Building a Social Platform Using FLOSS to Support Collaborative Communities: The ReWeee Case Study

Ioannis Routis(✉), Anargyros Tsadimas, and Mara Nikolaidou

Harokopio University of Athens, 70, El. Venizelou Str, 17671 Kallithea, Greece
{i.routis,tsadimas,mara}@hua.gr

Abstract. In this paper we present the development of a collaborative community using exclusively open source software. After the definition of the functional requirements of the project, we focus on finding specific software components to satisfy these requirements. The intention was to minimize the development effort and labor, relying on open source software. As a result, the platform was developed writing less than 10% of the required code and reusing more than 20 software components, not counting the software dependencies. The new components developed form our contribution to the community.

Keywords: Collaborative communities
Open source software development · Social networks
Component-based development

1 Introduction

Social networks have influenced the way that modern web applications are operating. A large number of them have adopted many characteristics of social networks, such as user profiles, real time notifications, instant messages, definition of users relationships, history of user actions, etc. Moreover, social network technology has been established as a prominent way of communication between members of an organization or enterprise [3]. Smart Communities as these are defined in [5], understand the potential of Internet technology, and make a conscious decision to adopt this technology to transform life and work in significant and positive ways [7]. Social software systems aim at the production of specific artifacts, thereby inviting users to participate in goal-oriented activities [4]. The user to user interaction is a major feature of collaborative communities.

In collaborative communities, leadership is decentralized and structured horizontally, a feature that makes communication on those groups more personal and more conversational than in other traditional groups. For that reason, in environments like those described above, the workflow and processes in general need to be highly adaptive and loosely structured in order to make ideas cross-fertilized and generate rich opportunities for innovation [6].

Such an adaptive process lies within the framework of the LIFE ReWeee Project, which aims to facilitate and promote Electrical and Electronic Equipment exchange and donation among households or households and public/private bodies so as to prevent the creation of Waste Electrical and Electronic Equipment (WEEE) [8].

Reuse of electrical and electronic equipment is among the top priorities in the EU waste hierarchy. In order to enhance the public perception towards the reuse of electric appliances and the prevention of WEEE generation, an initiative has been undertaken by a group of partners, which is implemented via the LIFE+ ReWeee project[1]. ReWeee Project includes a major action which is the development of a web-based collaborative platform for donating and exchanging Electrical and Electronic Equipment (EEE). That platform is used by households, companies and public services and its success lies within the social communication between volunteers and their collaboration in order to achieve the best possible result.

The challenge was to build the ReWeee platform using FLOSS and reuse as many software components as possible. Due to the previous experience of building applications with social characteristics, the development team decided to build the platform using the Django framework[2].

The remainder of this paper is organized as follows. In Sect. 2, the project requirements are presented alongside with the collaborative platform perspective. Section 3 highlights how FLOSS could satisfy the above presented platform requirements by not only using as-is or extending existing Django applications, but also by developing new ones. The final section refers to the conclusion that can be drawn for FLOSS adoption in web application development, while the contribution of this work is presented as well.

2 Project Requirements

In ReWeee Platform, users are categorized in three main types. These are guest users, registered users and administrators, that differentiate themselves as far as their granted permissions upon the use of the platform is concerned. More analytically, registered users are divided in two categories: a regular user, called civilian and an NGO user, indicating a representative of an Non Government Organization, while administrators include two different role types. The first one, the manager, which is responsible to validate a NGO through a provided official document (e.g., statute) and update the terms and conditions document for the platform and the second one the administrator whose responsibilities include user management, product categories management etc.

As far as the platform user authentication is concerned, when any unregistered user visits the web platform for the first time, he gets prompted to register into it, by creating a user account. The platform can authenticate users either as usual, namely, via an email and a password or from the most popular social

[1] ReWeee Project. https://www.reweee.gr/en.
[2] Django Project. https://www.djangoproject.com/.

networks (i.e., Twitter, Facebook and Google Plus). This account can be created by giving to the platform the necessary permissions for using personal user data. For NGOs there is a second level of validation when a new NGO user subscribes to the ReWeee platform.

After a successful registration, the, from now on, registered platform user, is able to submit an advertisement donating or exchanging an item, to declare interest for an existing product and propose an offer to acquire it, as well as to communicate with any other user who owns a desirable electric device. Products are categorized in a multilevel hierarchy which was provided by the committee of the ReWeee project. A product belongs to a lower level of hierarchy.

ReWeee platform enables messaging and notifications for its users so as to enhance communication and collaboration for product exchange. Namely, whenever user expresses interest on a product, the owner of the product is notified via email and through a notification inside the platform. In this product exchange process the involved role interaction can either be civilian-to-civilian and civilian-to-NGO.

Moreover, a registered user is not only able to search a product but also to has a profile with information about his/her location, products, and rating in which he can activate or deactivate an uploaded product. A registered platform user can also comment in any advertisement that he had made use of. That way, either the platform administrators or the appropriate users will be notified for either the category change proposal or the commenting in an advertisement.

Finally, the stakeholders of the ReWeee project would like to view some statistics about the products exchanged in specific periods of time, how many users have been registered and are active etc. These reports are produced at any time through the administration environment of the platform.

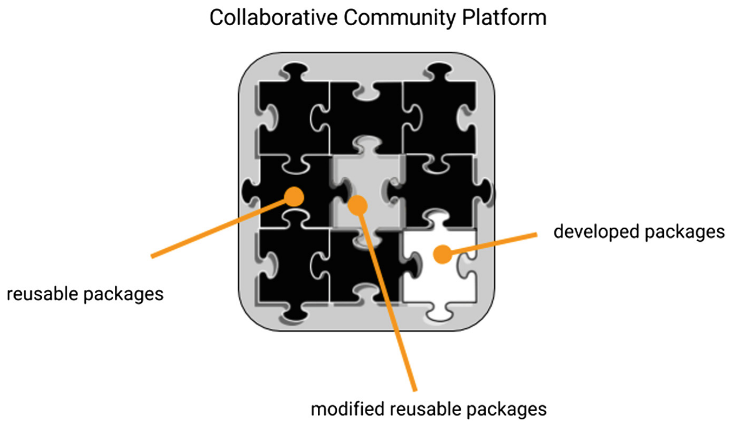


Fig. 1. Collaborative platform perspective

For the development of the collaborative platform and the satisfaction of its requirements described above, an implementation strategy should be adopted.

More specifically, as it is projected in Fig. 1, firstly, a set of already existing packages should be reused in order to reduce coding effort and testing. Secondly, some existing packages could undergone minor changes so as to adapt in project development requirements. Thirdly, several new packages should be developed for the satisfaction of requirements that the reuse of existing packages could not satisfy.

3 Satisfying Project Requirements with FLOSS

The collaborative platform architecture is illustrated in Fig. 2. The operating system of all servers is Debian Linux. The platform is a multi-layered application, where the user interface is based on bootstrap front-end framework (forming the django views), the data are kept in a Postgres database and the business logic is implemented using django framework. A number of reusable software components, called django packages or django applications were used in order to provide the required functionality. Users management is relying on python-social-auth and registration applications, while users interaction/communication is based on django-messages and notification applications. The products definition and exchange mechanism is based on the product application, which was developed because there was not available any related application. Table 1 summarizes the reused django applications and the reusability level of each one. Moreover, some new django packages were developed to support the complete set of functional requirements.

3.1 Django Framework

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It is based on Python programming language. Django follows the model-view-template (MVT) architectural pattern[3]. It is distributed under BSD license. Django-packages[4] is a directory of reusable django applications.

3.2 Extending Django User Model

To support the required user roles, the default django user model was extended. A UserProfile model class holds the common attributes of the users such as the display name and two specific classes, extending the UserProfile class, namely CivilianProfile and NGOProfile are defined for the specific user roles of the community.

[3] The Model-View-Controller design pattern. https://djangobook.com/model-view-controller-design-pattern/.
[4] Django-packages. https://djangopackages.org/.

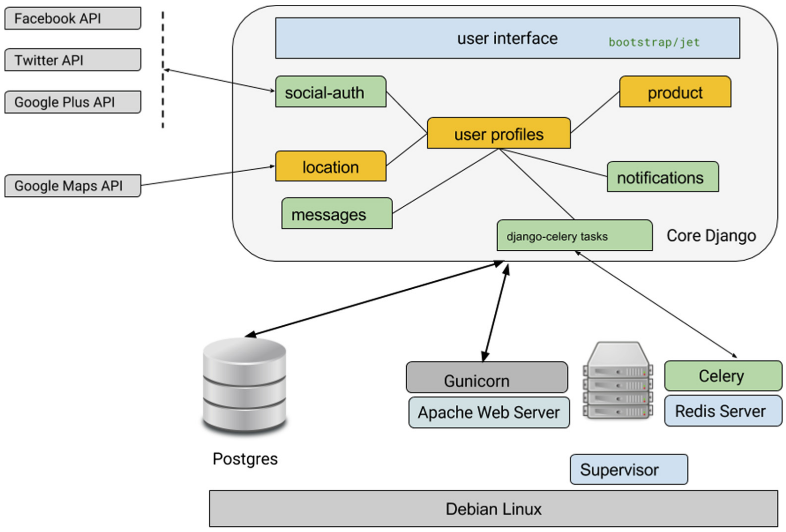


Fig. 2. ReWeee platform architecture

3.3 Supporting Periodic Tasks

To satisfy specific functional requirements, such as constraints about the amount of products that are exchanged between the users, it was necessary to periodically check the amount of products exchanged per user. Moreover, for optimal process completion control, it is required to check whether an exchange is completed on time.

More specifically, reminders are being sent to user for completing a product exchange/donation on time. An asynchronous task manager software could satisfy these requirements. Celery[5] is an asynchronous task queue/job queue based on distributed message passing . It is focused on real-time operation, but supports scheduling as well. Due to the fact that it is written in Python, it is easy to integrate with django. Celery uses "brokers" to pass messages between a Django Project and the Celery workers. Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. Redis can serve as the message broker for the Celery. Supervisor[6] was used in order to run the celery worker and scheduler as daemon processes.

3.4 Reusable Applications with Minor Changes

Although the plenty of the reusable django application were exploited, in two specific cases there was a need to modify some of them to feet the requirements of the platform. The first case is the notification mechanism, where the html

5 Celery. http://www.celeryproject.org/.
6 Supervisor. A process control system. http://supervisord.org/.

templates of the application needed an extension to support all the attributes of the exchange events. The second case was the registration application, where a new step of verification was injected at the registration process.

Notifications. Notifications are a major characteristic of social networks. For that reason, in the case of our platform the notifications application was widely used as the whole task flow was mainly driven by the interchange of notifications and actions from the platform users. More analytically, notifications occurred whenever a user expressed interest on a product, when an offer was accepted/rejected or even when a user confirms or aborts an agreement about an exchange. That way a notification event was triggered for another platform user which could lead to another platform activity. To support this functionality, the django-notifications package (see Table 1) was extended.

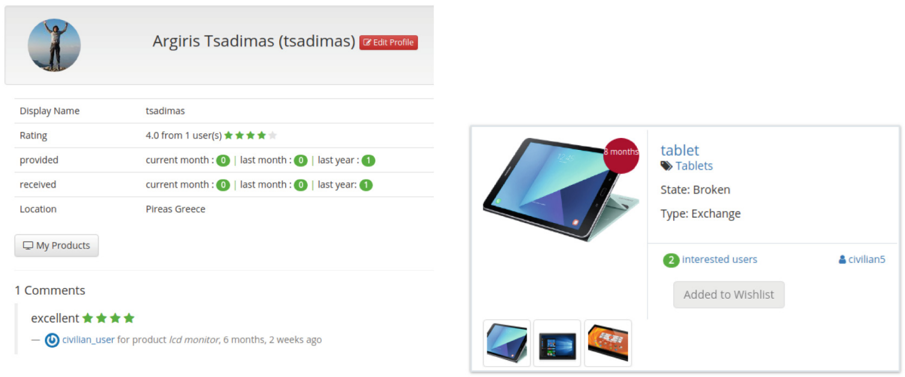
Table 1. Reused Django applications

Operation	Library name	Reusability
Messaging	django-messages	as is
Notifications	django-notifications	modified
Categories	django-categories	as is
Import/Export	django-import-export	as is
Rating	django-star-ratings	as is
Autocomplete	django-autocomplete-light	as is
Upload multiple files	django-multiupload	as is
Avatar	django-avatar	as is
Pagination	django-endless-pagination	as is
Captcha	django-recaptcha	as is
Terms and conditions	django-termsandconditions	as is
Cookies disclaimer	django-cookie-law	as is
Contact form	django-envelope	as is
Notify on model changes	django-fieldsignals	as is
Bootstrap forms	django-forms-bootstrap	as is
Registration	django-registration	modified
Social network authentication	python-social-auth	as is
Model translation	django-modeltranslation	as is
Admin ui	django-jet	as is
Task manager	celery	as is
Message broker	django-redis	as is

Registration. Due to the platform user requirement analysis and the categorization of roles that are involved, an extension was made to the default Django Registration application. More specifically, on the one hand, for the NGO user, an advanced authentication method was implemented as it was mentioned above. Registration process was the same as with the plain users but its complexity involved, a formal document inspection (organization statute) from one of the administrator roles, the Platform Manager. The NGO user was inactive unless its statute was verified. On the verification, an email informs the user of the successful registration. On the other hand, for all the user types, it was planned to provide the functionality of signing in the platform using a social network choosing from the major ones (i.e. Facebook, Twitter, Google Plus). For the implementation of this platform feature, the appropriate APIs were used. During the authentication on each social network, the user defined which information platform could use (email, username, etc.).

3.5 Developed Applications

User Profiles. A civilian has a private profile (Fig. 3a), which is accessible only from logged in users. An NGO user has a public profile, where basic info are presented such as the name of the organization, the location (only area/city, no specific location), the document that certifies the fact that this organization is an NGO. For all user profiles rating information, along with comments from other users are presented. Moreover some statistics about the products that this individual has received and provided. Moreover a products list link and a send message link are provided.



Argiris Tsadimas (tsadimas) Edit Profile

Display Name	tsadimas
Rating	4.0 from 1 user(s) ★★★★
provided	current month : 0 | last month : 0 | last year : 1
received	current month : 0 | last month : 0 | last year: 1
Location	Pireas Greece

My Products

1 Comments

excellent ★★★★

— civilian_user for product *lcd monitor*, 6 months, 2 weeks ago

8 months

tablet
Tablets
State: Broken
Type: Exchange

2 interested users civilian5

Added to Wishlist

(a) A user profile page (b) A product in a leaf category

Fig. 3. A user profile and product view

Product. A user can create products that he/she would like to exchange. For each product a name, a state, a description and some images can be used to characterize the product (Fig. 3b). Also the date that the product was added is kept. Moreover, a user can express interest on acquiring a product. Upon each interest declaration a user can define if he/she wants to provide some of his/her products to support the specific exchange. Every time a uses expressed interest on a product, the owner of the product is notified via email and by a notification inside the platform. Here he/she can review the details of the exchange and he/she can accept or decline the exchange. On the acceptance, the users are physically exchanging the products and one last step is that the user that acquires the product(s) is responsible to verify the exchange and has the ability to rate the acquired products.

Products are categorized to leaf categories and a user can check the number of products in each category hierarchy, as shown in Fig. 4. To effectively categorize the products, the django-categories application was exploited. Django-categories relies on django-mptt package[7] [1], which provides utilities for implementing Modified Preorder Tree Traversal with Django Models and working with trees of Model instances.

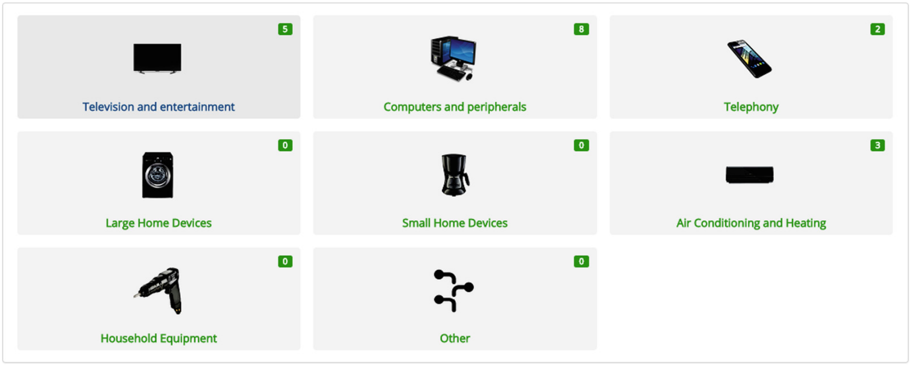
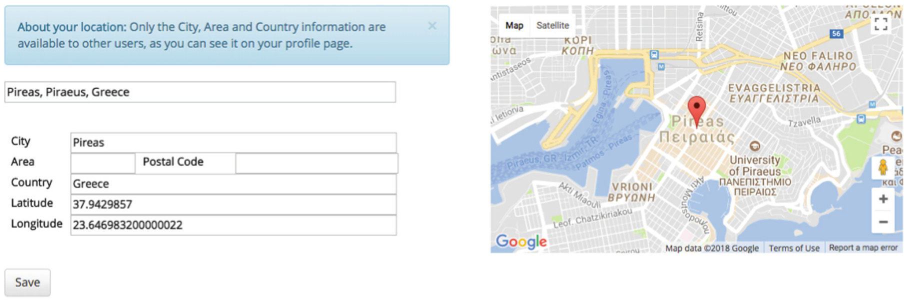

5
Television and entertainment
8
Computers and peripherals
2
Telephony
0
Large Home Devices
0
Small Home Devices
3
Air Conditioning and Heating
0
Household Equipment
0
Other

Fig. 4. Products categories

Location. The need for the development of a new django application for location is based on that there was not available a simple, reusable application where a user could select a location using the Google Maps API and store this location to user profile. Figure 5 presents the form where a user declares his/her area. A developer can define specific attributes that are maintained in the datastore (for example he/she could store only the area and the city, not a specific address).

[7] django-mptt. Modified preorder tree traversal with django models. https://github.com/django-mptt/django-mptt/.



About your location: Only the City, Area and Country information are available to other users, as you can see it on your profile page.

Pireas, Piraeus, Greece

City	Pireas		
Area		Postal Code	
Country	Greece		
Latitude	37.9429857		
Longitude	23.646983200000022		

Save

Fig. 5. User location definition

3.6 Discussion

What was obtained throughout this development process was mainly the inspiration not only to reuse but also to contribute to the FLOSS community. Using existing software components from other developers promotes the interoperability and applicability of software components as the result of minor modifications ending in code refactoring and optimization. As an observation, in our case, the development phase was decreased by around 70%, since the developed -from scratch- code represented less than 10% of the total project. The remainder of the development phase was used to test and integrate the adopted code, to make the appropriate modifications and integrate the feedback of the final users in order to ensure the platform acceptance. Obviously, the developed software components are planned to return to the open source community as our contribution in building a collaborative community. The ReWeee platform is currently in internal testing mode from the participants. We are currently refactoring the code and developing integration tests in order to ensure the interoperability with the Django underlying environment.

4 Conclusions

Open source software developers reuse code because they want to integrate functionality quickly, because they operate under limited resources in terms of time and skills, and because they can mitigate development costs through code reuse [2]. Although there is a large number of reusable software components available, the reusability, the customization and the development of new components is not a straightforward process. The documentation, the code quality and the tests included in a software repository are some of the criteria that a developer could rely on in order to select the appropriate software.

However, using existing software, it helps the maintainability of the software, software bugs are resolved and refactoring is performed in order optimize the performance and the security. Our contribution was two-fold, as it primarily aimed to minimizing labor through the reuse of existing work and secondly to be set as a contribution of "FLOSS on top of FLOSS", namely to create reusable code and libraries which would be returned back to the FLOSS community.

Acknowledgment. This work is co-funded by the European Commission through the LIFE+ Funding program, LIFE14 ENV/GR/000858: LIFE REWEEE.

References

1. Modified preorder tree traversal with django models. https://github.com/django-mptt/django-mptt/. Accessed 29 Jan 2018
2. Haefliger, S., Von Krogh, G., Spaeth, S.: Code reuse in open source software. Manage. Sci. **54**(1), 180–193 (2008)
3. Hatzi, O., Meletakis, G., Katsivelis, P., Kapouranis, A., Nikolaidou, M., Anagnostopoulos, D.: Extending the social network interaction model to facilitate collaboration through service provision. In: Bider, I., Gaaloul, K., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P. (eds.) BPMDS/EMMSAD -2014. LNBIP, vol. 175, pp. 94–108. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-43745-2_7
4. Johannesson, P., Andersson, B., Wohed, P.: Business process management with social software systems – a new paradigm for work organisation. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008. LNBIP, vol. 17, pp. 659–665. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00328-8_66
5. Lindskog, H.: Smart communities initiatives. In: Proceedings of the 3rd ISOneWorld Conference, vol. 16 (2004)
6. London, S.: Building collaborative communities. On Collaboration. Tate, London (2012)
7. Meletakis, G., Hatzi, R., Katsivelis, P., Nikolaidou, M., Anagnostopoulos, D., Anastasiou, C.A., Karfopoulou, E., Yannakoulia, M.: MedWeight smart community: a social approach. In: Ismail, L., Zhang, L. (eds.) Information Innovation Technology in Smart Cities, pp. 151–162. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-1741-4_11
8. Routis, I., Nikolaidou, M., Anagnostopoulos, D.: Using CMMN to model social processes. In: Teniente, E., Weidlich, M. (eds.) BPM 2017. LNBIP, vol. 308, pp. 335–347. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74030-0_25