Better Call the Crowd. Using Crowdsourcing to Shape the Notation of Domain-Specific Languages

Marco Brambilla
Politecnico di Milano
Milano, Italy
marco.brambilla@polimi.it

Jordi Cabot
ICREA
UOC
Barcelona, Spain
jordi.cabot@icrea.cat

Javier Luis Cánovas Izquierdo
UOC
Barcelona, Spain
jcanovasi@uoc.edu

Andrea Mauri
Politecnico di Milano
Milano, Italy
andrea.mauri@polimi.it

Abstract

Crowdsourcing has emerged as a novel paradigm where humans are employed to perform computational tasks. In the context of Domain-Specific Modeling Language (DSML) development, where the involvement of end-users is crucial to assure that the resulting language satisfies their needs, crowdsourcing tasks could be defined to assist in the language definition process. By relying on the crowd, it is possible to show an early version of the language to a wider spectrum of users, thus increasing the validation scope and eventually promoting its acceptance and adoption. We propose a systematic method for creating crowdsourcing campaigns aimed at refining the graphical notation of DSMLs. The method defines a set of steps to identify, create and order the questions for the crowd. As a result, developers are provided with a set of notation choices that best fit end-users' needs. We also report on an experiment validating the approach.

CCS Concepts • **Software and its engineering** → *Context specific languages*; *Software development process management*; *Collaboration in software development*;

Keywords crowdsourcing, domain-specific languages, model-driven development

ACM Reference Format:
Marco Brambilla, Jordi Cabot, Javier Luis Cánovas Izquierdo, and Andrea Mauri. 2017. Better Call the Crowd. Using Crowdsourcing to Shape the Notation of Domain-Specific Languages. In *Proceedings of 2017 ACM SIGPLAN International Conference on Software Language Engineering (SLE'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/

1 Introduction

Domain-Specific Modeling Languages (DSMLs) are a special kind of languages which have been tailored to solve a particular problem in a domain. As they target a concrete domain, the development of DSMLs requires a tight collaboration between language developers and end-users, who are usually the domain experts. While language developers provide the technical knowledge, end-users help setting the language concepts and shaping the notation most suitable for the domain. Involving end-users therefore enriches the process and assures the resulting language satisfies their needs [Kelly and Pohjonen 2009; Völter 2011].

In the last years, crowdsourcing has emerged as a novel paradigm in which humans are employed to perform computational tasks [Doan et al. 2011]. Crowdsourcing has been successfully used in very different scenarios such as fact checking, opinion mining or localized information gathering. Among their applications, crowdsourcing techniques can be used to create and validate software artifacts [LaToza et al. 2014]. In the context of DSMLs, crowdsourcing tasks could be defined to create and validate language constructs.

In particular, we envision a collaborative development process of DSMLs illustrated in Figure 1 which follows these phases:

1. The language is initialized by language developers with initial definitions for the abstract (i.e., concepts and relationships) and concrete (i.e., graphical and/or textual notation) syntaxes.
2. Language elements are polished by a set of end-users who are experts in the domain and collaborate closely with language developers to shape the language (approaches like Collaboro [Cánovas Izquierdo and Cabot 2016] can be used here). At this point, developers can

SLE'17, October 23–24, 2017, Vancouver, Canada
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5525-4/17/10…$15.00

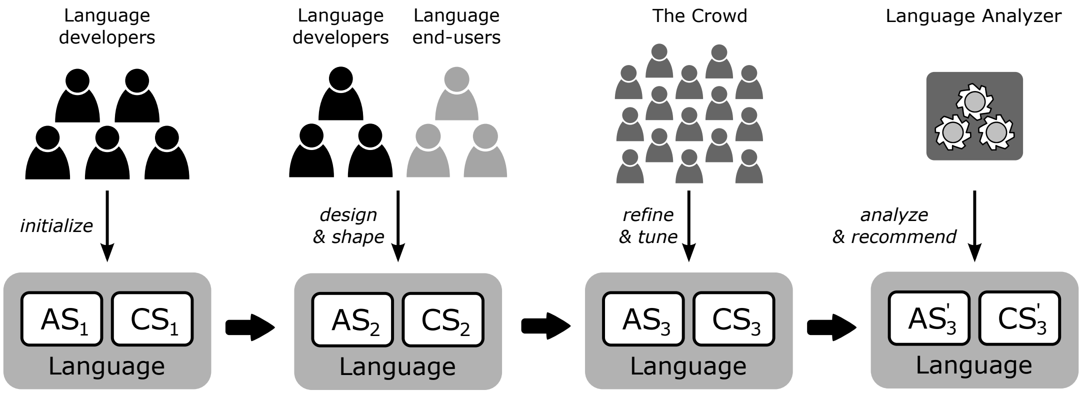


Figure 1. Language development process proposed. AS = Abstract Syntax. CS = Concrete Syntax.

be sure that the language fulfills the core end-users' needs.

3. Crowdsourcing techniques are applied to validate the language notation. While the previous phase concentrates the changes in the concepts and main shape of the language, this phase leverages the crowd to show the language to a wider spectrum of users and tune its notation.
4. (Optional) Once the language has been developed and deployed, an analyzer (manual or automatic) studies how the language is being used in practice to detect common patterns and propose improvements (e.g., removing hardly used elements or promoting as new primitives the most typical elements).

In this paper we focus on the third step of the process, in which the crowd is used to validate and refine the concrete syntax of a DSML. By relying on the crowd, it is possible to show the language to a bigger spectrum of users, thus increasing the validation scope and eventually promoting its acceptance and adoption. In particular this is necessary when the language aims to be adopted in a global scale, where the heterogeneity of the end-users could lead to misinterpretation and errors in its usage.

Our contribution bridges the world of crowdsourcing and DSML design by providing a systematic method for creating crowdsourcing campaigns aimed at validating the graphical notation of DSMLs. By taking as inputs the abstract syntax of the language and its candidate concrete syntaxes (i.e., possible graphical symbols), our process generates the set of crowdsourcing tasks including the questions for the crowd. The collected answers can then be used to refine the concrete syntax that most fit the end-users' needs. At the best of our knowledge, no previous works studied a systematic approach for involving large crowds in the validation of DSML concrete syntaxes, addressing the problems of the cost (e.g., minimizing the number of questions) and quality.

The paper is structured as follows. Section 2 presents some background on Crowdsourcing and Section 3 describes our approach. Section 4 describes the validation of our approach while Section 5 discusses the lessons learned. Section 6 presents the related work and Section 7 ends the paper.

2 Crowdsourcing for Language Design

A crowdsourcing campaign is composed of a set of tasks including one or more questions to be asked to the crowd. The interaction is generally among (1) the requester, who poses a task; (2) the system, which organizes the computation by mixing conventional and crowd-based modules; and (3) a potentially wide set of performers, who are in charge of performing crowd tasks (i.e., answering the questions) and are typically unknown to the requester.

A campaign lifecycle consists of three phases: (1) design by the requester, (2) execution by asking the crowd and (3) answers aggregation. Figure 2(a) shows this workflow. Notice that these phases can be often overlapping, for instance, the design of the campaign can change while the tasks are being executed and the answer aggregation phase can start while the crowd is working on the tasks [Brambilla et al. 2015].

To support the definition and execution of crowdsourcing campaigns for language design, we record the campaign information by means of the model shown in Figure 2(b), which is inspired by the work presented by Bozzon et al. [Bozzon et al. 2015a]. In the following, we describe each phase in detail along with the model elements involved.

Design. This phase consists in configuring the tasks and their corresponding questions for the campaign. It is conducted by instantiating the left part of the model shown in Figure 2(b), in particular, a crowdsourcing *Campaign* is composed of a set of *Task*s, which ask the *Performer* to answer some *Question*s. As we want the performer to select the most suitable candidate concrete syntax elements for a language, we devise the employment of classification tasks, where questions include a set of options and performers have to chose one. Thus, each question asks about a specific *Abstract Syntax Element* and it includes two or more options, each one referring to a *Candidate Concrete Syntax Element*.

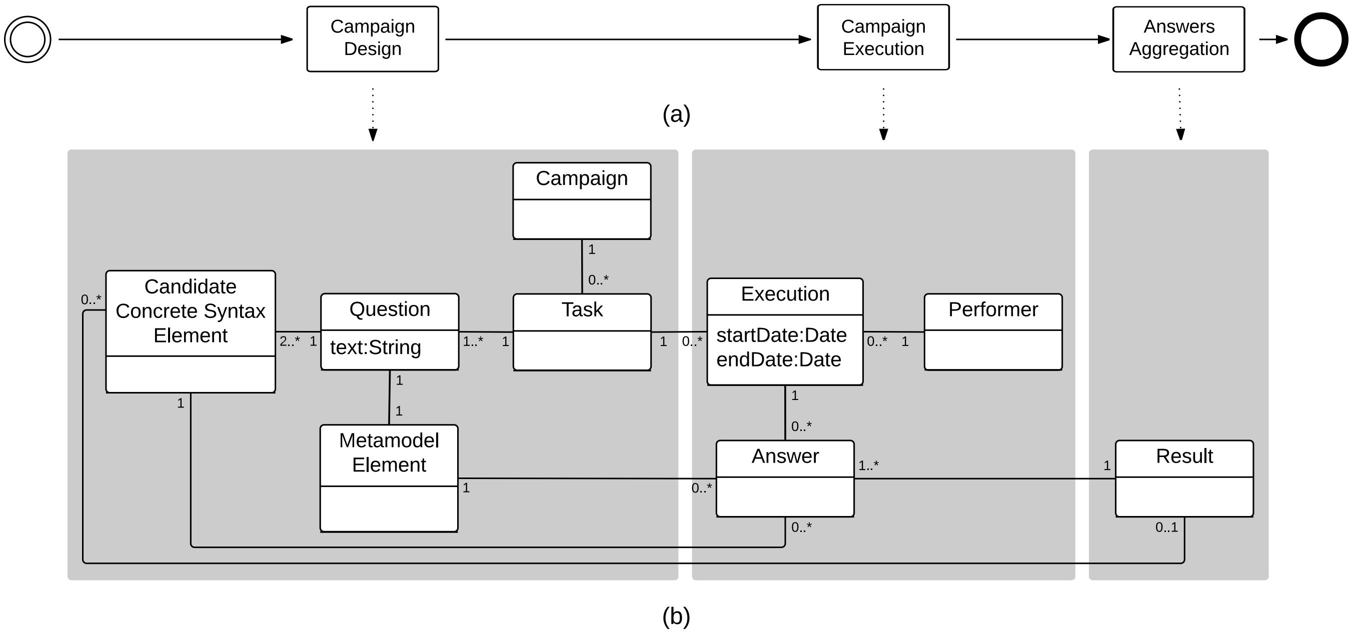


Figure 2. (a) Crowdsourcing campaign lifecycle and (b) the supporting model.

Execution. In this phase, performers conduct the tasks providing their answers. In our metamodel, the *Performer* executes the *Task* by answering its *Questions*, which involves choosing which *Candidate Concrete Syntax Element* he/she prefers for representing the *Abstract Syntax Elements*. Thus, a task *Execution* includes some statistics about the performer actions and tracks the performer choices by means of *Answers*.

Answers aggregation. The final result of a crowdsourcing campaign is obtained by combining the different contributions provided by the crowd, for which several methods have been proposed as it depends on the particular scenario. In the case of language design, where classification tasks are used, we simply count the preferences for each option [Bozzon et al. 2013]. The *Answers* gathered are aggregated and analyzed in order to find the *Result*.

3 Derivation of Crowdsourcing Campaigns

Our process receives as input: (1) the abstract syntax of the language (specified as a metamodel), (2) a set of symbols representing the candidate concrete syntax and (3) the mapping between the two (i.e., which symbol/s correspond to each concept/relationship). Optionally, the user can also provide a set of models to be used as templates when illustrating the usage of the different candidate notations. If this is not provided, questions will display the symbols isolated. With these inputs, the method we describe herein generates a crowdsourcing campaign to let users choose the best language syntax.

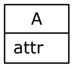
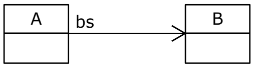
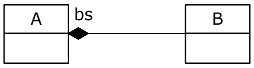
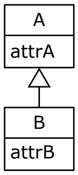
We will use as running example, a simplified business process modeling language, to illustrate our approach. We define a simple metamodel defining the abstract syntax (Figure 3) and a set of candidate symbols for each concrete metamodel element (except for the *Flow* concept, for which we decided to set the use of an arrow for the sake of simplicity). The definition of candidate symbols was inspired by the BPMN standard notations and following the recommendations from Moody’s work on “Physics of Notations” [Moody 2009]. Table 4 shows the proposed symbols (see option columns).

3.1 Design

In order to design the crowdsourcing campaign, we have to instantiate the elements of our campaign model corresponding to this phase. In particular, it is necessary to first identify the set of *Task* elements composing our crowdsourcing *Campaign* and then define the *Questions* to be included in each *Task*. We devise a crowdsourcing model generation driven by the abstract syntax metamodel of the language. To this aim, we define a set of metamodel patterns which, once applied to the language abstract syntax metamodel, will generate the required campaign model elements. The patterns have been defined by an extensive review of existing metamodels such as IFML , UML, and BPMN, and our expertise creating DSMLs and the recurring structural patterns appearing therein.

Tables 1 and 2 describe the patterns and how they are used to create *Tasks* and *Questions*. For each pattern we show an example model (i.e., matching example), the set of

Table 1. Simple patterns.

Id	Pattern (Matching example)	Generated Elements in Campaign Metamodel	Question Example
P1		• One *Task* • One *Question* for the class element (i.e., *A* class) • One *Question* for each attribute element (i.e., *attr* attribute)	• Which symbol do you prefer for representing the element *A*? • How do you prefer to represent the attribute *attr*?
P2		• One *Task* • One *Question* for the relation element (i.e., *bs* relation)	• How do you prefer to represent the relation *bs* between *A* and *B* elements?
P3		• One *Task* • One *Question* for the containment relation element (i.e., *bs* relation)	• How do you prefer to represent the containment relation *bs* between *A* and *B* elements?
P4		• One *Task* • One *Question* for the parent class (i.e., *A* class) • One *Question* for each attribute in parent class (i.e., *attrA* attribute) • One *Question* for the child class (i.e., *B* class) • One *Question* for each attribute in child class (i.e., *attrB* attribute)	• How do you prefer to represent the element *A*, and its attribute *attrA*? • How do you prefer to represent the element *B*, and its attribute *attrB*?

elements to create as a consequence of the match in the campaign metamodel and some illustrative questions. Note that *Question*s refer to the abstract syntax and concrete syntax elements involved in the pattern. We identified two types of patterns: simple (see Table 1), which includes the basic constructs of the metamodel; and composite (see Table 2), which represents unified specification practices and reduces the number of tasks by aggregating questions on related elements (e.g., pattern P7 creates a single task asking about all the elements of a hierarchy).

More specifically, given an abstract syntax metamodel, our pattern application and task generation process follows these steps:

1. Mark all the abstract syntax elements as *nonvisited.*
2. Identify patterns to be applied, from the most general (i.e., P8) to the most specific (i.e., P1).
3. For any *nonvisited* elements, apply the most general pattern identified.
4. Generate a *Task* and a set of *Question*s according to pattern:

   4.1. Generate *Question*s only for abstract syntax elements marked as *nonvisited* and having a mapping to a concrete syntax element.

   4.2. Mark the abstract syntax elements involved in the questions as *visited.*

   4.3. Associate to the question the examples of models provided by the designer if available; otherwise generate simple diagrams showing element usage.
5. Go to step 3 until all abstract syntax elements have been marked as *visited.*

If a *visited* element is included in a pattern involving also at least one element marked as *nonvisited*, the *Question* for the visited one is not generated again; instead, the previously selected symbol is shown.

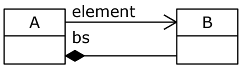
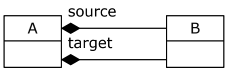
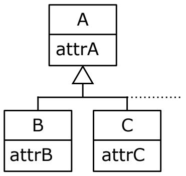
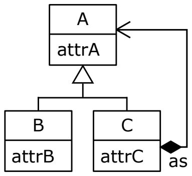
Figure 3 shows in grey boxes the patterns found in our example metamodel (step 2). In particular the algorithm detected three P7 patterns corresponding to each hierarchy (i.e., hierarchies with roots *Gateway*, *Event* and *Node*). Table 3 shows the set of iterations required to apply the patterns (steps 3-5). The text of the *Questions* is generated using a template-based approach.

3.2 Execution

Tasks are executed sequentially but in random order to avoid a possible bias due to the order in which the questions are asked. At the same time, past answers of a user in the same campaign must affect future questions by dynamically rendering the graphical examples in a question with the choices already made in past questions.

The first time a performer sees a task, the involved example is built either selecting randomly the symbols of the

Table 2. Composite patterns.

Id	Pattern (Matching example)	Generated Elements in Campaign Metamodel	Question Example
P5		• One *Task* • One *Question* for the relation element (i.e., *element* relation) • One *Question* for the containment relation element (i.e., *bs* relation)	• How do you prefer to represent the containment relation *bs* between *A* and *B* elements? • How do you prefer to represent the relation *element*?
P6		• One *Task* • One *Question* for each containment relation (i.e., *source* and *target* relations)	• How do you prefer to represent the containment relations *source* and *target* between *A* and *B* elements?
P7		• One *Task* • One *Question* for the parent class (i.e,. *A* class) • One *Question* for each attribute in parent class (i.e., *attrA* attribute) • One *Question* for each child class (i.e,. *B* and *C* classes) • One *Question* for each attribute in child classes (i.e., *attrB* and *attrC* attributes)	• How do you prefer to represent the element *A*? • How do you prefer to represent the elements *B* and *C*?
P8		• One *Task* • One *Question* for the parent class (i.e,. *A* class) • One *Question* for each attribute in parent class (i.e., *attrA* attribute) • One *Question* for each child class (i.e,. *B* and *C* classes) • One *Question* for each attribute in child classes (i.e., *attrB* and *attrC* attributes) • One *Question* regarding the containment relation (i.e., *as* relation)	• How do you prefer to represent the element *A* and its attribute *attrA*? • How do you prefer to represent the elements *B* and *C*, and their attributes *attrB* and *attrC*? • How do you prefer to represent the containment relation *as* between *C* and *A* elements?

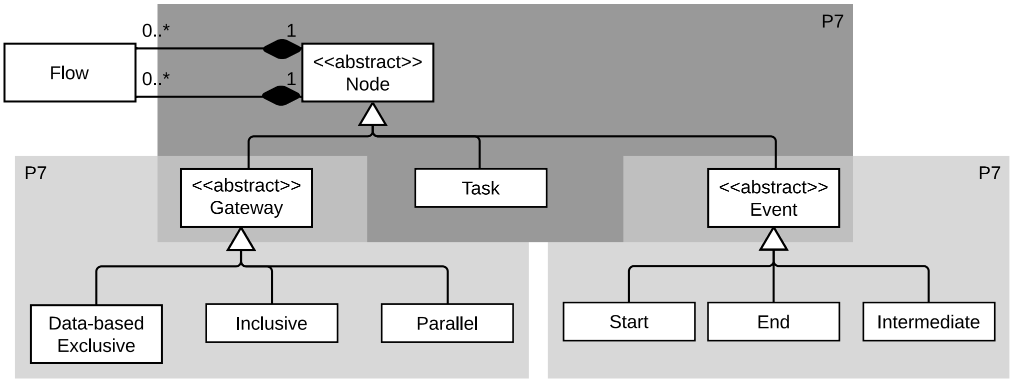


Figure 3. Running example metamodel (patterns identified in grey boxes).

various elements or, if enough answers have already been gathered, using the symbols preferred by the crowd up to that moment (this is configurable at the beginning of the campaign). In the following tasks the performer will see in the examples the symbols he/she previously chose. Furthermore, in this way, when model examples are given, the

Iteration	Pattern	Generated Task	Elements involved in the *Question*
1	P7	Task 1	*Conditional, And, Parallel*
2	P7	Task 2	*Start, End, General*
3	P7	Task 3	*Task*

Table 3. Tasks generated for the patterns found in Figure 3.

performer also takes into account the expressiveness of the single symbol together with the overall visual style.

Differently from the cases where questions imply an objectively correct answer, in our case there is no right or wrong notation, just a preference. This makes it difficult to monitor that the performer is really putting in effort when selecting the preferred symbols and therefore that we obtain high quality answers. To identify performers who answer quickly and randomly to maximize their reward, we compare their task execution time to the average time of the crowd. If the time is too low we will not consider their contribution [Gadiraju et al. 2015].

3.3 Answers Aggregation

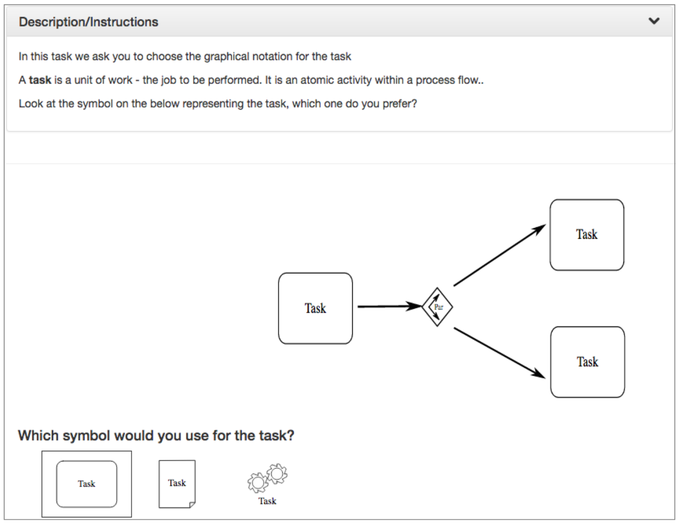
For the phase of aggregating the collected answers we adopt two well-known state of the art strategies: (1) *static majority* and (2) *targeted agreement* [Bozzon et al. 2013]. With the *static majority* approach the tasks are executed until a fixed amount, decided at design time, of answers is gathered. Then the final result is obtained by counting each preference for the single element and we select the symbol that was chosen by the majority of the performers. *Targeted agreement* consists in choosing an agreement level among the performers that needs to be reached (e.g., the 70% must agree on the concrete syntax symbol of an element). Then the task is executed until such agreement is reached.

4 Experiment

To evaluate our approach we conducted an experiment where we asked the crowd to address the running example of this paper. To deploy our crowdsourcing campaign, we extended and configured CrowdSearcher[1] [Bozzon et al. 2015a], a tool for developing crowd-based applications which uses a variety of systems for interacting with crowds, spanning from crowdsourcing platforms (e.g., Amazon Mechanical Turk) to social networks and other platforms.

4.1 Scenario

Based on the abstract syntax metamodel and the corresponding candidate concrete syntaxes, we generated automatically the crowdsourcing campaign, tasks and questions according to the strategies described so far, and deployed them to our own cloud infrastructure.



Description/Instructions

In this task we ask you to choose the graphical notation for the task

A **task** is a unit of work - the job to be performed. It is an atomic activity within a process flow..

Look at the symbol on the below representing the task, which one do you prefer?

Which symbol would you use for the task?

Figure 4. User interfaces for the crowdsourcing tasks asking about the *task* symbols.

The experiment was composed by three tasks asking about the symbols of (1) the three types of gateways, (2) the three types of events and (3) the task element (Table 4). Figure 4 shows the user interface for the selection of the *task* symbols. Each task included a short description, a picture with a model example and the questions themselves. When a performer selects an option, the picture with the example changes accordingly, thus helping to decide which symbol is the most suitable. As commented before, user choices made in the previous tasks influence the remainder of the crowdsourcing tasks by showing the previously selected symbols.

At the beginning of the crowdsourcing campaign, each performer was also asked to provide demographic information and an evaluation of his/her expertise in modeling, language specification and business processes.

We conducted our experiment for one week and we engaged a heterogeneous set of participants (85 in total) composed of: undergraduate/graduate computer science students, modeling experts and IT professionals coming from more than 10 different countries. We used the *static majority* policy as answers aggregation strategy during the experiment execution and we also simulated the *targeted agreement* strategy using the collected answers, at the purpose of comparing effectiveness and affordability of the two approaches.

4.2 Results

Table 4 (last column) shows the barcharts reporting the collected results per question. For each question we show the aggregated result of all the performers (left bar), beginners (center), and experts (right). The degree of expertise was computed in the following way. As mentioned above, before executing the task the performers were asked to asses their expertise in *modeling*, *language specification* and *bpmn*

[1] http://crowdsearcher.deib.polimi.it/

experience by saying if they did not have any knowledge (none), if they had some basic skills (medium) or if they were experts (high). We assigned a value at each level: 0 to none, 1 to medium and 2 to high; then we computed an expertise value of each performer by making the average on the three values. Finally we considered experts the users that had an expertise value greater than 1. This aggregated "expertise" concept helped us to simplify the amount of dimensions to consider in the analysis reported in the paper[2].

Option B is the winner in most cases, regardless the expertise profile. Option A has been preferred for the *Task* and *End Event* symbols, while Option C has been selected for the *Exclusive Gateway*. Similar results were obtained when considering the expertise level, although some questions were more influenced by the option A, which was the symbol proposed by the BPMN standard (e.g., *End Event*).

We also tested the *targeted agreement* answer aggregation method described in Section 3.3. We configured the method by setting the minimum number of answers per question to 10, the agreement level at 60% and as closing condition a maximum of 50 task executions. Table 5 shows the agreement reached and the number of answers needed to close the tasks. The agreement level was reached for three elements (i.e., *Task*, *Data-based Exclusive Gateway* and *Intermediate Event* symbols). The remaining tasks were closed after reaching the maximum number of answers (see rows with the value of 50 in the *Answers Needed* column). The total number of answers was significantly smaller than the one needed in the *static majority* strategy (235 vs. 595). Moreover, we assessed that the distribution of the answers didn't change.

4.3 Validation

In order to verify that a graphical notation obtained using crowdsourcing does provide advantages (in term of clarity, acceptance, and intuitiveness), we set up an additional experiment on Amazon Mechanical Turk, where we asked the crowd to compare the different graphical notations obtained with our approach. We defined the winning notation as composed of all the symbols ranked first in the preference of all the performers, as per Table 4; the second notation as the set of symbols ranked second; and the third with the symbols ranked third.

We used these notations to build a simple model describing the booking process on a travel website. An example is shown in Figure 5: the user can login using his credentials or alternatively register a new account, in the latter case he must validate his data by clicking on a confirmation link he receives via email. Then the user has to book both the flight and the hotel, subsequently he can buy optional services such as: car rental, travel insurance, and airport shuttle bus. Finally, he confirms his choices and the process ends.

Every worker was shown two example diagrams, built with different graphical notations, describing the aforementioned business case and we asked them to choose which one they preferred in term of intuitiveness, coherence, and readability (and to motivate their answer). We paid 0.05 USD per answer, and we collected a total of 204 answers. As a result, our "best" notation was preferred 60% of the times over the second best, and 69% of the times over the third one. Moreover our second best is preferred the 67% of times over the third one.

This provides a good intuition that with our method is indeed possible to create graphical syntaxes that are more accepted and easy to understand for non experts.

4.4 Threats to validity

Our study is subjected to some threats to validity. The limited number of options for each symbol narrows down the set of candidate alternatives and may not reflect the performer's preference. Moreover, performers do not have the option to fully disagree with the proposed options (i.e., they are forced to choose among the fixed set of option). On the positive side, this simplifies the questions and promotes the selection process.

Another threat to validity is the subjectivity in understanding the reasons behind the selections made by the performers. It is hard to assess why different people choose one symbol over another and even the reasons to select the same symbol may be different for each individual. Therefore, the method works to find the popular choices but cannot be used to generalize global preferences on language notations or styles. In this sense, note that our results cover a simplified version of a well-known language and therefore our results may differ for other DSMLs.

Finally, we would like to remark the chosen notation is not the best in absolute terms, it is the preferred one (e.g., the chosen one could be more complex than other alternatives). Language designers must evaluate this trade-off when making a final decision.

5 Discussion

This paper represents the first attempt to a systematic approach for involving the crowd in the selection and validation of DSMLs concrete syntaxes, starting from a metamodel definition and a candidate set of symbols to be potentially used as concrete syntax. The results of the experiments allows us to draw some conclusions and open some discussion points regarding our approach:

Conclusive results. In our experiment, out of the 7 symbols, the crowd came to an absolute majority agreement in 5 of them, thus, making the crowdsourcing campaign useful to decide most of the target symbols;

[2] The full results are available online http://crowdsearcher.deib.polimi.it/casestudies/

Table 4. Candidate concrete syntax alternatives of our running example and collected answers for the experiment.

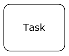
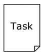
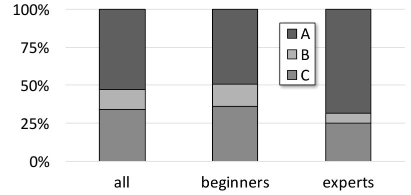
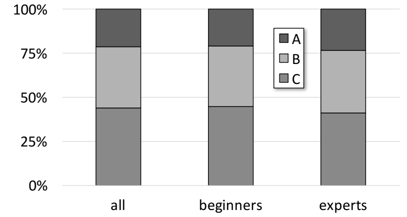
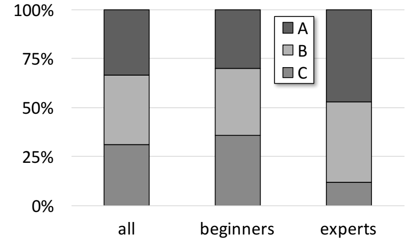
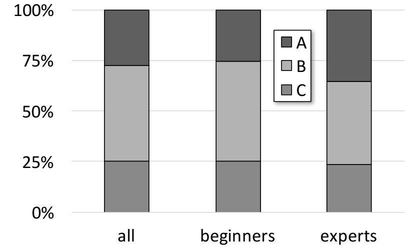
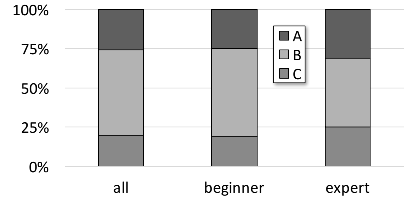
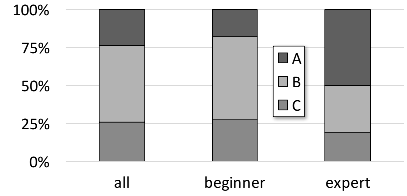
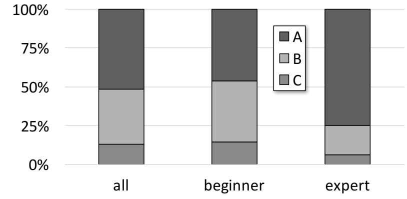
Abstract Syntax Element		**Concrete Syntax Symbols**			**Crowd Assessment Results (% of preferences for each symbol)**
		Option A	**Option B**	**Option C**	
Task					
Gateway	*Data-based Exclusive*				
	Inclusive				
	Parallel				
Event	*Intermediate*				
	Start				
	End				

Table 5. Agreement levels reached and number of answers needed for each element when using the *targeted agreement* strategy.

Metamodel Element		Agreement Level Reached	# Answers Needed
Task		**60%**	10
Gateways	*Data-based Exclusive*	**60%**	10
	Inclusive	38%	50
	Parallel	44%	50
Events	*Intermediate*	**60%**	15
	Start	42%	50
	End	40%	50

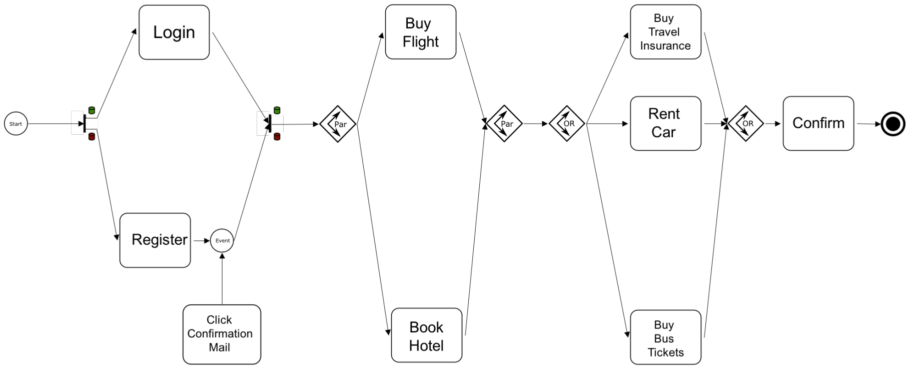


Figure 5. Example of process shown to the Amazon Mechanical Turk workers built with our "best" notation.

Iterative nature. You may not reach a definite result for all your symbols. In those cases, a second campaign can be useful, this times focusing only on those conflicting symbols.

Quicker results employing the dynamic targeted agreement. Our experiment confirms that targeted agreement could be a useful way to narrow down the conflicting decisions and save effort.

Several notations may be a good idea. At first sight the analysis of the answers split up according to expertise profile reveals differences depending on the knowledge of the performer. This is the case for the symbols of *Start Event*s, *End Event*s, *Inclusive Gateway*s and *Task*s. Even if a further analysis based on Chi-Square tests reveals that only the answers for the symbol of *End Event* are significantly different among the two expertise levels and thus further research needs to be conducted, we believe that on large user bases for a language, having more than one concrete notation (and the corresponding translators among them) for the same abstract syntax could be necessary.

Coherent answers. We analyzed the coherence of the performers in the tasks that contained multiple questions. Our hypothesis is that the symbols whose elements belong to a hierarchy need to be evaluated at the same moment because they should have similar "look and feel". The analysis shows that for the *Events* the users selected the same type of symbols for 72% of times, while for the *Gateway*s the coherence was 54%, confirming that patterns help to reach a certain coherence level for the language.

In the future we plan to experiment on multiple domains and to better exploit the disagreement that may appear in the results, in order to generate possibly multiple notations targeting specific profiles of users. Finally, we will investigate the involvement of the crowd in other steps of the process of modeling language development, for instance, in the definition/validation of the abstract syntax and whether there is a correlation between the chosen notation and other typical language metrics.

6 Related Work

Several works have emphasized the need of involving the end-user community of the language in the development process. However such involvement has been tried at a small-scale so far. For instance, López-Fernández et al. [López-Fernández et al. 2015] aims at deriving the language by parsing concrete examples provided by the users. A more interactive approach is proposed by Cánovas et al. [Cánovas Izquierdo and Cabot 2016] with Collaboro, a tool for enabling collaborative modeling language definition and by Umuhoza et al. [Umuhoza et al. 2015] where authors work with end-users to understand which part of the language were too hard to understand and simplify it.

Other studies focused on how to improve the notation of languages: works on I* [Genon et al. 2012], UML [Khendek 2015] and BPMN [Genon et al. 2011] examined the visual notations using the "Physics of Notations" theory [Moody 2009], highlighting problems related to their complexity. All of them conclude suggesting that the involvement of non-expert people could increase the cognitive effectiveness of the language.

Not strictly related to our field of investigation, interesting is the work of Xu et al. [Xu et al. 2014] where the authors use the crowd to gather feedbacks on works of visual design.

Many studies have been done on task decomposition and result aggregation. For instance in [Bernstein et al. 2010] the authors propose to decompose the task in three phases called *Find, Fix and Verify*. In [Kittur et al. 2011] the authors describe a framework that split complex tasks using map-reduce. Finally, [Bozzon et al. 2015b] proposes a general purpose methods and tools for designing crowd-based workflows.

Nevertheless, while we do not claim that our approach is novel with respect to the aforementioned ones, at the best of our knowledge no previous works studied a systematic approach for involving big crowds in the design of the concrete syntax of a DSML, addressing the problems of the cost (e.g., minimizing the number of questions) and quality.

7 Conclusion

In this paper we proposed a systematic approach for involving the crowd in the validation of modeling language concrete syntaxes. We showed how a crowdsourcing campaign can be configured starting from the metamodel elements and a set of candidate graphical symbols. The method covers the main phases of crowdsourcing campaings, ranging from task generation to control execution and answer aggregation.

In the future we plan to study whether the number and quality of questions could be optimized (e.g., driven by the developers to refine/remove specific elements) and to better exploit the disagreement that may appear in the results. It would be interesting to correlate the performer profiles to their choices, in this way we can understand if the users' disagreement is caused by cultural differences. In this case it may be more suitable to have several winning concrete syntaxes, instead of trying to promote only one. Furthermore we plan to analyze how the end-user usage of the language can be mined to further define its properties (e.g., by removing the not used elements). We would also like to investigate the involvement of the crowd in other steps of the process of modeling language development, for instance, in the validation and also definition of the abstract syntax. Finally, we would like to explore how to adapt our approach to evaluate textual notation or language semantics.

Acknowledgments

This work has been supported by the Spanish government (TIN2016-75944-R project).

References

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010. Soylent: A Word Processor with a Crowd Inside. In *ACM Symposium on User Interface Software and Technology*. 313–322.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Andrea Mauri. 2013. Reactive Crowdsourcing. In *World Wide Web Conference*. 153–164.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Andrea Mauri, and Riccardo Volonterio. 2015a. Designing Complex Crowdsourcing Applications Covering Multiple Platforms and Tasks. *Jounal on Web Engineering* 14, 5&6 (2015), 443–473.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Andrea Mauri, and Riccardo Volonterio. 2015b. Designing Complex Crowdsourcing Applications Covering Multiple Platforms and Tasks. *Journal on Web Engineering* 14, 5-6 (2015), 443–473.

Marco Brambilla, Stefano Ceri, Andrea Mauri, and Riccardo Volonterio. 2015. Adaptive and Interoperable Crowdsourcing. *IEEE Internet Computing* 19, 5 (2015), 36–44.

Javier Luis Cánovas Izquierdo and Jordi Cabot. 2016. Collaboro: a collaborative (meta) modeling tool. *PeerJ Computer Science* 2, e84 (2016).

Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. 2011. Crowdsourcing Systems on the World-Wide Web. *Commun. ACM* 54, 4 (2011), 86–96.

Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. 2015. Understanding Malicious Behavior in Crowdsourcing Platforms: The Case of Online Surveys. In *ACM Conference on Human Factors in Computing Systems*. 1631–1640.

Nicolas Genon, Patrice Caire, Hubert Toussaint, Patrick Heymans, and Daniel L. Moody. 2012. Towards a more semantically transparent i* visual syntax. In *RE conf.*, Vol. 7195. LNCS, 140–146.

Nicolas Genon, Patrick Heymans, and Daniel Amyot. 2011. Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In *ACM SIGPLAN International Conference on Software Language Engineering*. 377–396.

Steven Kelly and Risto Pohjonen. 2009. Worst Practices for Domain-Specific Modeling. *IEEE Software* 26, 4 (2009), 22 –29.

Ferhat Khendek. 2015. On the Semantic Transparency of Visual Notations: Experiments with UML. In *SDL 2015: Model-Driven Engineering for Smart Cities: International SDL Forum*. 122–137.

Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E. Kraut. 2011. CrowdForge: Crowdsourcing Complex Work. In *ACM Symposium on User Interface Software and Technology*. 43–52.

Thomas D. LaToza, W. Ben Towne, Christian M. Adriano, and André van der Hoek. 2014. Microtask Programming: Building Software with a Crowd. In *ACM Symposium on User Interface Software and Technology*. 43–54.

Jesús J. López-Fernández, Jesús Sánchez Cuadrado, Esther Guerra, and Juan De Lara. 2015. Example-driven Meta-model Development. *Software and Systems Modeling* 14, 4 (2015), 1323–1347.

Daniel Moody. 2009. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering* 35, 6 (2009), 756–779.

Eric Umuhoza, Marco Brambilla, Davide Ripamonti, and Jordi Cabot. 2015. An Empirical Study on Simplification of Business Process Modeling Languages. In *ACM SIGPLAN International Conference on Software Language Engineering*. 13–24.

Markus Völter. 2011. MD*/DSL Best Practices. http://voelter.de/data/pub/DSLBestPractices-2011Update.pdf. (2011).

Anbang Xu, Shih-Wen Huang, and Brian Bailey. 2014. Voyant: Generating Structured Feedback on Visual Designs Using a Crowd of Non-experts. In *Conference on Computer Supported Cooperative Work*. 1433–1444.