symmetry

MDPI

Article

SCPL: A Social Cooperative Programming Language to Automate Cooperative Processes in (A)Symmetric Social Networks

José M. Conejero *, Fernando Sánchez-Figueroa, Roberto Rodríguez-Echeverría and Juan Carlos Preciado

Quercus Software Engineering Group, Universidad de Extremadura, Cáceres 10003, Spain; fernando@unex.es (F.S.-F.); rre@unex.es (R.R.-E.); jcpreciado@unex.es (J.C.P.)

* Correspondence: chemacm@unex.es; Tel.: +34-927-257-100

Academic Editor: Yuhua Luo

Received: 7 April 2016; Accepted: 20 July 2016; Published: 28 July 2016

Abstract: In recent years, the increasing use of social networks and applications—especially those used in an asymmetric way—has significantly changed the business processes in many organizations. These applications provide new cooperative ways of performing these processes by taking advantage of the interactions among users. However, the high number of these applications has led to a lack of automation in their interactions and, thus, the need of manually connecting to these networks to perform recurrent and repetitive tasks. In order to automate these operations, this paper presents SCPL, a Domain Specific Language (DSL) that enables connectivity among different social networks and applications and provides a way to automate their management. The main contribution of this paper is showing how SCPL can be used to specify collaborative tasks using symmetric and asymmetric social networks in a transparent way.

Keywords: cooperative processes; symmetric and asymmetric social networks; domain specific languages

1. Introduction

Today, the increasing use of social networks and applications has significantly changed the business processes in many organizations. These applications include traditional social networks (e.g., Facebook, Twitter, etc.) but also social applications (e.g., Google Drive, Dropbox, Doodle, etc.) Among the benefits provided by these tools, it is worth mentioning the new cooperative ways of performing business processes that they have introduced, for example, offering support, answering questions, voting on the quality of a service, commenting, following the progress of a service request, networking with other colleagues to share experiences, etc. The appearance of techniques such as Cloud Computing, which simplifies data storage, processing, and security, is accelerating their adoption.

Broadly speaking, social networks can be categorized into symmetric and asymmetric [1,2]. While the former is represented by those networks that require confirmation by a user in order to allow others to follow her and see her updates, the latter refers to social networks that do not require this confirmation. Examples of symmetric social networks are Facebook or Linkedin, whilst Twitter or Instagram belong to the asymmetric category. Both kinds of social networks are being more and more used in the industry (e.g., the health domain [3]), although they are more usually used in an asymmetric manner (as a way of sharing information with their clients/users and also obtaining some feedback from them [4]).

Independent from their use (symmetric or asymmetric), a single business process usually involves interactions with more than one social network. Although there are tools aiming at the coordination of different tasks in different social applications, their scope is quite limited, mainly due

to them not having enough expressive power: either they have simple programming structures (if–condition–then–action) or they do not have the possibility of setting advanced conditions (i.e., counting the number of "Like" operations on a specific post). Moreover, these tools are based on graphical notations and templates containing the rules to be applied, which makes it impossible for the user to add new rules not previously defined in the templates.

This paper presents SCPL (Social Cooperative Programming Language), a Domain Specific Language (DSL) [5] that enables connectivity among different social networks and applications and provides a way to automate their management. This DSL relies on the definition of a textual language with which the interconnections among different social networks may be defined. In that sense, the main contribution of this paper is showing how the company business processes may be improved by integrating cooperative and collaborative tasks provided by both symmetric and asymmetric social applications in a transparent and flexible way.

Although DSLs have been previously used with social networks, they have been used mainly in an asymmetric way for querying operations [6] or monitoring and simulating activities [7]. To the best of our knowledge, the utilization of a DSL to coordinate both symmetric and asymmetric flows in social applications has not been considered before.

The rest of the paper is as follows: Section 2 introduces a motivating example. Section 3 presents an analysis of the problems that arise when using traditional tools to manage social networks and the properties that a tool to solve these problems should exhibit. Section 4 presents an analysis of tools with similar aims to SCPL. Section 5 presents the social cooperative language, Section 6 provides details on the implementation of the approach, Section 7 presents an evaluation of the approach, Section 8 discusses some threats to validity, and Section 9 concludes the paper.

2. Motivating Example

This section presents a running example that will drive the explanation of both the comparison with other tools and the approach. This case study represents an actual situation in the political domain where decisions must be quickly made (although the example is also applicable to many other domains). Let us consider the process to publish a press release by a political party, where the cooperation of different groups of people belonging to that political party is required. Let us consider the following steps in the process:

(1) The press manager of the organization writes the press release and sends it by email to the organization managers in order to be reviewed.
(2) The press manager waits for at least three confirmation answers from the managers.
(3) Then, the press release is published into a LinkedIn group where the organization has a set of members. These members belong to a working group that is responsible for identifying possible errors in the organization documents. If the members agree on the press release, they will use the "Like" option of the social application in order to show their approval.
(4) When the press release has 30 "Likes", the press manager considers that the press release is accepted to be published.
(5) Finally, it is submitted to the media and it is also published through other social networks like Twitter or Facebook.

Note that, to follow this process, the press manager must perform several tasks in different applications, such as checking the email, reviewing the publication in LinkedIn, waiting for the expected number of "Like", sending the press release to the media, and using the social networks selected to publish the release.

3. Properties Elicitation

In order to identify the main problems that could arise when performing the process described in the previous section by using the traditional tools, we contacted several social media professionals

by e-mail who were working in different kinds of companies or organizations and had at least 5 years of experience. As reference, we used the professional directory of the Spanish Association of Social Media Professionals and Community Managers [8], which lists around 200 Spanish experts. After a difficult convincing process, a group of ten community managers (CMs) in charge of managing the social media campaigns of different kinds of organizations from different domains (politics, health, or press) finally agreed to take part in our study.

Once the study population was obtained, we sent them the previous example and asked them to implement the process by using the social networks and tools that they usually utilize, including those tools that allow the interaction with different networks as a hub (e.g., IFTTT [9] or Zapier [10]). Together with the example, the respondents received a questionnaire about the realization of the process. Its main goal was to determine the main factors hindering each task. The questionnaire was defined as a transversal study [11] with open questions in order to let the community managers describe the problems that they found. The questionnaire was electronically auto-managed by means of web forms.

Additionally, to test possible problems related to software engineering concepts such as reusability or extensibility of the process, we also provided the community managers with two more processes (similar to the press release one) but focused on different domains. Concretely, we used adapted versions of: (i) an example of a health domain process to establish a treatment for patients by using professional networks [3]; and (ii) a process for thesis approval in a university context [12]. These two processes were selected because of the similarities of some steps contained in them with those involved by the motivating example. Thus, the reusability degree for recurrent tasks could also be evaluated in the different tools.

Then, based on the results obtained in these questionnaires, we observed that the main problems detected by the community managers could be summarized in four main points:

(1) Lack of reusability of the process. The answers provided by the community managers revealed that when similar processes are performed, some tasks have to be carried out over and over again.
(2) Continuous context swapping. Since the definition of the process sometimes involves different tools and social networks, the users evidenced the need for frequently swapping between different contexts.
(3) Lack of expressivity in the tools. Note that, sometimes, single operations such as "if–condition–then–action" (used by tools with hubs for social networks) are not enough to express the process being defined. In that sense, we observed that there was a need for advanced programming control structures which allow the user interaction to be taken into account.
(4) Lack of a unified user interface. This problem leads to the lack of reusability and continuous context swapping previously mentioned. Precisely, this last issue was the motivation for the work presented here, where a unified tool is presented to define and manage reusable processes among different social networks.

Based on the results of the previous questionnaire, we built a second one, but in this case with closed questions. The questionnaire was sent to the CMs with the aim of obtaining the main properties or features that tools used to manage these kinds of social processes (by using social networks) should exhibit.

Basically, the questionnaire was formed by a collection of uniform questions. Each question was presented according to the following structure:

- Property Category. The properties under evaluation were organized into three different categories: user experience, solution adequacy, and quality.
- Property Name. We used software engineering standard terminology for property names [13].
- Property Description. We clearly explained the meaning of every property to users in order to keep a consistent interpretation of the question among users.

- Property Illustrative Example. Some positive and negative examples were provided to avoid common pitfalls.
- Property Rank. Every user had to sort the priorities according to their preferences and experiences with the examples, giving a unique value between 1 and 20.
- Property Rationale. Users had to explain why they gave that rank (position) to this property. As a result, we obtained qualitative data about users' priorities which (1) gave additional information to consider results that overlap and (2) allowed us to assess that users had properly understood the question.

The questionnaire was formed by 20 questions, asking about 20 different software solution properties [13]. To reduce possible biased results by the order of the questions, questions were shuffled for every participant. These properties are listed and organized by the aforementioned categories as follows:

- User experience: user familiarity (P01), user diversity (P02), learnability (P03), consistency (P04), minimal surprise, recoverability, and user guidance
- Solution adequacy: sufficiency (completeness) (P05), primitiveness, abstraction (P06), modularization, encapsulation, tool integration.
- Solution quality: maintainability, portability, testability, performance, security, reusability (P07), and modifiability.

Based on the results obtained in this questionnaire, we selected the features that got a position in the top third. Table 1 shows these features and a code used for further references.

Table 1. Properties with higher scores according to the community managers (CMs).

Category	Property	Code
User Experience	User familiarity	P01
	User diversity	P02
	Learnability	P03
	Consistency	P04
Solution Adequacy	Sufficiency	P05
	Abstraction	P06
Solution Quality	Reusability	P07

As expected, the features that the respondents considered of utmost importance were those related to the problems mentioned in the previous section, so that a tool that fulfils these characteristics could conceivably solve those problems. Following, the selected properties are described at a more detailed level:

- **User experience**. Inside this category, CMs selected the following four properties:
    - **User familiarity (P01).** The interface should use terms and concepts drawn from the experiences of the people who will use the software.
    - **User diversity (P02).** The interface should provide appropriate interaction mechanisms for diverse types of users. In this case, most of the enquired users highlighted the necessity of distinguishing between experienced users with complex tasks (textual interface—P02.1) and new users or simpler tasks (graphical interface—P02.2).
    - **Learnability (P03).** The solution should be easy to learn so that the user can rapidly start working with it.

- **Consistency (P04).** The interface should be consistent so that comparable operations are activated in the same way. In this case, it refers to the ability to permit users to keep focused on a concrete task despite some context swapping that may occur because several social networks are involved.

- **Solution adequacy**. Inside this category, CMs selected the following two properties:

  - **Sufficiency (P05).** Achieving sufficiency and completeness means ensuring that a solution captures all the important characteristics of an abstraction and nothing more. The solution should give support for "advanced" control structures and allow users to define complex actions and conditions. Note that from the computational viewpoint, simple branching structures ("If–condition–then–action") are not expressive enough to solve complex problems.
  - **Abstraction (P06).** It refers to both the process and result of generalisation by reducing the information of a concept, a problem, or an observable phenomenon so that one can focus on the "big picture". In our case they could be filters that refer to the possibility of limiting a problem including conditions and requirements, such as the number of "Likes" for a post.

- **Solution quality**. Inside this category, CMs selected just one property:

  - **Reusability (P07).** This feature refers to the capability of the solution for reusing a whole process or some of the steps defined within the process.

4. Related Work

This section shows similar approaches or tools that may be found in the literature or Internet to coordinate tasks through several social applications. We distinguish between professional and academic solutions.

4.1. Professional Solutions

A group of widespread solutions are characterized attending to the features that were selected in the previous section. To this purpose, 18 students from our university were selected to test those tools and evaluate their features, specifically focusing on the aforementioned ones. The rationale behind using students for this task was twofold: (1) we wanted to avoid any bias based on previous knowledge of the participants about the purpose of the study or the tools under evaluation; and (2) after being trained, students could perform this task properly, so there was no need to bother busy professionals for a relatively simple activity. The students selected were all enrolled in a Masters in Entrepreneurship and Innovation offered by our university. The Masters program was divided into two main areas: on one hand, a set of subjects covered the technical issues regarding launching a technology-based business (especially those related to the Internet); on the other hand, a second set of subjects was focused on business matters in order to provide the students with the management and communication skills needed to start these kinds of businesses, including courses about social media and digital marketing. The students selected for our study had previously finished a Computer Science bachelor degree (note that a degree is required to enroll in a Masters program in Spain) so they were already holding a strong technical profile.

To characterize the tools, the students were firstly trained in their use and, then, they were grouped into teams of two members so that each member tested four different tools. To accomplish the comparison, we selected the same three processes used by CMs to identify the features.

At the end of the process, each tool had been tested by nine different students. The students were also asked to collaboratively build a table where they indicated the features that each tool met. The result of this study may be observed in Table 2. Note that this table only includes those approaches that provide tools freely available to be tested. Other approaches are considered later in next section.

Table 2. Comparison of functionalities of the analyzed tools.

Tool	P01	P02.1	P02.2	P03	P04	P05	P06	P07
IFTTT [9]	☑		☑	☑				
Zapier [10]	☑		☑	☑				
We Wired Web [14]	☑		☑	☑				
Elastic.io [15]	☑		☑	☑				
Cloud Work [16]	☑		☑	☑				
WappWolf [17]	☑		☑	☑	☑			
itDuzzit [18]		☑	☑			☑	☑	
Kissflow [19]			☑	☑	☑		☑	

After analyzing the existing tools, it could be observed that most of the tools centered their efforts on providing a simple, easy to learn, and intuitive graphical user interface (P01, P03, and P02.2). The main problems found by students were related to sufficiency and abstraction (P05, P06), reusability (P07), context changing (P04), and textual representation of the process (P02.1). Although some of these tools (e.g., IFTTT and Zapier) allow reusing a recipe (or a zap) by different people, they always connect the same pair of social networks. The meaning given to reusability here is when the same process can be reused in different social networks and by different users.

4.2. Academic Solutions

There are other works in the literature presenting ideas for coordinating tasks among different social networks. In [20], Simple Flow is presented, a process based approach that cannot be considered a DSL but a domain-specific mashup. Simple Flow targets end-users and programmers with no experience in programming for social networks, giving them the possibility to design processes by concatenating social network actions (like posting a message or commenting a photo). As far as we know, Simple Flow does not provide a textual representation of the process. In [21], Jabberwocky is presented as a programming environment in which they use a syntax similar to a structured query language to interrogate social networks and define simple tasks.

Similarly, the work in [12,22] proposed the introduction of social instructions in a BPM notation [23], but without using a DSL. Although DSLs have been used with social networks, they have been used mainly for querying operations [6] or monitoring and simulating activities [7]. To the best of our knowledge, the utilization of a DSL to coordinate different social applications has not been considered before.

Next we show the main ideas behind SCPL, conceived to solve most of the problems highlighted in Table 2.

5. The Social and Cooperative Programming Language

As mentioned previously, SCPL is a language that allows the definition of rules that connect different social applications. This language has been defined as a grammar based on XText [24]. Before showing details of its implementation, the main characteristics of SCPL are described, all of them illustrated by simple examples.

Easy to use: the language used by the DSL is very close to natural language, so that the definition of rules is very intuitive for the user. Although it is not as friendly as a graphical interface, it is friendly enough for those users that are familiar with social networks. In particular, Figure 1 represents the fragment that solves the problem presented in the motivation example (publishing a cooperative press release).

```
1  send Email (directives@googlegroups.com) body "Press release" = $1
2    wait response('Ok' , 3) then
3       send Linkedin (group "Affiliated and members") body $1
4          wait Ilikes(30) then
5             send Email(mediaA@unex.es & mediaB@unex.es) body $1
6             send Twitter body "Press release" = $1
```

Figure 1. Case study.

Social network configurability: the user may select the social networks that she will use at development time (observe lines 3 and 6 in Figure 1). However, she also has the chance of deciding the social networks to use at instantiation time (just before the execution of the final application, Figure 2). That allows the rule to be reused for different cases.

```
1  send SocialNetwork body "Press release" = $1
```

Figure 2. Sending to a generic social network.

Social Apps: Besides the traditional social networks, other social applications may be used in the language—e.g., Google cloud, Dropbox, Doodle pools, and so on (Figure 3).

```
1  send Email ('user@scpl.es') body "Press release" = $1
2  send GoogleSpreadsheet("/myTable") body $msg1
```

Figure 3. Sending to social applications.

Global Vars: In order to avoid repeating the content of the messages in different rules, global vars are used. These vars are defined in the language with the $ symbol (see line 1 in Figure 1). Global vars also help at avoiding context switching and solves the problem of information reusability and repetition.

Sending public, private, or group messages: Obviously, one of the main functionalities of current social networks is to allow the publication of messages in timelines, pages, groups, or as private. These functionalities are also contemplated in the proposed language; see lines 3 and 6 in Figure 1. Sending a private message may be done as described in Figure 4.

```
1  send Twitter (@scpl) body "This is a private message"
2  send Linkedin("David Martín Rodríguez") body "Private message"
```

Figure 4. Sending private messages.

Event programming: Among the functionalities offered by the language, it is worth mentioning the possibility of programming and, thus, automating events. Based on this functionality, the user does not need to wait for an event end in order to start another one. The language allows the user to define the conditions that make the event to be executed.

By using the right expression, the language allows that the execution waits for one or more events before resuming the execution. This is done by using the wait clause, as shown in lines 2 and 4 in Figure 1.

Identifying and counting "Like". One of the main innovations of social networks is the "Like" concept. This concept is currently used in many other domains and applications as a way to approve or support an idea. Unlike the rest of applications analyzed, SCPL supports using, counting, and managing the "Like". As an example, line 4 in Figure 1 shows how the number of "Like" may be counted and used as a condition for a publication.

Reply identification. The automation of the replying process is also important. Note that if any publication or mail is not valid, the reply must avoid context switching and must allow working in the

same conversation flow. Although a whole analysis of the reply text is out of the scope of this work, a simple analysis of some word in the reply may be performed, as is represented in line 2 of Figure 1.

Replying to messages. An automatic reply to the messages received in the social networks may also be specified (similar to the automatic reply identification). This automatic reply may be applied to both public and private messages or, even to received emails (see Figure 5).

```
1  wait Email ('user@scpl.es') then reply "Thanks for your answer"
```

Figure 5. Replying to messages.

6. Implementation

Once the language and its main characteristics have been explained, this section presents how it has been implemented. Figure 6 represents the workflow that the process follows to produce the final applications that allow the interaction with the social networks. The process takes as input the source code that a programmer writes in SCPL (according to the grammar). This source code is used by Xtend [25] to be transformed into a simple and serializable structure, based on YAML [26]. The resulting document is then processed by a parser (developed by using Ruby [27]) that interprets and executes the sentences, producing the interaction with the social application APIs or with the users if needed (e.g., in order to introduce the authentication data). Next, all these steps are described in more detail.

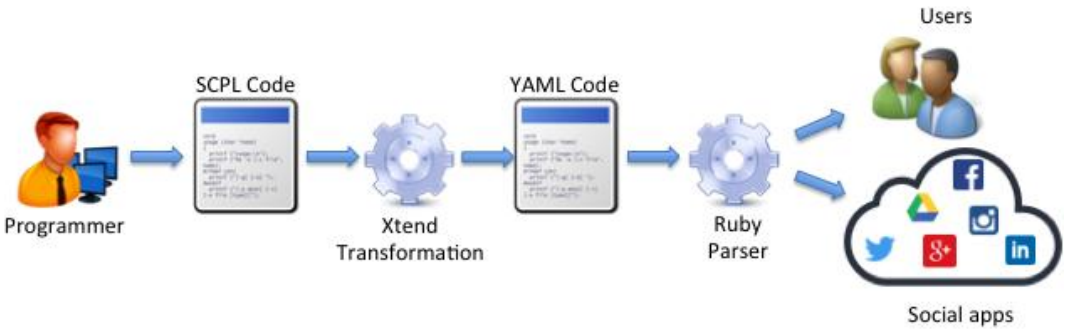

Figure 6. Tool architecture and execution flow. SCPL: Social Cooperative Programming Language.

6.1. Xtend Transformation from SCPL to YAML

Xtend is a programming language that eases the definition of transformations to interpret grammars generating a different output. Xtend uses a syntax similar to that used by Java although improving some of its aspects. In this case, Xtend has been used to generate a YAML representation of the SCPL specification. YAML is a serializable language that allows representing information in a similar format to that used by XML. YAML is used since its representation will be later used by an interpreter to generate the final code of the application.

Xtend allows the processing of the grammar as a tree so that it defines a Domain Model element that represents the grammar root, and this element contains a set of children (they may also have other children) that represent the keywords in the grammar. Each element may contain attributes that represent the values defined in the different keywords (content introduced by the user). Figure 7 shows a class specified in Xtend to define the transformation to YAML of the send structure in the grammar.

```
class YamlModelHelper {
    boolean to=false;
    boolean firstto=false;
    int send=0;

    def dispatch CharSequence toYaml(EObject m)
    '''    «IF m.eCrossReferences.size + m.eContents.size > 0»
                «FOR ref : m.eCrossReferences»
                    «ref.eContainingFeature.name» -> «ref.toYamlSig»
                «ENDFOR»
                «FOR child : m.eContents»
                    «IF child.eClass.name.equals("Send")»
                        «child.toYamlSend»
                        «child.toYamlSendDisp»
                    «ELSE»
                    «ENDIF»«restartFirstTo»
«ENDFOR»
«ENDIF»'''
```

Figure 7. Xtend class for generating the YAML for the send structure.

6.2. Ruby Interpreter Definition

Ruby has been used for the definition of the interpreter of YAML files. Ruby was selected since it provides an important set of gems (like libraries that may be imported to a project) that may be used to easily connect to external APIs like those provided by the social applications.

The interpreter is responsible of analyzing the YAML file line by line and executes them. Figure 8 shows an excerpt of the interpreter source code defined to interpret the YAML code generated for the send structure.

```
if ins[:type] == 'send' then
        if ins[:send] == 'Gmail' then
                subjt = ins[:body].split(" %/n ")
                gmail.deliver do
                  to ins[:to]
                  subject subjt[0]
                  text_part do
                    body ins[:body].sub(subjt[0],'').gsub('%twDot%',':').gsub(" %/n ", '

                  end
                end
```

Figure 8. Excerpt of the source code of the interpreter for the YAML *send* structure.

6.3. Connectivity with Social Networks and User Authentication

As mentioned previously, Ruby eases the connections with external applications by means of a wide set of gems. As an example, in Figure 9, a connection with the Twitter API is presented where authentication of the user is performed.

```
get '/twitter/newconnection' do
        user = User.get(session[:userId])

        consumer = OAuth::Consumer.new("", "")
        request_token=consumer.get_request_token(:oauth_callback => "http://#{request.host}:#{request.port}/twitter/callback")
        session[:twrequest_token]=request_token
        twitter = Twitterc.new(:requesttoken => request_token.token, :requestsecret => request_token.secret)
        user.socialnetworks << twitter
        redirect request_token.authorize_url

end
```

Figure 9. Connection with Twitter.

The connectivity with social networks could imply an important threat to the applicability of the approach, since any change in the network's API could imply a change in the approach. However,

this problem is mitigated by the utilization of the Ruby gems that abstract the developer of low level programming details and make the approach independent of the network's API.

The set of social applications being supported by SCPL includes both asymmetric ones like Twitter, GMail, Doodle, and Google Drive, and symmetric ones like Facebook and LinkedIn. However, it is currently being extended to incorporate other applications, such as Instagram or Dropbox.

7. Evaluation

In order to assess the validity of the proposed solution, we asked the same group of CMs to develop the three same processes again, but this time with SCPL. We also asked them to indicate which properties (from the selected ones) had been properly supported and to which degree (five-level Likert scale). P02.1 and P02.2 only contained two values (1 or 5) since it represented a binary situation (whether the feature was present or not). Additionally, we asked them to give their opinion about the advantages and disadvantages of the proposed solution from the point of view of the problems detected and addressed.

Table 3 presents the results. Each value indicates the arithmetic mean obtained for a given property. The properties best scored are P07 and P04, which are reusability and consistency (i.e., avoiding context switching). CMs highlighted the productivity gains thanks to these two properties in their comments. The second and third example took significantly less time than the first. In addition, they could take less if P0.3 (learnability) was better. In fact, the use of a textual representation of the language seems to penalize learnability. Although they felt comfortable with the syntax (P01), it was broadly admitted that non-expert users could have problems at the beginning. P05 (sufficiency) and P06 (abstraction) were also well considered. They are really related to the expressive power of the language, and all the CMs agreed that SCPL allowed some of the tasks at hand to be expressed better. P06 had a worse score due to CMs thinking that provided abstractions (e.g., number of "Likes") were too driven by the examples to perform, and they suggested the inclusion of a bigger set of abstractions in future versions. Asked about the overall evaluation, all the CMs agreed that SCPL presented better features than the tools they had used to fulfil the first questionnaire. Additionally, it may be observed how, unlike the rest of tools analyzed in Section 4.1 (see Table 2), SCPL fulfils all the features previously selected, with the exception of P02.2 (Graphical interface). Indeed, the lack of a graphical representation is the main drawback highlighted by CMs. We are aware of this problem and we are currently working on it. We did not want to develop the graphical version until having more accurate feedback from the community.

Table 3. SCPL evaluation.

Tool	P01	P02.1	P02.2	P03	P04	P05	P06	P07
SCPL	4.4	5	1	3.7	4.9	4.7	4.2	4.9

8. Threats to Validity

In this section, we elaborate on several factors that may jeopardize the validity of our results. In order to present these threats, we follow the well-known categorization introduced by [28], where threats are classified into four different validity categories: internal, external, construct, and conclusion. Although this categorization is mainly defined for experiments where quantitative data are obtained, it may also be applied in our case, where some qualitative data were collected.

8.1. Internal Validity

Internal validity refers to the relationship between the treatment for an experiment and the outcomes obtained—in other words, whether we are sure that treatment we used in an experiment really is related to the actual outcome we observed. In our study, we identified some threats that could influence internal validity, mainly classified into two main categories: (i) representativeness of the

subjects set; (ii) instrumentation used for the experiment. For all the threats identified, we also describe the actions that were considered to mitigate their possible effects.

Regarding the representativeness of the subjects used in the questionnaires, to avoid a potential influence in the results because of the respondent's experience and knowledge, the population contained professionals with the same level of experience and that were representative of the different kinds of organizations (companies) and domains of application. However, we are aware that the population only included Spanish professionals, which might introduce some bias in our results because of culture or education. Secondly, none of the selected CMs kept any kind of professional relationship with the companies providing the tools, so a possible bias caused by this issue was avoided.

Concerning the instrumentation used for the experiment (namely questionnaires), questions were shuffled for every participant so that possible inconsistencies in the answers could be identified. Furthermore, regarding quantitative data collected, we have calculated the standard deviation in order to detect any possible inconsistency, concluding that the results obtained by every CM may be, in general, considered consistent with the rest of participants. Additionally, the questionnaires were designed to be answered in less than 25 min (on average) so that the risk of obtaining low response rates was mitigated.

8.2. External Validity

External validity refers to the possibility of generalizing the results outside the scope of the study. In that sense, the main threats to external validity identified in our study concern the generalization of the population used for the experiment and also the application of the approach to other social applications where it has not yet been applied. For the former, the generalization of the population—as has already been mentioned—the subjects used to answer the questionnaires and evaluate our approach were professionals, so the applicability of the approach out of the academic scope has been proven by these professionals. Issues about the representativeness of the selected subjects have been already described in previous section, and they will be further discussed in Section 8.4.

Regarding the generalization of the approach, SCPL could conceivably be used in domains other than the ones illustrated here. Moreover, the most extended social applications—namely Facebook, Linkedin, Twitter, Gmail, Google Drive, and Doodle—have been used to test our approach. Although the results obtained may not be generalized for all the tools, most of them could conceivably be extrapolated to other social platforms, taking for granted that they are providing an API for connection (this connection is currently being implemented for some of these applications, such as Dropbox or Instagram) and they are following similar interaction styles. In that sense, the implementation of this new connection with each API could also be considered as a threat to validity; however, as previously commented, the development effort of these new connections is mitigated by the use of Ruby gems that provide facilities for the majority of the social networks, at least the most used or well-known (Examples of Ruby gems are available here [29]). Nevertheless, it should also be noted that the adaptation of the approach to these new scenarios might imply some conceptual changes in the language—e.g., to support new directives that could be introduced in the social applications being incorporated.

8.3. Construct Validity

Construct validity focuses on the relationship between the theory behind an experiment and the observations. In our case, considering that the results were obtained by using questionnaires, there is not a clear theory behind the data. However, in this section, we should remark some threats that arise from the qualitative data obtained by these questionnaires. For example, it is worth mentioning that we observed a productivity gain obtained by using the tool. Concretely, as previously mentioned (Section 7), the CMs highlighted a productivity improvement when using our tool, mainly due to reusability and the lack of context switching. This conclusion is supported by their claim about implementing the second and third examples in less time than the first. However, we are aware that

the data obtained to support this assumption are qualitative and, thus, they characterize (or describe) the property measured but they do not define (or measure) it. In that sense, additional quantitative data should have been collected—such as the measurement of the time spent in developing the motivating cases by each CM—in order to have stronger evidence of this important conclusion. We plan to include these data in an evaluation of the future graphical version of the tool.

8.4. Conclusions Validity

Obviously, the population of the study might be too small to be statistically representative, only covering 5% of the professionals registered in the association they were selected from. Additionally, it was completely formed by Spanish professionals. Thus, again, the results obtained may not be completely generalized. However, as commented above, the CMs were selected as a representative sample of the whole population by considering a representative set of the different kinds of organizations (companies) and domains of application.

9. Conclusions and Further Work

This paper has presented a DSL for the definition of social and cooperative applications. Based on this DSL, the user may define rules that connect different social applications—both symmetric and asymmetric—without the need of manually interacting with each of them. The DSL is supported by a tool that allows the human intervention to be reduced to only the definition of the SCPL program (the starting point of the process) and automating the rest of tasks (except, obviously, for the authentication in the social applications). The benefits provided by the language include: (i) reusability of the rules so that the user does not need to perform repetitive tasks; (ii) productivity improvement, since the user may interact with all these applications without a context change; (iii) user experience improvement, based on the previous benefits commented. The main drawback is that SCPL is based on a textual syntax, so it is not as user friendly as other approaches, although experienced developers usually do prefer textual interfaces because of the powerful flexibility that they provide.

In order to guide the development and validation of this approach, we have conducted a study to identify the most requested features for social media management applications. Basically, this study has been based on quantitative and qualitative data collected from a group of social media professionals or community managers (CMs) by means of questionnaires. Having addressed the main characteristics of our approach above, here we just focus on summarizing the main limitations of the process. First, the population of the study might be too small to be statistically representative and too localized (only Spanish professionals) to be representative enough in a general scenario. Second, some conclusions are derived only from qualitative data. Finally, the applicability of the approach could also be limited by the existence of public APIs for the social applications and the need to develop specific connectors for all of the APIs.

As further work, the approach is currently being extended to include more social applications and a graphical notation in order to provide support to those users that prefer this notation instead of the textual one. This improvement will obtain a trade-off between flexibility and user-friendliness. Furthermore, we plan to perform new user-based evaluations for the future graphical version of the approach so that the representativeness of the population may also be extended.

Acknowledgments: This work has been partially supported by Proyecto TIN2015-6957-R (MINECO/FEDER, UE) and Gobierno de Extremadura.

Author Contributions: All the authors work for the Quercus Software Engineering Group at Universidad de Extremadura and obtained their PhD from the same university. José M. Conejero collaborated with Fernando Sánchez-Figueroa in supervising the development of the different tools presented in this work. Roberto Rodríguez-Echeverría and Juan Carlos Preciado developed the tests that were needed to ensure that the approach fulfilled the requirements established. All the authors collaborated in writing the paper.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DSL	Domain Specific Language
SCPL	Social Cooperative Programming Language

References

1. Porter, J. Relationship Symmetry in Social Networks: Why Facebook Will Go Fully Asymmetric. Available online: http://bokardo.com/archives/relationship-symmetry-in-social-networks-why-facebook-will-go-fully-asymmetric/ (accessed on 20 July 2016).
2. Governor, J. Asymmetrical Follow: A Core Web 2.0 Pattern. Available online: http://redmonk.com/jgovernor/2008/12/05/assymetrical-follow-a-core-web-20-pattern/ (accessed on 20 July 2016).
3. Sánchez-Figueroa, F.; Preciado, J.C.; Conejero, J.M.; Rodríguez-Echeverría, R. Designing cooperative social applications in healthcare by means of SocialBPM. In Proceedings of the 11th International Conference on Cooperative Design, Visualization and Engineering, Seattle, WA, USA, 14–17 September 2014.
4. Nair, H.S.; Manchanda, P.; Bhatia, T. Asymmetric Social Interactions in Physician Prescription Behavior: The Role of Opinion Leaders. *J. Mark. Res.* **2010**, *47*, 883–895. [CrossRef]
5. Fowler, M. *Domain-Specific Languages*; Addison-Wesley Professional: New York, NY, USA, 2010.
6. Serrano, D.; Stroulia, E.; Barbosa, D.; Guana, V. SociQL: A Query Language for the SocialWeb. In *Advances in Network Analysis and Its Applications*; Mathematics in Industry; Springer-Verlag: Berlin, Germany, 2012; Volume 18, pp. 381–406.
7. Franchi, E. An Agent-Based Modeling Framework for Social Network Simulation. In *State of the Art Applications of Social Network Analysis*; Springer International Pusblishing: Cham, Switzerland, 2014; pp. 75–96.
8. Spanish Association of Social Media Professionals and Community Managers. Available online: http://www.aercomunidad.org (accessed on 20 July 2016).
9. IFTTT. Available online: https://ifttt.com/ (accessed on 20 July 2016).
10. Zapier. Available online: https://zapier.com/ (accessed on 20 July 2016).
11. Kitchenham, B.A.; Pfleeger, S.L. Principles of survey research: part 3: constructing a survey instrument. *ACM SIGSOFT Softw. Eng. Notes* **2002**, *27*, 20–24. [CrossRef]
12. Brambilla, M.; Fraternali, P.; Vaca, C. A Notation for Supporting Social Business Process Modeling. In *Business Process Model and Notation (BPMN)*; Lecture Notes in Business Information Processing; Springer-Verlag: Berlin, Germany, 2011; Volume 95, pp. 88–102.
13. Bourque, P.; Fairley, R.E. (Eds.) *Guide to the Software Engineering Body of Knowledge, Version 3.0*; IEEE Computer Society: Piscataway, NJ, USA, 2014. Available online: www.swebok.org (accessed on 20 July 2016).
14. We Wired Web. Available online: https://wewiredweb.com/ (accessed on 20 July 2016).
15. Elastic.io. Available online: http://www.elastic.io/ (accessed on 20 July 2016).
16. Cloud Work. Available online: https://cloudwork.com/ (accessed on 20 July 2016).
17. WappWolf. Available online: http://wappwolf.com/ (accessed on 20 July 2016).
18. itDuzzit. Available online: http://cloud.itduzzit.com/ (accessed on 20 July 2016).
19. Kissflow. Available online: https://kissflow.com/ (accessed on 20 July 2016).
20. Jara, J.; Daniel, F.; Casati, F.; Marchese, M. From a simple flow to social applications. In *Current Trends in Web Engineering*; Springer LNCS: Aalborg, Denmark, 2013; Volume 8295, pp. 39–50.
21. Ahmad, S.; Battle, A.; Malkani, Z.; Kamvar, S. The jabberwocky programming environment for structured social computing. In Proceedings of the 24th ACM User Interface Software and Technology Symposium (UIST), Santa Barbara, CA, USA, 16 October 2011; pp. 53–68.
22. Brambilla, M.; Fraternali, P.; Vaca, C.; Butti, S. Combining Social Web and BPM for Improving Enterprise Performances: the BPM4People Approach to Social BPM. In Proceedings of the 21st International Conference Companion on World Wide Web (WWW), European-Projects Track, Lyon, France, 16–20 April 2012.
23. OMG Business Process Model and Notation. Available online: http://www.bpmn.org/ (accessed on 20 July 2016).
24. Xtext. Available online: https://eclipse.org/Xtext/ (accessed on 20 July 2016).
25. Xtend. Available online: http://eclipse.org/xtend/documentation/index.html (accessed on 20 July 2016).

26. YAML. Available online: http://yaml.org/spec/ (accessed on 20 July 2016).
27. Ruby. Available online: https://www.ruby-lang.org/ (accessed on 20 July 2016).
28. Wohlin, C.; Höst, M.; Runeson, P.; Ohlsson, M.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering: An Introduction*; Kluwer Academic Publishers: Norwell, MA, USA, 2000.
29. The Ruby Toolbox. Available online: https://www.ruby-toolbox.com/categories/api_clients (accessed on 20 July 2016).