharing, software project management facilities (e.g., activity tracking, versioning, milestones' management), instructions and scripts for model deployment, application binaries, and direct model execution. The few social networking features that were suggested were: messages with multiple recipients and attachments, chat facility, flagging of discussions as interesting to follow.

Users were asked to identify up to three most-liked and three most-disliked features. Most-liked features included (Fig. 23): the employment of social features in a development environment; the use of charts and statistical information to represent data; the detailed model information that could be retrieved and the model execution histories; the automatically-generated hints provided by the network as replies in discussion topics; the concept of sharing one's models; the overall UI design; the direct connection between projects and users, and the user profile tags that allowed them to find users that would be interesting to connect with.

Most-disliked social networking platform characteristics included aesthetics issues (e.g., fonts, appearance of error messages, responsiveness); low visibility for certain components (e.g., model filters, suggested groups); extensive information in the model page; the lack of

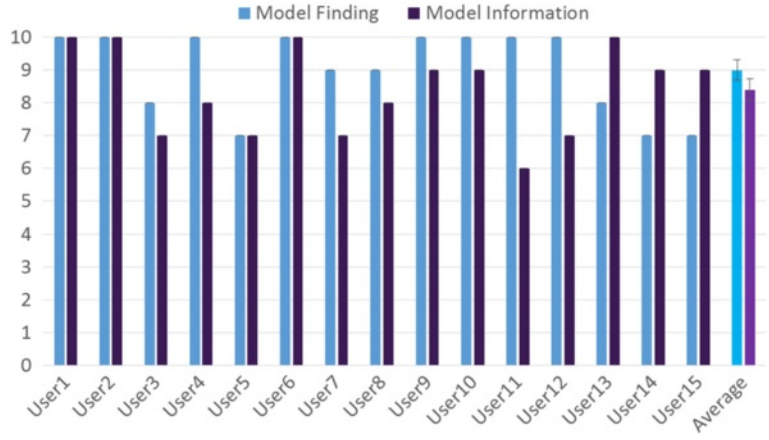

Fig. 21 Users' rating of model finding and model information

Fig. 22 User requests for DevOps features to be included in the PaaSage social network

fully-implemented search facility, and the current representation of models through XMI files. While there was mostly consensus between participants on most-liked features, most-disliked features were raised with low frequency (usually one -different- issue per participant), while several participants provided only two disliked features.

Wrapping up the interview, participants were asked if they would be willing to share their models in a social networking platform like PaaSage. For positive answers, they were further asked what benefits they thought it would bring to community members, as well as what kind of models they would like to see in such a platform. For negative answers, they were asked why they were skeptical. All participants provided a positive answer; some participants however expressed concerns regarding privacy issues for sharing source code and application models, especially in the case of commercial projects. The foreseen benefits for the users relate to knowledge sharing, learning by example (e.g., starting from existing models and adapting them), finding resources related to one's interests through the concepts of similar models and model contributors, as well as network building with field experts. Regarding the type of models that users would like to find in the PaaSage social network most participants suggested well-known open source projects and applications, while some users suggested specific models related to their interests (e.g., from the IoT, Big Data area, and NoSQL datastore space).

6.3.5 Discussion

A common request of most participants was support for code sharing. In most instances we had to explain that our platform managed application *models* not their software impementations, which can be hosted independently on platforms such as GitHub. That said, there is an inter-relationship between the model and the software implementation of an application that we must take into account: A specific version of the model corresponds to a specific version of the code, and a new software release may necessitate a change in the model representing the application. An important reason for such *coevolution* is that a specific execution history should refer to the code used in the deployment that produced it (as performance obviously depends on the application version used). There is thus a need for our platform to maintain model versions referencing code versions (when possible) in code-hosting platforms, which is a topic of ongoing work.

Participants additionally highlighted the need for direct model deployment and execution (component binaries, execution scripts, etc.) and access control and privacy of models, executions, etc. Direct model deployment is addressed by C2C (Section 5.2), which unfortunately

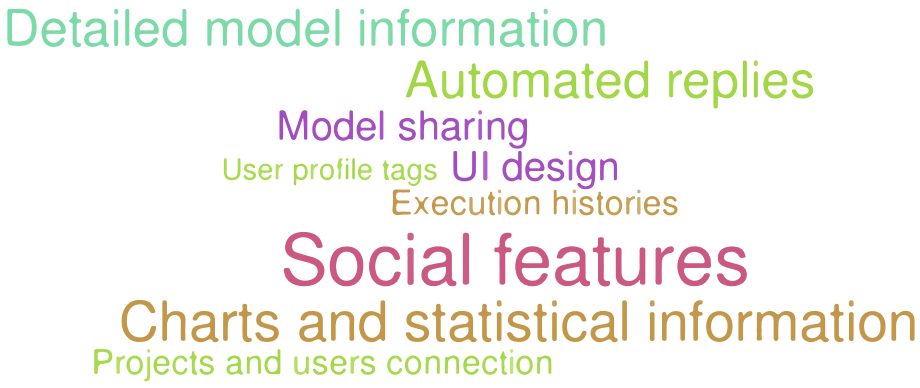

Fig. 23 PaaSage features that were mostly liked by the users

could not be showcased in interviews due to time constraints. Access control and privacy are active areas of research within the PaaSage EU project [14] and beyond the scope of this paper.

Overall, the results underscore the positive aspects of the PaaSage social networking platform. Positive highlights include the importance of application execution histories, the automatically generated hints based on data analysis, and the employment of social features in community-building activities. The results confirm our expectation that DevOps users see value in joining this online community and contributing to it.

6.4 Application modeling, deployment on multi-cloud configurations, and analysis of execution histories

In this section we focus on SPEC jEnterprise2010 [86], a distributed application and full system benchmark that allows performance measurement and characterization of Java EE 5.0 servers and supporting infrastructure.

Using a standard EMF model editor [87] we created a CAMEL model for jEnterprise2010 that uses three software components corresponding to the business logic of the application, the application server (JBoss 6.0), and a RDBMS (MySQL 5.5).

The CAMEL model of jEnterprise2010 is automatically deployed using C2C as described in Section 5.2. By varying the deployment configurations on the CAMEL model (number and type of VMs and their cloud provider) we deployed SPEC jEnterprise2010 on 14 different setups comprising single public-cloud providers (Amazon EC2, Microsoft Azure), private clouds (Openstack Nova), and multi-cloud configurations combining resources from Amazon EC2 and the private cloud platform. Every deployment uses a different configuration and/or different types of VMs.

A deployment configuration may be executed a number of times. Each execution is monitored and a variety of performance and reliability metrics (otherwise called an *execution history* in CAMEL) collected for each run. A monitoring mechanism is envisioned and under development within the PaaSage execution platform [14], however other mechanisms can be adapted to contribute execution histories to this repository. Each such execution history comprises aggregate statistics rather than long and detailed time series about each metric. It is inserted in the CAMEL model for the application within the repository via an appropriate CDO client [45].

A large number of execution histories over different deployment configurations collected through contributions by the community can provide valuable knowledge to PaaSage social network users. As one example, we can answer questions about which deployments work best in terms of performance, reliability, cost, and their combinations. An excerpt from the UI shown in Fig. 24 depicts execution histories of the SPEC jEnterprise 2010 application for our 14 different deployment configurations. Execution histories are ordered by *cost effectiveness*, defined as performance (the exact metric may vary) divided by estimated cost. Our user evaluation confirms that such data-driven analysis and feedback is indeed perceived as offering significant new value to the DevOps community.

7 Conclusions

In this paper we presented the design and implementation of the PaaSage social networking platform, a novel platform for professional networking targeting the community of DevOps engineers. The network combines community knowledge with information from two repositories, Chef Supermarket and the CAMEL repository of application models and executions, to improve the configuration, deployment, and optimization of distributed multi-cloud applications, tasks of major interest to cloud deployment specialists. The design of our professional network applied best practices aiming to support

Start	End	Uptime	AVG throughput	browse RT	manage RT	purchase RT	Cost	Cloud Provider	Geography	Cost Effectiveness	Status
Fri Sep 05 20:31:20 EEST 2014	Fri Sep 05 21:08:00 EEST 2014	2200 sec	16.98	0.22	0.22	0.17	0.07	Amazon	Ireland	242.63	stopped
Sat Sep 06 19:52:11 EEST 2014	Sat Sep 06 20:28:51 EEST 2014	2200 sec	17.22	0.24	0.24	0.18	0.07	Amazon	Ireland	246.10	stopped
Fri Sep 05 18:13:00 EEST 2014	Fri Sep 05 18:49:40 EEST 2014	2200 sec	17.055	0.20	0.20	0.14	0.14	Amazon	Ireland	121.82	stopped
Fri Sep 05 18:27:55 EEST 2014	Fri Sep 05 19:04:35 EEST 2014	2200 sec	16.86	0.19	0.19	0.14	0.14	Amazon	Ireland	120.46	stopped
Sun Sep 07 14:48:40 EEST 2014	Sun Sep 07 15:25:20 EEST 2014	2200 sec	17.13	0.13	0.13	0.09	0.35	Amazon	Ireland	48.95	stopped
Sun Sep 07 16:02:18 EEST 2014	Sun Sep 07 16:38:58 EEST 2014	2200 sec	17.06	0.15	0.15	0.10	0.35	Amazon	Ireland	48.75	stopped
Tue Sep 02 22:40:29 EEST 2014	Tue Sep 02 23:17:10 EEST 2014	2201 sec	17.09	1.41	1.41	1.08	0.06	Azure	Netherlands	284.90	stopped
Tue Sep 09 16:12:32 EEST 2014	Tue Sep 09 16:49:13 EEST 2014	2201 sec	16.99	0.00	0.00	0.48	0.03	GWDG	Germany	566.33	stopped
Tue Sep 09 17:31:11 EEST 2014	Tue Sep 09 18:07:51 EEST 2014	2200 sec	17.03	0.60	0.60	0.58	0.03	GWDG	Germany	567.90	stopped
Wed Sep 10 13:24:28 EEST 2014	Wed Sep 10 14:01:10 EEST 2014	2202 sec	16.95	0.06	0.06	0.67	0.03	GWDG	Germany	565.13	stopped
Wed Sep 10 15:07:15 EEST 2014	Wed Sep 10 15:43:56 EEST 2014	2201 sec	16.97	0.67	0.67	0.63	0.03	GWDG	Germany	565.83	stopped
Tue Aug 05 22:44:11 EEST 2014	Tue Aug 05 23:19:11 EEST 2014	2100 sec	20.11	0.58	0.58	0.44	0.07	Amazon	Ireland	287.29	stopped
Wed Aug 06 13:49:59 EEST 2014	Wed Aug 06 14:09:39 EEST 2014	1000 sec	20.04	0.73	0.73	0.60	0.07	Amazon	Ireland	286.39	stopped
Wed Aug 06 16:05:33 EEST 2014	Wed Aug 06 16:22:33 EEST 2014	1020 sec	20.25	0.36	0.36	0.27	0.14	Amazon	Ireland	144.70	stopped

Fig. 24 The execution history of different cloud deployments

the creation of a vigorous community, to allow users to retrieve timely and appropriate information and to carry out actions in a small number of steps. Tightly interweaving professional with social content and actions, as well as providing personal information wherever and whenever suitable was one of the main design concerns.

The positive findings of the three user evaluation experiments conducted with domain experts, cloud computing engineers, and representative end-users, highlight the potential of the proposed system since many of its abilities such as model sharing, discussions and networking with users of similar interests were highly appreciated. We therefore believe that our social networking platform is successful in providing DevOps engineers with new value and thus strong incentives to use it. Future work will include full-scale testing with larger communities of users in order to further evaluate the usability of the designed network and validate the design practices that have been applied.

Endnote

[1]Models and modeling languages enable developers to work at a high level of abstraction by focusing on design rather than implementation details.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
KM conceived of the study, coordinated the design and implementation of the infrastructure of the social networking platform, and had a major role in drafting the manuscript. CP carried out a major part of the implementation of the infrastructure of the social networking platform and was a contributor to Sections 2, 5 (up to 5.1) and 6. AP participated in the implementation of the deployment methodology and was a contributor to Sections 2, 5.2 and 6. FK participated in the implementation of the deployment methodology, was a contributor to Section 5.2 and participated in the analysis of the Chef Supermarket repository. DA participated in the analysis of the Chef Supermarket and contributed to Sections 5.1 and 5.3. NP participated in the analysis of the Chef Supermarket and contributed to Sections 5.1 and 5.3. MK participated in the design of the user interface, participated in the implementation of the UI mockups, and was a contributor to Section 6. AL participated in the design of the user interface, participated in the implementation of the UI mockups, and was a contributor to Sections 4.1, 4.2 and 6. SN participated in the design of the user interface, participated in the implementation of the UI mockups, and was a contributor to Sections 4.3 and 6. CS coordinated the design of the user interface and participated in drafting the manuscript. All authors read and approved the final manuscript.

Acknowledgements
We thankfully acknowledge the support of the PaaSage (FP7-317715) EU project. We are also thankful to Al Innes, Craig Sheridan and Douglas A. Imrie of Flexiant Ltd. for their contributions in the user evaluation study of Section 6.1.

Author details
[1]Institute of Computer Science (ICS), Foundation for Research and Technology – Hellas (FORTH), GR-70013 Heraklion, Greece. [2]Department of Computer Science and Engineering, University of Ioannina, GR-45100 Ioannina, Greece. [3]Computer Science Department, University of Crete, GR-70013 Heraklion, Greece.

Received: 15 October 2014 Accepted: 21 July 2015
Published online: 01 September 2015

References

1. Loukides M (2012) What is DevOps? O'Reilly. http://radar.oreilly.com/2012/06/what-is-devops.html
2. Chef. https://www.chef.io/ Accessed 8/2015
3. Puppet. http://www.puppetlabs.com/. Accessed 8/2015
4. OASIS (2013) OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0 Committee Specification 01
5. Ferry N, Hui S, Rossini A (2014) CloudMF: Applying MDE to Tame the Complexity of Managing Multi-cloud Applications. In: Proceedings of Ucc 2014: 7th ieee/acm international conference on utility and cloud computing, London, UK. pp 269–277
6. Binz T, Breitenbücher U, Haupt F, Kopp O, Leymann F, Nowak A, Wagner S (2013) OpenTOSCA – a runtime for TOSCA-based cloud applications. In: 11th International Conference on Service-Oriented Computing
7. Chauvel F, Ferry N, Morin B, Rossini A, Solberg A (2013) Models@ Runtime to Support the Iterative and Continuous Design of Autonomic Reasoners. In: MoDELS@ Run. time. pp 26–38
8. Howe J (2006) The rise of crowdsourcing. Wired Magazine
9. Github: Social Coding. http://github.com/. Accessed 8/2015
10. Sourceforge. http://sourceforge.net. Accessed 8/2015
11. Google Code. https://code.google.com Accessed 8/2015
12. CodePlex - Project Hosting for Open Source Software. https://www.codeplex.com. Accessed 8/2015
13. Supermarket: Opscode Community. https://supermarket.chef.io/. Accessed 8/2015
14. PaaSage EU FP7 project. http://www.paasage.eu/. Accessed 8/2015
15. Papaioannou A, Magoutis K (2013) An architecture for evaluating distributed application deployments in multi-clouds. In: Proceedings of 5th IEEE International Conference on Cloud Computing Technology and Science. CloudCom'13
16. Leimeister JM (2010) Collective intelligence. Business and information systems engineering 2(4):245–248
17. Faraj S, Jarvenpaa SL, Majchrzak A (2011) Knowledge collaboration in online communities. Organization Science 22(5):1224–1239
18. Bulmer D, DiMauro V (2010) The New Symbiosis of Professional Networks: Social Media's Impact on Business and Decision-Making
19. Social Networks Popular Among Programmers. http://www.informationweek.com/wireless/social-networks-popular-among-programmers/d/d-id/1078472. Accessed 8/2015
20. Gitter: The chat for Github. http://gitter.im/. Accessed 8/2015
21. Bidding farewell to Google Code. http://google-opensource.blogspot.gr/2015/03/farewell-to-google-code.html. Accessed 8/2015.
22. Microsoft snubs Codeplex, moves big projects to GitHub. http://www.theregister.co.uk/2015/01/15/codeplex_repository_out_of_favour_as_microsoft_moves_major_projects_to_github/. Accessed 8/2015
23. StackOverflow. http://stackoverflow.com/. Accessed 8/2015
24. Vasilescu B, Filkov V, Serebrenik A (2013) Stackoverflow and github: Associations between software development and crowdsourced knowledge. In: Social Computing (SocialCom), 2013 International Conference On. pp 188–195
25. IBM Blue Mix. https://ace.ng.bluemix.net/. Accessed 8/2015
26. IBM Blue Mix for developers. https://developer.ibm.com/bluemix/.
27. IBM DevOps best practices. http://www.ibm.com/developerworks/devops/practices.html Accessed 8/2015
28. Stack Exchange API. https://api.stackexchange.com/docs.Accessed 8/2015
29. Sumbaly R, Kreps J, Shah S (2013) The BigData Ecosystem at LinkedIn
30. Chiu CM, Hsu MH, Wang ET (2006) Understanding knowledge sharing in virtual communities: An integration of social capital and social cognitive theories. Decis Support Syst 42(3):1872–1888
31. Chen IY (2007) The factors influencing members' continuance intentions in professional virtual communities—a longitudinal study. J Inf Sci 33(4):451–467
32. Preece J, Shneiderman B (2009) The reader-to-leader framework: Motivating technology-mediated social participation. AIS Trans Hum Comput Interaction 1(1):13–32
33. Lueninghoener C (2011) Getting started with configuration management ;login: 36(2):12–17
34. Bcfg2. http://www.bcfg2.org/. Accessed 8/2015

35. CFEngine. http://www.cfengine.com/. Accessed 8/2015
36. Tsalolikhin A (2010) Configuration management summit ;login: 35(5):104–105
37. Delaet T, Joosen W, Vanbrabant B (2010) A survey of system configuration tools. In: Proceedings of the 24th International Conference on Large Installation System Administration. LISA'10. pp 1–8
38. Fowler M, Foemmel M (2006) Continuous integration. Thought-Works). http://www.thoughtworks.com/ContinuousIntegration.pdf
39. CI T. https://travis-ci.org/. Accessed 8/2015
40. Jenkins A. e. o. s. c. i. s. https://jenkins-ci.org/. Accessed 8/2015
41. Breitgand D, Epstein A (2011) SLA-aware placement of multi-virtual machine elastic services in compute clouds. In: 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011). pp 161–168
42. Lucas-Simarro JL, Moreno-Vozmediano R, Montero R, Llorente IM (2013) Scheduling strategies for optimal service deployment across multiple clouds. Future Generation Comput Syst 29(6)
43. Li W, Svard P, Tordsson J, Elmroth E (2013) Cost-optimal cloud service placement under dynamic pricing schemes. In: 6th IEEE/ACM International Conference on Utility and Cloud Computing, Dresden, Germany
44. Ardagna D, Casale G, Ciavotta M, Pérez JF, Wang W (2014) Quality-of-service in cloud computing: modeling techniques and their applications. SpringerOpen J Internet Serv Appl 5(11)
45. Eclipse Connected Data Objects. https://eclipse.org/cdo/. Accessed 8/2015
46. Rossini A, Nikolov N, Romero D, Domaschka J, Kritikos K, Kirkham T, Solberg A D2.1.2 CloudML Implementation Documentation (First version). PaaSage Project. http://www.paasage.eu/images/documents/paasage_d2.1.2_final.pdf.Accessed 8/2015
47. Kritikos K, Korozi M, Kryza B, Kirkham T, Leonidis A, Magoutis K, Massonet P, Ntoa S, Papaioannou A, Papoulas C, Sheridan C, Zeginis C D4.1.1 Ű Prototype Metadata Database and Social Network. PaaSage Project. http://www.paasage.eu/images/documents/PaaSage-D4.1.1_final.pdf.Accessed 8/2015
48. Konstas I, Stathopoulos V, Jose JM (2009) On social networks and collaborative recommendation. In: Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '09. ACM, New York, NY, USA. pp 195–202. http://doi.acm.org/10.1145/1571941.1571977
49. Garrett JJ (2011) The Elements of User Experience : User-centered Design for the Web and Beyond. Voices that matter. Pearson Education, Berkeley, Calif. New Riders London
50. Nielsen J (2001) Converting Search into Navigation. http://www.nngroup.com/articles/search-navigation/. Retrieved October 6, 2014
51. Budiu R Search Is Not Enough: Synergy Between Navigation and Search. http://www.nngroup.com/articles/search-not-enough/. Accessed 8/2015
52. Ames M, Naaman M (2007) Why we tag: Motivations for annotation in mobile and online media. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '07. ACM, New York, NY, USA. pp 971–980. http://doi.acm.org/10.1145/1240624.1240772
53. Hughes J (2011) iPhone and iPad Apps Marketing: Secrets to Selling Your iPhone and iPad Apps. Que Publishing
54. Ho KK, See-To EW, Chiu GT (2013) How does a social network site fan page influence purchase intention of online shoppers: A qualitative analysis. Int J Soc Organ Dyn IT (IJSODIT) 3(4):19–42
55. Nielsen J (1993) Iterative user-interface design. Computer 26(11):32–41
56. Mayhew DJ (1999) The usability engineering lifecycle. In: CHI '99 Extended Abstracts on Human Factors in Computing Systems. CHI EA '99. ACM, New York, NY, USA. pp 147–148. http://doi.acm.org/10.1145/632716.632805
57. Ferreira J, Noble J, Biddle R (2007) Agile development iterations and ui design. In: Agile Conference (AGILE), 2007. IEEE. pp 50–58
58. Stone D, Jarrett C, Woodroffe M, Minocha S (2005) User Interface Design and Evaluation. Morgan Kaufmann
59. von Krogh G, Spaeth S, Lakhani KR (2003) Community, joining, and specialization in open source software innovation: a case study. Res Policy 32(7):1217–1241
60. Begel A, DeLine R, Zimmermann T (2010) Social media for software engineering. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research. ACM. pp 33–38
61. DiMicco J, Millen DR, Geyer W, Dugan C, Brownholtz B, Muller M (2008) Motivations for social networking at work. In: Proceedings of the 2008 ACM Conference on Computer Supported Cooperative Work. ACM. pp 711–720
62. Brusilovsky P, Kobsa A, Nejdl W (2007) The Adaptive Web: Methods and Strategies of Web Personalization, Vol. 4321. Springer
63. Tam KY, Ho SY (2005) Web personalization as a persuasion strategy: An elaboration likelihood model perspective. Inf Syst Res 16(3):271–291
64. Sundar SS, Marathe SS (2010) Personalization versus customization: The importance of agency, privacy, and power usage. Hum Commun Res 36(3):298–322
65. Nielsen J (1999) Site design. In: Designing Web Usability: The Practice of Simplicity. New Riders Publishing, Indianapolis. pp 166–167
66. Krug S (2006) The first step in recovery is admitting that the home page is beyond your control: Designing the home page. In: Don't Make Me Think!: a Common Sense Approach to Web Usability. New Riders, Berkeley, CA. pp 94–121
67. Swan K (2001) Virtual interaction: Design factors affecting student satisfaction and perceived learning in asynchronous online courses. Dist Educ 22(2):306–331
68. Lazar J, Preece J (2002) Social Considerations in Online Communities: Usability, Sociability, and Success Factors
69. Liu IF, Chen MC, Sun YS, Wible D, Kuo CH (2010) Extending the tam model to explore the factors that affect intention to use an online learning community. Comput Educ 54(2):600–610
70. Deterding S, Sicart M, Nacke L, O'Hara K, Dixon D (2011) Gamification. using game-design elements in non-gaming contexts. In: CHI11 Extended Abstracts on Human Factors in Computing Systems. ACM, New York, NY, USA. pp 2425–2428
71. Antin J, Churchill EF (2011) Badges in social media: A social psychological perspective. In: CHI 2011 Gamification Workshop Proceedings (Vancouver, BC, Canada, 2011)
72. Engine ESN. http://elgg.org/ Accessed 8/2015
73. Spurlock J (2013) Bootstrap: Responsive Web development. O'Reilly Media, Sebastopol, CA
74. Chef Documents: knife cookbook site. https://docs.chef.io/knife_cookbook_site.html Accessed 8/2015
75. Project Focus: Chef Supermarket - An Overview. http://gofullstack.com/chef-supermarket-an-overview/ Accessed 8/2015
76. Chef Supermarket - The New Community Site. https://www.chef.io/blog/2014/03/24/chef-supermarket-the-new-community-site/ Accessed 8/2015
77. Karniavoura F, Papaioannou A, Magoutis K (2015) C2C: An Automated Deployment Framework for Distributed Applications on Multi-Clouds. In: Proceedings of 9th Symposium and Summer School On Service-Oriented Computing, Hersonissos, Crete, Greece
78. JClouds. https://jclouds.apache.org/ Accessed 8/2015
79. Azure SDK. https://github.com/Azure/azure-sdk-for-java. Accessed 8/2015
80. Virzi RA, Sokolov JL, Karis D (1996) Usability problem identification using both low- and high-fidelity prototypes. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '96. ACM, New York, NY, USA. pp 236–243. http://doi.acm.org/10.1145/238386.238516
81. Catani MB, Biers DW (1998) Usability evaluation and prototype fidelity: Users and usability professionals. In: Proceedings of the Human Factors and Ergonomics Society Annual Meeting. SAGE Publications Vol. 42. pp 1331–1335
82. Jordan PW (1998) An Introduction to Usability. CRC Press
83. Brooke J (1996) Sus-a quick and dirty usability scale. Usability Eval Ind 189(194):4–7
84. Gillham B (2001) The nature of the interview. Research Interview (Real World Research) 6
85. Skype communications software. www.skype.com
86. SPEC jEnterprise2010 Benchmark. http://www.spec.org/jEnterprise2010/. Accessed 8/2015
87. Eclipse Graphical Modeling Project. http://www.eclipse.org/modeling/gmp/