Expert Sourcing to support the Identification of Model Elements in System Descriptions

Marta Sabou, Dietmar Winkler, Sanja Petrovic

Institute of Software Technology and Interactive Systems,
Vienna University of Technology, Austria
<firstname>.<lastname>@tuwien.ac.at

Abstract. [*Context*] Expert sourcing is a novel approach to support quality assurance: it relies on methods and tooling from crowdsourcing research to split model quality assurance tasks and parallelize task execution across several expert users. Typical quality assurance tasks focus on checking an inspection object, e.g., a model, towards a reference document, e.g., a requirements specification, that is considered to be correct. For example, given a text-based system description and a corresponding model such as an Extended Entity Relationship (EER) diagram, experts are guided towards inspecting the model based on so-called expected model elements (EMEs). EMEs are entities, attributes and relations that appear in text and are reflected by the corresponding model. In common inspection tasks, EMEs are not explicitly expressed but implicitly available via textual descriptions. Thus, a main improvement is to make EMEs explicit by using crowdsourcing mechanisms to drive model quality assurance among experts. [*Objective & Method*] In this paper, we investigate the effectiveness of identifying the EMEs through expert sourcing. To that end, we perform a feasibility study in which we compare EMEs identified through expert sourcing with EMEs provided by a task owner who has a deep knowledge of the entire system specification text. [*Conclusions*] Results of the data analysis show that the effectiveness of the crowdsourcing-style EME acquisition is influenced by the complexity of these EMEs: entity EMEs can be harvested with high recall and precision, but the lexical and semantic variations of attribute EMEs hamper their automatic aggregation and reaching consensus (these EMEs are harvested with high precisions but limited recall). Based on these lessons learned we propose a new task design for expert sourcing EMEs.

Keywords: Review, Models, Model Quality Assurance, Model Elements, Empirical Study, Feasibility Study, Crowdsourcing, Task Design

1 Introduction

During the design of software systems, a variety of models are created in the process of transforming requirements and/or specifications of the desired system into the corresponding software. These models include *Extended Entity Relationship* (EER) diagrams or UML model variants for designing databases and software system structures and behavior. The tasks of creating such models from software specifications and their subsequent verification to ensure their quality, i.e., through software model inspection [1], are cognitively intense tasks, that require significant time and effort investment from

software engineers. In particular, large software models and large associated reference documents, such as requirement specifications, are challenging to inspect with limited resources in one inspection session as overly long sessions typically lead to cognitive fatigue [6]. Thus, a typical inspection session is scheduled for about two hours; large artifacts have to be scheduled in multiple inspection sessions to achieve sufficient coverage of the inspection object. Therefore, reference documents and inspection objects need to be split accordingly. Furthermore, different inspection tasks can be distributed among a group of domain experts for inspection. *Human Computation and Crowdsourcing* (HC&C) mechanisms can help splitting the workload and distributing inspection tasks among a group of experts [9, 10, 11].

In general, HC&C techniques rely on splitting large and complex problems into multiple, small and easy tasks solvable by an average contributor in a suitable population and then coordinating the collection and aggregation of individual micro-contributions into a larger result [8]. As a result, we defined and introduced a novel *Crowdsourcing-based Software Inspection* (CSI) process, previously described in [9, 10, 11] and illustrated in Figure 1.

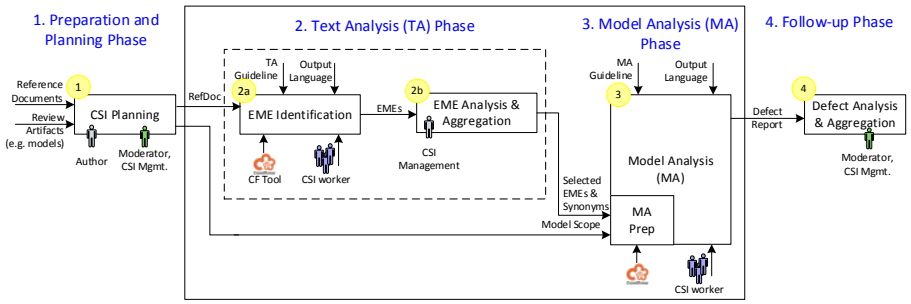

Figure 1: Crowd-based Software Inspection (CSI) process [11].

The CSI process covers the *Preparation* and *Software Inspection* core steps of the traditional inspection process and consists of four fundamental phases: (1) *CSI Preparation and Planning*; (2) *Text Analysis* (TA) of reference documents to identify model building blocks, i.e., Expected Model Elements (EMEs) and aggregation of identified EMEs; (3) *Model Analysis* (MA) to identify candidate defects in the software model; and (4) *Follow-up* for defect analysis and aggregation.

During the *Preparation and Planning Phase* (Phase 1), the moderator performs inspection planning. He additionally fulfills the CSI management role. His tasks include (a) scoping of inspection artifacts, (b) preparing the crowdsourcing environment, and (c) uploading reference documents and inspection artifacts into the crowdsourcing platform, such as *Crowdflower*[1].

The goal of the *Text Analysis (Phase 2)* is the elicitation of key components (i.e., building blocks or co-called Expected Model Elements (EMEs)) of the model based on the reference document. EMEs include entities, attributes, and relations, that are present in system requirements specifications (i.e., reference documents) and need to be modeled in the corresponding software model, e.g., in an EER diagram. Table 1 in

[1] *CrowdFlower*: www.crowdflower.com

Section 4 lists typical EMEs extracted from the description of a restaurant billing system, representing the reference document (i.e., a requirements specification). Key EME classes include correct EMEs, synonyms and variations, and incorrect EMEs (see Section 4 for details). Note that the EME identification phase is supported by a crowdsourcing application. This phase focuses on two sub-phases: (2a) identification of candidate EMEs by a group of inspectors. We experiment with a task design where inspectors are shown one specification sentence at a time and asked to provide, based on an input syntax, the EMEs appearing in that sentence (see Figure 4 for a task screenshot). Results are sets of candidate EMEs provided by every crowd worker in this sub-phase; (2b) EME analysis and aggregation where the moderator compiles an agreed set of EMEs based on individual results. Result is an agreed set of EMEs as input for the model analysis phase of the CSI process (Phase 3). Note that in this paper we focus on phase 2b, i.e., the EME identification and aggregation step.

In the *Model Analysis* phase *(Phase 3)*, inspectors verify the model itself (e.g., an EER diagram), or a subset thereof, while being guided by EMEs. In other words, EMEs are used as anchors for splitting the model verification task into smaller instances that can be solved in a distributed fashion according to HC&C principles. Output of this phase is a set of candidate defect lists provided by individual inspectors during their crowdsourcing task. The role of the CSI management in phase 3 is to prepare the model or a sub-model to be inspected (based on CSI planning definitions). In the *Follow-Up* Phase (Phase 4), the CSI management aggregates reported defects. Output of this final phase is a set of defects as a consensus of individual defect lists. Note that this task is completed by the CSI management.

To investigate the efficiency and effectiveness of defect detection in context of model inspection with CSI, we have conducted a large-scale empirical study involving university students as study participants in the fall of 2016. We have already reported initial findings of the CSI process approach regarding effectiveness and efficiency of defect detection when guided by high quality EMEs. High quality EMEs have been identified by the study authors [9, 10, 11]. In context of the CSI process approach EME identification is the main output of the Text Analysis process phase. In this paper, we turn our attention to the feasibility of this text analysis task. Namely, we want to assess the feasibility of distributing the task of identifying EMEs in textual documents that describe a system specification to a number of experts as opposed to being solved by a single expert. Another key goal is to use lessons learned to improve the design of the text analysis task. For evaluation purposes, we perform a partial analysis of the CSI feasibility study reported in [9, 10] with focus on the text analysis output. We look at the EME's identified in one of 4 sessions of the feasibility study and compare these EMEs to a gold standard created by the study authors (i.e., high quality EMEs used in previous investigations). We conclude that, while entity EMEs can be identified with high precision, precision heavily deteriorates for more complex EME types where more variations are possible. This points to the need of new task designs for collecting EMEs in a reliable manner.

The remainder of this paper is structured as follows: Section 2 presents related work and Section 3 introduces to the research questions. In Section 4 we present the setup of the feasibility study and in Section 5 the experimental results and discussions. Section 6 provides a discussion of the experimental results and reflects on limitations. Section 7 summarizes lessons learned for text analysis task design. Finally, Section 8 concludes and identifies future work.

2 Related Work

HC&C methods have been recently used to solve a diversity of Software Engineering (SE) tasks and lead to the emerging research area of Crowdsourced Software Engineering (CSE) defined as *"the act of undertaking any external software engineering tasks by an undefined, potentially large group of online workers in an open call format"* [4, 5].

The intensified interest in the application of crowdsourcing techniques in SE can be seen as a response to the appearance of mechanized labor (micro tasking) platforms such as *Amazon Mechanical Turk*[2] (AMT) or *CrowdFlower*[3] (CF). These platforms have popularized the Micro tasking crowdsourcing model. Micro tasking differs from more traditional models of distributed work in SE, such as collaboration and peer-production, by being more scalable, thanks to enabling parallel execution of small task units by non-necessarily expert contributors [4]. As such, Micro tasking is promising to address challenges in several cognitively complex software engineering tasks, including software inspection and model element identification. Some of the benefits of the Micro tasking model are [5]: improved coordination (e.g., of inspection team members, tasks, and results), reduction of cognitive fatigue (by removing redundant work and reusing intermediate results from previous steps), increased coverage (as some parts of large artifacts might not be covered with traditional, weakly-coordinated approaches), more diversity (support for dealing with various inspection artifacts and access to a wider variety of inspectors), and accelerated processes (by parallelization of small tasks and access to more inspectors).

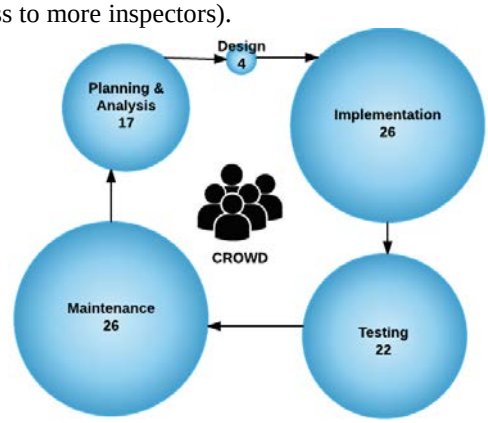

Figure 2: Overview of number of papers classified by Mao et al. [5] as using crowdsourcing for tasks corresponding to each software development life-cycle stage.

Because of these benefits, crowdsourcing has been applied to address a variety of tasks pertaining to all stages of the Software Development Life Cycle (SDLC) as reported by Mao *et al.* in a recent, extensive survey [5]. Based on their classification, we derived an intuitive overview of the intensity of crowdsourcing research in each SDLC stage as shown in Figure 2. Seventeen (17) papers report work within the

[2] Amazon Mechanical Turk: www.mturk.com
[3] *CrowdFlower*: www.crowdflower.com

Planning and Analysis phase on tasks such as requirements acquisition, extraction and categorization. The problems from the *Design* phase have attracted less approaches, with only four papers attempting crowdsourced user interface and architecture design. *Implementation* phase specific tasks such a coding and debugging have been investigated in 26 papers. Problems that were crowdsourced from the *Testing* phase were reported by 22 papers and included usability, performance and GUI testing. Within the *Maintenance* phase, crowdsourcing was used for software adaptation, documentation and localization among others, attracting a similar volume of approaches as the implementation phase. None of the investigated papers cover static quality assurance approaches such as software inspection, where the CSI process provides benefits for early defect detection in software models [9, 10, 11].

However, extracting higher-level conceptual elements from text with HC&C techniques has been investigated before. For example, *Cascade* [3] is an approach to build classification hierarchies (i.e., taxonomies) from collections of independent text items (e.g., textual answers from question answering sites*).* An important step is to identify categories that best describe each text item. *Cascade* is designed to run on collections of independent items and it would not be suitable to extract higher-level conceptual elements (e.g., entities, relations) from ordered, logically-connected sequences of items such as sentences in a systems description document.

Approaches that can extract conceptual elements from interconnected item sets include *PhraseDetectives* [7], for co-reference resolution in text corpora, or, work on extracting categories in interconnected texts [2]. Unlike previous works, André *et al.* also focuses on tasks where domain expertise is desirable and shows that non-expert users can be supported in making sense of specialized texts [2]. This is an important aspect in the context of software inspection, which is more amenable to be solved by experts through *expert sourcing* (i.e., enlisting experts to solve tasks by using HC&C principles) rather than layman crowds.

In summary, a variety of HC&C approaches have been proposed to solve a wide range of SE tasks in all phases of the SDLC. Yet, the software model inspection task, situated within *Planning and Analysis* and *Design* SDLC phases, has not been addressed. Our work focuses on this gap and explores how HC&C micro tasking principles can be used to improve software inspection processes through expert based crowdsourcing (i.e., expert sourcing).

3 Research Questions

With focus on EME identification in context of the CSI process approach, the main research questions focus on:

RQ1: What is the overall quality of the CSI-based EME identification? To what extent can this distributed process lead to a high-quality set of EMEs? In the study context, we consider the EMEs provided by the study authors as a gold standard example of high quality EMEs.

RQ2: What are performance differences when identifying different types of EMEs? What challenges arise when distributing the identification of EMEs of increasing structural complexity? For the purpose of this paper, we define the structural complexity

of an EME as the number of relations between atomic entities. For example, entities (e.g., customer) are the least complex EMEs as they refer to a single concept. Entity attributes such as *customer.name* establish a binary relation between an entity and its attribute and therefore have a higher structural complexity than entity EMEs. Relations, e.g., *(customer, orders, order)* relate three atomic entities in a triple being thus more complex than entity attribute EMEs. Relation attributes and relation cardinalities are the most structurally complex EMEs as they express more knowledge that relation EMEs.

RQ3: What are alternative expert sourcing task designs? Based on lessons learned from the analysis of the collected data, we want to identify new task design strategies that could address the limitations of the current design and lead to better results.

4 Feasibility Study Setup

The analysis reported in this paper relies on data collected in a large-scale feasibility study of the CSI-process which was described in [9-11]. Figure 3 shows the overall experiment design with three study groups: Study group A conducts two stages of the CSI process, i.e., a text analysis and a model analysis; Study group B performs similar tasks in a different sequence; finally Study group C performs a traditional inspection process as a control group. All study groups received a tutorial at the beginning of the experiment run. Because the main goal is to investigate the effects on eliciting Expected Model Elements (EMEs) we focus on Study group A and B in this paper (Study group C does not include a task to explicitly identify and report EMEs – thus, we excluded this study group in the rest of the paper). Note that analysis results of all study groups have been described in [9-11].

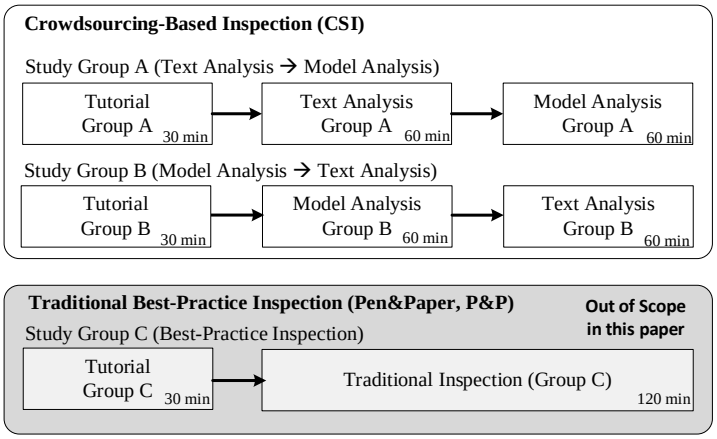

Figure 3: Feasibility Study Setup [9].

In context of this paper, the *study design* for EME elicitation consists of two main groups A and B performing the two stages of the CSI process. For group A and B, we applied a cross-over design, i.e., text analysis (60 min) followed by the model analysis (60 min) for group A and similar tasks in an inverse order for group B. This cross-over design was chosen in order to address one of the goals of the overall study, namely to assess the impact of previous domain knowledge on the defect detection performance of

the inspectors. In other words, we want to compare defect detection performance for inspectors that have read the specification beforehand (group A) with the performance of inspectors that detect defects only based on the EMEs and the model (group B). The time inspectors spent on each task was limited to one hour per task to (a) investigate the impact of splitting larger tasks into smaller pieces of work – a key capability of HC&C approaches, (b) enable efficient defect task execution (prior pilot runs showed that 1h is sufficient to enable efficient EME identification and defect detection), and (c) to have a manageable experiment duration. Note that the size of the inspection artifact under investigation is suitable for a two hour inspection task, evaluated during pilot runs.

Common to all study groups is a tutorial (30 min) related to the method applied including a small practical example to get familiar with methods under investigation and related tool support. For the model analysis we used a pre-defined set of EMEs to avoid dependencies between different tasks within the experiment groups. These EMEs were provided by the experiment team, i.e., the authors. We used different experimental material for the tutorials (application domain: parking garage scenarios) and the experiment (application domain: restaurant scenarios).

Study Population. The study was an integral part of a university course on "Software Quality Assurance" with undergraduate students at Vienna University of Technology. We captured background experience in a questionnaire before the study. Because most of the participants work at least part time in industry, we consider the participants as junior professionals [11]. We applied a class-room setting with an overall number of 75 participants (63 crowd workers and 12 inspectors) which attended 4 experiment workshops organized on different days. The group assignment was based on a random distribution of participants to study groups. Overall we had 12 study groups (four groups A, four groups B, and four croups C) during the 4 experiment workshops. Note that group C has been excluded from the analysis in context of this paper.

Study Material. We used study materials from a well-known application domain, i.e., typical scenarios and processes of a restaurant to avoid domain-specific knowledge limitations. The core study material were (a) a *textual reference document*, i.e., a system requirements specification including 3 pages in English language and consisting of 7 scenarios and (b) an *Extended Entity Relationship* (EER) diagram including 9 entities, 13 relationships, and 32 attributes. Furthermore, we used an experience questionnaire to capture background skills of the participants and feedback questionnaires after each step of experiment process. Finally, we provided guidelines that drove the experiment and the inspection process.

Gold Standard EMEs. The authors of the study identified 110 EMEs representing entities, relations, entity and relation attributes as well as relation multiplicities, which together constitute a set of *Gold Standard* EMEs (see some examples of EMEs from the Gold Standard in Table 1). For each type of EME, authors identified 4 categories:

1. *Correct* EMEs as those that appear in the EER model (EME Type 1).
2. *Synonyms* and *lexical variants* of correct EMEs (EME Type 2).
3. *Correct variants of modeling that are different from the input model*, i.e., the textual requirements specification (EME Type 3).
4. *Wrong* EMEs which should not be part of a correct model (e.g., superfluous) (EME Type 4).

Tooling. The *CrowdFlower* application has been used to drive the text and model analysis tasks. For the *text analysis*, the specification document was divided in sentences and participants were asked to identify EMEs in one sentence at a time. The corresponding *CrowdFlower* task contained as set of instructions and a form-based input space. The instructions explained the task in general, provided an introductory paragraph about the system that needs to be built, contained EME examples including an explanation of their specification syntax as well as some examples of sentences (from another domain) and the EMEs extracted from those sentences. The instructions are available for each task, but the instruction pane can be minimized for optimal viewing. The data collection space listed a sentence and provided forms for collecting the different EME types. Under each form, instructions were repeated about the expected naming convention and syntax.

Table 1: Example EME's from the Gold Standard.

EME Type 1: "Correct EMEs as those that appear in the EER model"	
Entities:	customer; order; invoice; setMenu
Relations:	(customer, orders, order) (order, orderedFoodItem, foodItem)
Entity / Relation Attribute	customer.name invoice.sum (customer, orders,order).date (ingredient, isProcuredBy, shoppingTour).price
Relation Cardinalities	(customer(1), orders,order(0..n)) (order (0..n), orderedFoodItem, foodItem (0..n))
EME Type 2: "Synonyms and lexical variants of correct EMEs."	
Entities:	person; menu
Relations:	(foodItem, partOf, order) (order, has, foodItem
Entity / Relation Attribute	Invoice.amount order.date (shoppingList, contains, ingredient).price
Relation Cardinalities	(customer(1), orders, order(n))
EME Type 3: "Correct variants of modeling different from the input model"	
Entities:	dailyPlan; payment
Relations:	(foodItem, has, foodItemPrice)
Entity / Relation Attribute	order.status invoice.paid_date (customer, orders,order).status
Relation Cardinalities	customer(1), cancels, order(0..n))

EME Type 4: "Wrong EMEs which should not be part of a correct model"	
Entities:	cook; restaurant; guest
Relations:	(order, contains, ingredient)
Entity / Relation Attribute	food.calorie table.seats (order, has, ingredient).amount
Relation Cardinalities	(customer(1..n), orders,order(1))

Table 1 presents examples for identified EMEs based on the identified categories. Figure 4 illustrates and example implemented in the *Crowdflower* User Interface to identify EMEs as a crowdsourcing task. Note that we provided a so-called "Output language", i.e., a data format that unifies the data input from experiment participants.

Figure 4: CrowdFlower task interface (form-based data collection).

5 Experimental Results

For the initial data analysis process we focus on the evaluation of data collected within one of the 4 experiment workshops by groups A and B participating in this workshop. Individual workshops were self-contained events and had similar numbers of participants as depicted in Table 2; the 6 participants of Group A received 14 sentences to process while the 7 participants of Group B worked on 17 sentences.

Table 2. Overview of experimental results in terms of identified EMEs (before / after aggregation) as well as precision & recall with respect to gold standard EMEs.

	Group A	**Group B**	**Group A+B**
Number of sentences	14	17	31
Group participants	6	7	Not relevant
Entity EMEs			
All (unique) entity EMEs	201 (42)	239 (43)	440 (76)
Entity EMEs from all sentences (unique)	25 (11)	37 (13)	62 (17)
Precision	91%	92%	88%
Recall	88%	88%	100%
Entity and Relation Attribute EMEs			
All (unique) attribute EMEs	137 (104)	238 (171)	375 (270)
Attribute EMEs from all sentences (unique)	17 (14)	28 (26)	45 (37)
Precision	100%	100%	100%
Recall	15%	43%	59%
All (unique) relation EMEs	114 (97)	168 (145)	282 (240)

Table 2 also contains statistics about the actual numbers of EMEs collected from all participants before applying algorithms to aggregate individual contributions to each sentence as described in sections 5.1 and 5.2 (see corresponding Aggregation heading). Overall, when merging data from groups A and B, we collected 440 individual entity EME suggestions, which correspond to 76 syntactically unique entity EMEs. Entity and relation attributes accounted to 375 EMEs, out of which 270 were syntactically unique. Lastly, we obtained 282 relationship EMEs, with 240 syntactically unique values. An immediate observation here is that the number of recommended entities decreases as EMEs get more complex. At the same time, the percentage of unique EMEs from all EMEs is decreasing which is due to the following two factors: (1) syntactic heterogeneity increases with more complex EMEs and therefore they cannot be aggregated with syntactic string matching; (2) there is also an increased modeling heterogeneity, as participants might adapt different modeling approaches for the same information, thus hampering reaching consensus.

5.1 Identification of ENTITY Model Elements

This section discusses the analysis of the entity type EMEs by both groups A and B.

Aggregation. The aggregation of EMEs suggested by individual participants took place at the level of each sentence. From the entity EMEs provided by all participants in the group for a given sentence, only the EMEs that were recommended by more than half of the participants were selected. In practical terms, we counted the frequency of each entity EME and computed a popularity score by dividing this frequency to the number of participants that rated the corresponding sentence. EMEs with popularity score higher than or equal to 0.5 were selected. To identify the EME output of a group, we created the union of sentence level entity EMEs, and selected the unique EMEs from these. Note that, in the case of entity EMEs, recommended EMEs were not pre-processed neither syntactically nor semantically. For example, we did not correct typos and did not stem the EMEs from plural to singular form. This means, that the EMEs "ingredient" and "ingredients" were treated as two syntactically different EMEs. For the final set of EMEs, we mapped these to the gold standard to determine their type (i.e., 1- correct EME, 2-synonym EME, 3-alternative EME, and 4-incorrect EME). Based on this mapping we computed the precision of the final EME set as the ratio between the EMEs of type 1-3 (i.e., correct of syntactic/semantic variations) and all identified EMEs.

For group A of the investigated workshop, a total of 25 EMEs were derived from 14 sentences based on contributions from 6 participants, corresponding to 11 unique EMEs. Based on the alignment to the gold standard we obtained precision of 91% (only 1 of 11 entities was wrongly identified). Recall was computed with respect to the 9 EMEs of type "1" (i.e., those EMEs that are part of the EER diagram). All EMEs were identified except "recipe", leading to a recall of 88%.

Group B, received more sentences therefore lead to the identification of more sentence level EMEs (37), corresponding to 13 unique EMEs in the final EME set, a precision value of 92% and recall of 88% (as in the case of group A, a single EME from the gold standard was not found, namely "invoice").

Joining the results of the two groups to obtain the EMEs for the entire reference document, we obtain a precision of 88% and a recall of 100%. Interesting observations to be drawn from here are as follows. Almost the entire set of entity EMEs was identified by each group although each group received just half of the sentence corpus (due to the cross-over design), since key entities are mentioned through the sentences. Exploiting such repetition could lead to more elegant task designs for crowdsourcing entity identification.

5.2 Identification of Entity and Relation ATTRIBUTE Model Elements

This section discusses the analysis of the entity attribute and relation attribute type EMEs collected from both groups A and B.

Aggregation. In the case of entity and relation attributes, participants contributed a large variation of syntactically diverse EMEs. Indeed, in group A, from 137 contributed EMEs (100 entity attributes, 37 relation attributes) contributed by all participants for

all sentences, 104 were unique EME strings (73 entity attributes, 31 relation attributes). Therefore, given this high syntactic variation, an automatic aggregation per sentence as performed in the case of entity EMEs was not possible. Instead, we have inspected the proposed EMEs per sentence and replaced syntactic variations of the same EME to allow for aggregation. For example, for sentence Sc6S1 (the first sentence in scenario 6), participants contributed EMEs such as: *order.fullfilled*, *foodOrder.isFulfilled*, *order.fulfilledAsPlanned* which were all replaced with their semantic equivalent *order.fulfilled?*. Aggregation in terms of majority voting was performed then on the new set of EMEs. As for entity EMEs, the agreement threshold was kept at 0.5.

For Group A, a total of 17 attribute EMEs resulted from the sentence level majority voting, 14 of these were unique. All resulting EMEs were of type 1, 2, or 3, therefore the precision was 100%. However, when compared to the proposed gold standard of 39 attribute EMEs of type 1, only 6 of these were identified, leading to a recall of 15%.

From Group B, 28 attribute EMEs were collected, 26 of these being unique and all of type 1, 2, or 3. Therefore precision was 100%, while recall was 43% (17 of the identified EMEs corresponded to group 1 EMEs from the gold standard).

When merging the output EME sets from groups A and B, the EME set contains 45 EMEs out of which 37 are unique. We observe here that, unlike in the case of the entity EMEs, the overlap between the attribute EMEs derived by the two groups is quite low, as different parts of the specification point to different attributes. As a consequence, the recall of the overall EME set is 59% (23 EMEs correspond to EMEs from the gold standard).

6 Discussion and Limitations

This section focuses on the discussion of individual results towards the introduced research questions and addresses limitations and threats to validity.

6.1 Discussion

Answering our first research question (*RQ1: What is the overall quality of the CSI-based EME identification?*), we conclude that the process of distributed identification of EMEs following HC&C principles is feasible and leads to a set of EMEs that have a high overlap with a manually created gold standard set of EMEs.

Our second research question, RQ2, focused on: *What are performance differences when identifying different types of EMEs?* Based on our analysis, we conclude that the performance of verifying EMEs varies depending on the EME type:

- Entity type EMEs were identified with a high recall (100%, see Table 2) and precision (88%). We also observed that, since EME's are mentioned through the specification, a reliable set of EMEs can be extracted even just from half of the specification by either of the two study groups. This prompts to the possibility to

change task design in order to more efficiently use human processing when identifying entity EMEs.

- Attribute (and relation) type EMEs were identified with a high precision (100%), but with a low recall (59%) because the syntactic and semantic variations of these EMEs are considerably higher than for entity EMEs. Because of this syntactic heterogeneity, free-text collection makes aggregation with automatic tools not feasible and opens the need for more guided task design to enable such automation in the future.
- The number of proposed EMEs decreased as the complexity of the EMEs increased. That is, while many entity EMEs were proposed (404), participants provided less entity attributes or relation attributes (375) and relations (282). Potential reasons for this phenomenon are: (1) the cognitive overload of the task is too high, as the participants are asked to do too much and therefore naturally provide less input for the more complex parts of the task; (2) writing down complex EMEs such as relations is too time-consuming and requires high effort so participant avoid it.

From the overall CSI-process perspective, the high precision of the entire EME set means that this set can act as a reliable guidance for defect detection in the follow-up model analysis task. To further improve the EME detection task and to answer RQ3 (*What are alternative expert sourcing task designs?*), we propose a novel text analysis task design in Section 7.

6.2. Limitations and Threats to Validity

Internal validity concerns a causal relationship between the experiment treatment and the observed results, without the influence of potential confounding factors that are not controlled or measured [12]. Domain specific issues have been addressed by selecting a well-known application domain. The experiment package was intensively reviewed by experts to avoid errors. Furthermore, we executed a set of pilot runs to ensure the feasibility of the study design [9, 10, 11].

In terms of *external validity*, that is the generalization of the results to a larger population or to a different environment [12], we distinguish the following threats: Participants were 75 undergraduate students of computer science and business informatics at Vienna University of Technology. The study was a mandatory part of the course on “Software Quality Assurance”. Most of the participants work at least part-time in software engineering organizations. Thus, we consider them as junior professionals comparable to industrial settings. We used an experience questionnaire to capture and assess prior experiences and skills. Application domain. We used typical scenarios and requirements derived from restaurant processes. Thus, all participants are familiar with this application domain. Quality of specification documents. Admittedly, we have used a high quality specification document which has been carefully reviewed by the experiment team, i.e., the authors, during the preparation for the study. In addition we executed pilot runs of the study to ensure the quality and understandability of the study material.

7 Lessons Learned for Text Analysis Task Design

Lessons learned from this initial analysis of EMEs collected in a distributed manner prompt to several improvements at task design level, as follows (and in response to RQ3):

- *Process Improvement: Introduce a workflow of simpler tasks*. One way to overcome the complexity of the current task which hampers the acquisition of more complex EMEs, is to divide the task into a workflow of smaller tasks, each focusing on the acquisition of one EME type only.

- *Tooling: Replace free-text input tasks with more guided interfaces*. The structural complexity of the collected EMEs has an influence on two key factors. First, the higher input effort discourages participants from adding these EMEs, so less EMEs are collected. Second, the syntactic and semantic variation increases in such manner that reaching a consensus and automatic aggregation of these EMEs to allow majority voting become challenging and in some cases unfeasible. Replacing free-text input fields with more guided interfaces could overcome these limitations and (1) help EME aggregation while (2) fostering consensus building.

- *Reduce redundancy*. In the case of entity EMEs, these are frequently mentioned in several sentences of the specification and therefore lead to the redundant identification of EMEs at the cost of the participants' time and effort. Task design for entity EMEs should take this into consideration and try to reduce the redundant identification of EMEs.

Based on these envisioned improvements, we propose an updated workflow for distributed identification of EMEs, as shown in Figure 5. The process starts with a collection of sentences from the specification which serves as input to the *Entity Identification* task to identify entity EMEs. To reduce redundancy of EME identification, we envision a task design where an initial set of EMEs are identified and aggregated from a subset of the sentences and then this set is subsequently approved and if necessary extended by participants.

Even with the current design (see Figure 1), entity identification leads to a high quality and high coverage entity EME set. This can be used as an input to subsequent tasks for entity attribute and relation identification.

The *Entity Attribute Identification* task will consist of participants receiving a sentence and an entity EME in this sentence and being asked to enumerate attributes of that entity EME. For entities that appear in more sentences, two approaches could be considered. Firstly, if a high EME set coverage is desired, then this EME should be shown with all sentences it appears in as different sentences are likely to describe different attributes of the entity EME. Secondly, if fast execution is more desired than good coverage, the sentence to be shown could be determined according to the voting score obtained. In other words, a sentence where an entity EME was selected by many participants would be considered as most suitable for that EME and the EME would be shown only in combination with that sentence.

Each *Relation Identification* task would receive a sentence with a set of entity EMEs in this sentence and could be designed in diverse ways. In the first variant, participants could be asked to provide those entity pairs between which a relation exists as free text input following a recommended syntax (with or without naming the relation). In the second variant, all combinations of entity EME pairs from a sentence could be generated and participants asked to select only those pairs between which a relation holds. This second variant has a lower effort on participants and results are easier to integrate based on selections as opposed to free-text inputs from the first variant.

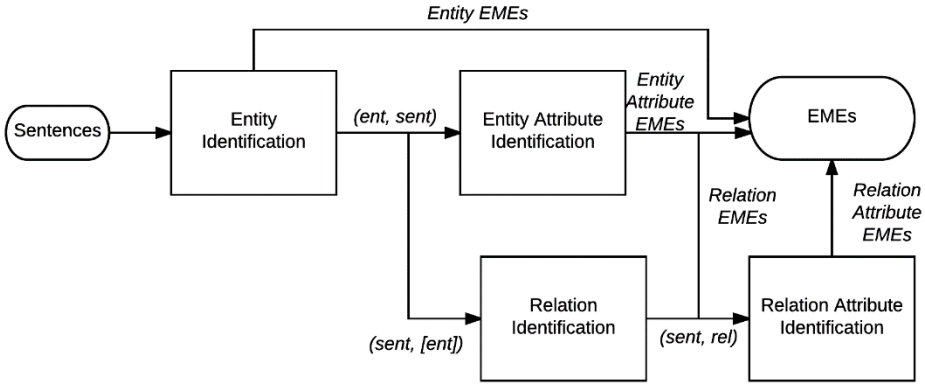

Figure 5: Proposed workflow for distributed EME identification.

Lastly, the *Relation Attribute Identification* task would benefit from the output of the Relation Identification task. For a given sentence and relation in this sentence, participants should identify potential relation attributes described in the sentence. The proposed and improved workshop process (see Figure 5) aims at guiding participants and foster reaching consensus to a set of EMEs that can be subsequently used as input to the Model Analysis task of the CSI process. While inspecting the data, we identified in the collected responses diverse modeling choices, such as the following examples:

- *(customer(1) , orders , order(0..n)).isTakeout* versus *order.isTakeout;*
- *(customer(1) , has , order(0...n)).orderNumber* versus *order.number;*
- *(setMenu , contains , foodItem).price* vs. *setMenu.price* and *foodItem.price.*

With the current design, less popular modeling choices will be lost in favor of those that the majority agrees on. Yet, harvesting diverse ways to model the same specification could be beneficial for other contexts, for example when building a model rather than verifying a given model. In this context, a crowdsourcing process could identify, from a specification different models that capture a variety of views. For that purpose, new workflows should be considered that focus on preserving modeling diversity as opposed of building consensus.

8 Conclusion and Future Work

In this paper, we focus on interpreting results from the *Text Analysis* step of the *CrowdSourcing-based* Inspection (CSI) process [9, 10] which supports early defect detection of large-scale software engineering artifacts and models based on textual system specifications. The goal of the *Text Analysis* step is the distributed identification of *Expected Model Elements* (EMEs) within the system specification. EME terms represent important model elements derived from a reference document and are used as an input in the subsequent *Model Analysis* step of CSI where the defect detection in software engineering models takes place.

We analysed a subset of the data collected in a large-scale feasibility study of CSI, focusing on the comparison of the EMEs collected in a distributed fashion with a gold standard set of EMEs created by a single expert. Our key finding was that the text analysis task can lead to a highly precise set of EMEs which can serve as a suitable input for the defect detection tasks in the *Model Analysis* phase of CSI. However, while the precision of EME set is high, its recall is low for EMEs other than entities. Indeed, when harvesting more complex EMEs (e.g., entity and relation attribute), study participants provided a wide range of syntactically diverse EMEs. These leads to two difficulties. First, automatic result aggregation is challenging. Second, even after manually mapping syntactic variants of EMEs to a base form, we noticed difficulties in reaching a consensus as participants provided different EMEs based on diverse modelling views.

We concluded that these limitations are a side-effect of the current task design, which is too complex and not sufficiently guided to help consensus building. Therefore, we propose a new task design which is based on a workflow of simpler tasks, each focusing on the acquisition of one entity type. Also, we propose that these tasks are more guided: we start with the acquisition of entity EMEs which can be acquired with high recall and precision and use these to guide the following tasks. To support automatic aggregation of results, where possible, we replace free-text input with selection from a set of EMEs derived from previous steps.

Future Work. This paper reports on preliminary findings after initial manual analysis of a subset of the data. Lessons learned from this manual stage will serve to define automatic processing scripts that can be used to (semi-)automatically interpret the rest of the data and more reliably verify the results of this paper. Follow-up experiments will also test the newly proposed task design with respect to improving the effectiveness of EME identification. We will also apply the results of this work to the task of model acquisition from textual specifications. In particular, here we are interested in investigating diverse types of workflows, namely those that foster EME set diversity as opposed to fostering consensus.

Acknowledgments.

We would like to thank the participants of the software quality course at Vienna University of Technology in the winter term 2016/2017 for participating in the study.

References

1. Aurum A., Petersson H., Wohlin C.: State-of-the-Art: Software Inspection after 25 years, In: Journal of Software, Testing, Verification and Reliability, 12(3), pp.133-154, 2002.
2. André P., Kittur A., Dow S.: Crowd Synthesis: Extracting Categories and Clusters from Complex Data, Proc. Conf. Computer Supported Cooperative Work (CSCW), pp. 989-998 2014.
3. Chilton L.B. Little G., Edge D., Weld D.S., Landay J.A.: Cascade: Crowdsourcing Taxonomy Creation, Proc. Conf. on Human Factors in Computing Systems (CHI), pp. 1999-2008, 2013.
4. LaToza T.D., van der Hoek A.: Crowdsourcing in Software Engineering: Models, Motivations, and Challenges, IEEE Softw. 33(1):74-80, 2016.
5. Mao K., Capra L., Harman M., Jia Y.: A survey of the use of crowdsourcing in software engineering; In: Journal of Systems and Software, 28p., 2016.
6. NASA: Software Formal Inspection Standards, NASA-STD-8739.9, NASA, 2013.
7. Poesio M., Chamberlain J., Kruschwitz U., Robaldo L., Ducceschi L.: Phrase detectives: Utilizing collective intelligence for internet-scale language resource creation. ACM Trans. Interact. Intell. Syst. 3(1), 44p., 2013.
8. Quinn A., Bederson B.: Human Computation: A Survey and Taxonomy of a Growing Field, In Proc. of Human Factors in Computing Systems (CHI), pp.1403-1412, 2011.
9. Winkler D., Sabou M., Petrovic S., Carneiro G., Kalinowski M., Biffl S.: Investigating Model Quality Assurance with a Distributed and Scalable Review Process, In: Proceedings of the 20th Ibero-American Conference on Software Engineering, Experimental Software Engineering (ESELAW) Track, Springer, Buenos Aires, Argentina, 2017.
10. Winkler D., Sabou M., Petrovic S., Biffl S., Kalinowski M., Carneiro G.: Improving Model Inspection with Crowdsourcing, In: Proceedings of the 4th International Workshop on Crowdsourcing in Software Engineering, ACM/IEEE International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, 2017.
11. Winkler D., Sabou M., Petrovic S., Biffl S., Kalinowski M., Carneiro G.: Improving Model Inspection Processes with Crowdsourcing: Findings from a Controlled Experiment. In: Proceedings of the 24th European System, Software and Service Process Improvement and Innovation (EuroSPI), 2017.
12. Wohlin C., Runeson P., Höst M., Ohlsson, M., Regnell, B. Wessl, A.: Experimentation in software engineering. Springer (2012).