Managing the Complexity in Ethical, Social and Environmental Accounting: Engineering and Evaluating a Modelling Language

Vijanti Ramautar[1], Sergio España[1]

[1]Utrecht University, Princetonplein 5, 3584 CC Utrecht, The Netherlands

Abstract
Assessing, reporting, and monitoring ethical, social and environmental is a key practice for sustainable business innovation. There are a plethora of methods that guide these assessments. Often these methods are supported by an ICT tool. In most cases, the tools are developed to support a single method only and do not allow any tailoring. Therefore, they are rigid and inflexible. To mitigate the risk of managerial problems, reporting fatigue, loss of confidence in sustainability practices, and to manage complexity in ESEA methods we offer a new model-driven approach. We have developed an open-source, model-driven, versatile tool, called openESEA. OpenESEA parses and interprets textual models, that are specified according to a domain-specific language (DSL). This article reports on a new iteration of the creation process of our modelling language, describes the most important modelling primitives of the DSL, and reports on the validation of the DSL through user testing.

Keywords
Organisational sustainability, model-driven engineering, domain-specific language, modelling language complexity, ethical social and environmental accounting, sustainability reporting

1. Introduction

Stakeholders are increasingly interested in the ethical, social and environmental (ESE) performance of organisations. For minority shareholders, perceptions of poor credibility and poor corporate social responsibility performance even result in a higher tendency to read sustainability reports [1]. Also, external stakeholders (clients, business partners, citizens) sometimes put pressure on organisations to disclose their sustainability reports publicly. To increase their ESE performance, organisations usually apply a continuous improvement cycle [2]. This article focuses on the second process of the continuous improvement cycle, the ethical, social and environmental accounting (ESEA) process. During the ESEA, organisations assess their performance in material ESE topics [3, 4]. To do this, first, an ESE accountant collects data from organisational stakeholders via surveys or extracts it from information systems. Examples of such data (i.e. direct indicators) are the monthly electricity consumption, or the number of men, women, and non-binary people in managerial positions. This data allows calculating

BIR 2022 Workshops and Doctoral Consortium, 21st International Conference on Perspectives in Business Informatics Research (BIR 2022), September 20-23, 2022, Rostock, Germany
v.d.ramautar@uu.nl (V. Ramautar); s.espana@uu.nl (S. España)
0000-0002-3744-0013 (V. Ramautar); 0000-0001-7343-4270 (S. España)

indirect performance indicators such as the percentage of women (and non-binary people) in managerial positions and annual electricity consumption. We refer to the set of indicator values collected by conducting an ESEA process as ESE account. Parts of this account are typically published in a sustainability (or non-financial, or integrated) report.

Several factors make ESEA methods complex from the process and ICT perspectives. The ESEA domain abounds with methods, standards and frameworks. Many ESEA methods are supported by ICT tools. Most of these tools are rigid and can solely be used to assess the single ESEA method they were developed for [5]. However, ESEA methods usually overlap in the indicators they require input for; e.g. many ESEA methods ask for the number of employees per gender, the minimum salary, and annual water consumption. We found that numerous organisations apply multiple ESEA methods, so given the rigidity of the tools, these organisations end up having to use several disconnected tools that ask them for the same data. Furthermore, organisations would often like to extend or tailor the methods to their needs but tools do not allow it.

To reduce the complexity of managing (i.e. defining and applying) ESEA methods we are engineering the openESEA framework. It consists of a domain-specific language (DSL) that allows modelling ESEA methods, and an interpreter tool that allows organisations to execute the methods. At the core of the framework lies the openESEA metamodel, which serves as an ontology that defines the main primitives of ESEA methods. The metamodel also constitutes the abstract syntax of the openESEA DSL. The concrete syntax is specified with a textual grammar that is used to model ESEA methods. We have operationalised the framework by means of an open-source, web-based, model-driven tool called openESEA. It can be configured by loading a textual model of an ESEA method; then the tool will automatically support the method application. Our approach allows organisations to not only apply existing ESEA methods, but also extend methods, combine them, or create new ones from scratch, using the DSL, without having to worry about developing or updating the tool support.

In earlier work, we presented a first version of our framework [5]. Since then, we have made improvements to the DSL and the interpreter. Regarding the DSL, we have extended the initial version with primitives to model surveys, and we have switched from an Extended Backus Naur Forma (EBNF) grammar with no editing tool support to an implementation in Eclipse Xtext [6] with its corresponding model editor. Regarding the interpreter [1], we have extended it to properly parse the models created with the new grammar, and we have re-implemented it completely to switch from a technology based on the React framework (interface and application tier) and Firebase Firestore, Authentication and Hosting services (back end), to a technology based on the Vue.js framework (interface tier), the Python-based Django framework (application layer), and Heroku Authentication and Hosting services (back end). In both versions, users can load ESEA method models that will be parsed with a JSON schema. However, the new version of the openESEA interpreter presented in this paper also allows specifying such methods through its interface. While in the following sections we touch upon all these improvements, the paper places the focus on the engineering and evaluation of the modelling language for specifying ESEA methods. We explain the DSL development process, the most important primitives of the grammar, and we identify potential improvements of the grammar through a user test. The

[1]Older version: https://github.com/sergioespana/openESEA; newer: https://github.com/sergioespana/open-sea

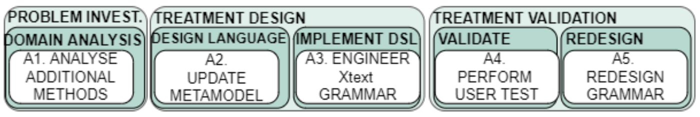

Figure 1: An overview of the research activities

contributions of the paper are (i) new, more mature versions of the DSL and (ii) the model interpreter, which practitioners and researchers can use to assess organisational sustainability; also, (iii) an evaluation of the DSL through user testing. The overall research objective is reduce the complexity of using the openESEA DSL to model ESEA methods.

Section 2 lists the research questions and describes the research method. Section 3 contains conceptual background on ESEA. Section 4 explains how we decided on the extensions of the modelling language. In Section 5 we present the modelling language for specifying ESEA methods. Section 6 explains the user test protocol, its results, and potential improvements of the grammar. The main findings, limitations and future work can be found in Section 7. At last, this article concludes in Section 8.

2. Research method

The two main research questions that are answered in this article are the following.

1. What modelling primitives are needed to specify ESEA methods that are intended to assess organisational sustainability?
2. How can the complexity of modelling ESEA methods be assessed and, if possible, reduced?

The goal of the first question is to find a balance between the expressiveness and complexity of the DSL. The motivation behind the second question is to define a protocol to evaluate the complexity experienced by modellers and to identify potential improvements of the DSL.

Since we aim to produce and evaluate the grammar for specifying ESEA models, produce knowledge around that grammar (e.g. its strengths and limitations), and aim to understand the complexity of creating ESEA models, we apply Design Science [7]. With respect to the language development, we follow conventional DSL engineering practices [8]. Fig. 1 shows the research method. We use the metamodel, EBNF grammar, and model interpreter from [5] as input for this research. We refer to these artefacts as metamodel V1, openESEA EBNF grammar V1, and openESEA interpreter V1, respectively. By executing the research method we aim to create new, more versatile versions of the metamodel and the grammar (i.e. V2).

Problem investigation. While we use the issues for improvement defined in [5] as a starting point, we perform another iteration of the research method in [5] to discover additional points of improvement. That is, in activity A1 we analyse additional ESEA methods. We model these ESEA methods using the Process Deliverable Diagram (PDD) notation [9]. We validate the PDDs with experts in the respective ESEA methods. After the validation interviews, we update and improve the PDDs, if necessary. The validated PDDs are used to create activity and concept comparisons, applying the method comparison approach [10]. **Treatment design.** Based on the concept comparison (output of activity A1) and the openESEA metamodel V1, we derive new (and update existing) ESEA method metaclasses (activity A2), resulting in metamodel V2. Metamodel V2 is, like its predecessor, a UML Class Diagram[11] and it specifies the data structure of an ESEA method and its applications. Metrology standards[12] also informed our design decisions. In activity A3 we engineer a textual Xtext grammar [6] based on the metamodel. While automatic transformation frameworks from (Ecore) metamodels to Xtext grammars exist, we decided to implement the textual grammar manually to have more control over the result. In this version of the DSL, we have opted for a textual grammar. For future versions we plan to create a diagrammatic DSL and perform user tests to find out which option is more preferable. **Treatment validation.** We run a user test (activity A4) to validate the grammar and to discover potential redesigns that would improve the modelling experience. Section 6 explains the user test design, which is based on the Method Evaluation Model (MEM)[13], and analyses the user test results. After completing the user test, we redesign the grammar (activity A5).

3. Background

The conceptual development of ESEA is attributed to Gray [14]. Over time numerous ESEA methods have been developed. These methods provide guidance and instructions on how to perform ESEA. Often, the methods prescribe a set of ESE topics that should be disclosed and define a procedure to successfully assess and report on these topics. Given that sustainability is a multifaceted concept, it is not directly measurable and therefore requires a set of indicators to measure performance [15]. Hence, ESEA methods usually refine topics further into a set of organisational sustainability performance indicators. Examples of ESEA methods are the Sustainability Tracking, Assessment & Rating System (STARS)[2], the B Impact Assessment[3] used by certified B Corporations, and the Common Good Balance Sheet prescribed by the Economy for the Common Good[4]. Some ESEA methods, such as the Global Reporting Initiative Standards[5] and Integrated Reporting framework[6], lay focus on establishing sustainable reporting guidelines, rather than formulating an approach for measuring and assessing ESE performance [16]. There are several reasons for performing ESEA. For instance, ESEA can be performed to address concerns from the public [17] or to obtain a specific certification [18]. Additionally, ESEA can improve business performance. There has been empirical evidence that shows that ESEA method certifications and CSR disclosures have a positive effect on organisations' financial performance [19, 20]. Küchler and Herzig found that ESEA methods that can be applied by organisations in any industry sector, do not always cover the necessary industry-specific indicators [21]. This phenomenon is one of the drivers for organisations to apply multiple ESEA methods (industry-specific and non-industry-specific), justifying the need for versatile ICT infrastructure.

There are prior works that have focused on managing the complexity of models and modelling languages. For instance, [22] defines a modularisation approach for large models. however, this paper focuses on evaluating and improving the user performance while understanding, updating

[2]https://stars.aashe.org/about-stars/
[3]https://bimpactassessment.net
[4]https://www.ecogood.org/apply-ecg/common-good-matrix/
[5]https://www.globalreporting.org/standards/
[6]https://www.integratedreporting.org/resource/international-ir-framework/

or creating ESEA method models, in the line of earlier work such as [23] and [24]. Numerous evaluation protocols for modelling languages have been reported in literature. Most of these protocols are applied to diagrammatic modelling methods [25, 26]. Our evaluation protocol is highly influenced by previously existing literature, nonetheless it introduces activities tailored for evaluating an Xtext grammar.

4. ESEA method comparison

To produce V1 of our DSL and tool, we analysed 13 ESEA methods. Now we have analysed six additional ESEA methods to identify new requirements of the openESEA modelling language. The new methods are the CDP company programs[7], EFQM Model[8], S-CORE[9], SMETA[10], STARS, and WFTO Guarantee System[11]. The full list of analysed methods and their PDDs can be found in the technical report [27]. To extend the metamodel, we create a super-method that acts as a generic method, based on previously- and newly-modelled PDDs. This article explains the extension of the metamodel with additional metaclasses. Therefore, we focus on the results of the concept comparison rather than the activity comparison. A concept is only included in the super-method if it is present in more than one ESEA method. Table 1 shows a sample of the concept comparison. If a super-concept is present in a method, we mark it with an equal sign (=). If a super-concept is present in a method but it is referred to by another name we mark the cell with the concept name that is used in the method documentation. If a concept is not present in a method, the cell is left empty. If a concept is not explicitly mentioned in the method, but by applying our knowledge of ESEA methods and method engineering we can deduce that such a concept must be present, we put a hyphen (-) in the cell. Based on the method comparison we added nine more concepts to the metamodel. Concepts that were only present in one method were omitted. Moreover, we only focused on the accounting phase (i.e. assessing and reporting) even though many ESEA methods define an auditing protocol as well. We plan to include audit-related classes in a next version of the metamodel. To do this we will extend the method comparison. In Section 5.1 we report on the metaclasses that we added based on the concept comparison.

5. Modelling language for specifying ESEA methods

The openESEA modelling language consists of two artefacts: the metamodel and the DSL. The metamodel depicts the classes that are necessary to support the application of ESEA methods. A number of these metaclasses is used as the basis for engineering a textual grammar. Find a full explanation of all metaclasses, their attributes, and relationships in a technical report [28].

[7]https://www.cdp.net/en/companies
[8]https://efqm.org/efqm-model
[9]https://doi.org/10.4324/9781849770217
[10]https://www.sedex.com/our-services/smeta-audit/
[11]https://wfto.com/what-we-do#our-fair-trade-standard

Table 1
Sample of the concept comparison. An asterisk symbol (*) depicts whether a concept is new in V2

Super-concept	B Impact Assessment	Common Good Balance Sheet	SMETA
Survey *	B Impact Assessment	-	SA Questionnaire
Question *	=	=	=
Question response *	=	Evidence	-
Stakeholder *	=	=	-
Topic	Impact area	Theme	Pillar
Indicator	=	=	=

5.1. Metamodel of ESEA methods

Fig. 2 depicts metamodel V2. The metamodel serves as an ontology for managing complexity in ESEA methods. It contains classes with generic names. Classes in ESEA methods that represent the same concepts, but are referred to with different names, can be mapped against the classes in the metamodel. For example, in the B Impact Assessment method, the assessment is structured in "Impact areas". Our metamodel contains a class "Topic". Each impact area of the B Impact Assessment can be expressed by instantiating the topic class. To express the impact area called "Environment" method engineers can instantiate the topic class and provide values to the attribute as follows, `Id:impact_area_1`, `Name: Environment`, `Description:Evaluates a company's overall environmental management practices`

In comparison to metamodel V1, the metaclasses *Survey*, *Section*, *Text Fragment*, *Question*, *Answer option*, *Question response*, *Survey response*, *Campaign*, and *Stakeholder* are added. Moreover, we have added a generalisation metaclass called *Indicator*, which is specialised in the metaclasses *Direct indicator* and *Indirect indicator*. Lastly, we have removed the metaclasses *Report Item* and *Requirement*. With respect to certification requirements, we have decided to support their specification using indirect indicators. In turn, report item has been removed because the corresponding reporting capabilities of the interpreter tool were too limited and we plan to implement more versatile sustainability reporting features in the future. By adding new concepts to the metamodel we are able to express more aspects of ESEA methods. For instance, the metamodel now allows for specifying a survey that consists of questions. Examples of questions are "What was your company's total energy consumption in 2021?" and "What percentage of energy use is produced from low-impact renewable sources?". These questions can then be grouped in a section within the survey, or even in subsections within a section. For instance, the example questions above can be grouped in the subsection "Energy Consumption". This subsection can be part of the section "Air & Climate".

The metamodel differentiates between metaclasses that are instantiated when the method is engineered and specified (these have a grey background in the metamodel) and metaclasses that are instantiated when the method is applied and executed (these have a white background). According to Brinkkemper [29], method engineering is the engineering discipline to design, construct and adapt methods, techniques and tools for the development of information systems. In this article, engineering the methods entails designing and constructing ESEA methods. Examples of metaclasses that are instantiated during method engineering are *ESEA method* (which gives the method a name, a description, and the version), *Topic*s, and *Question.* The

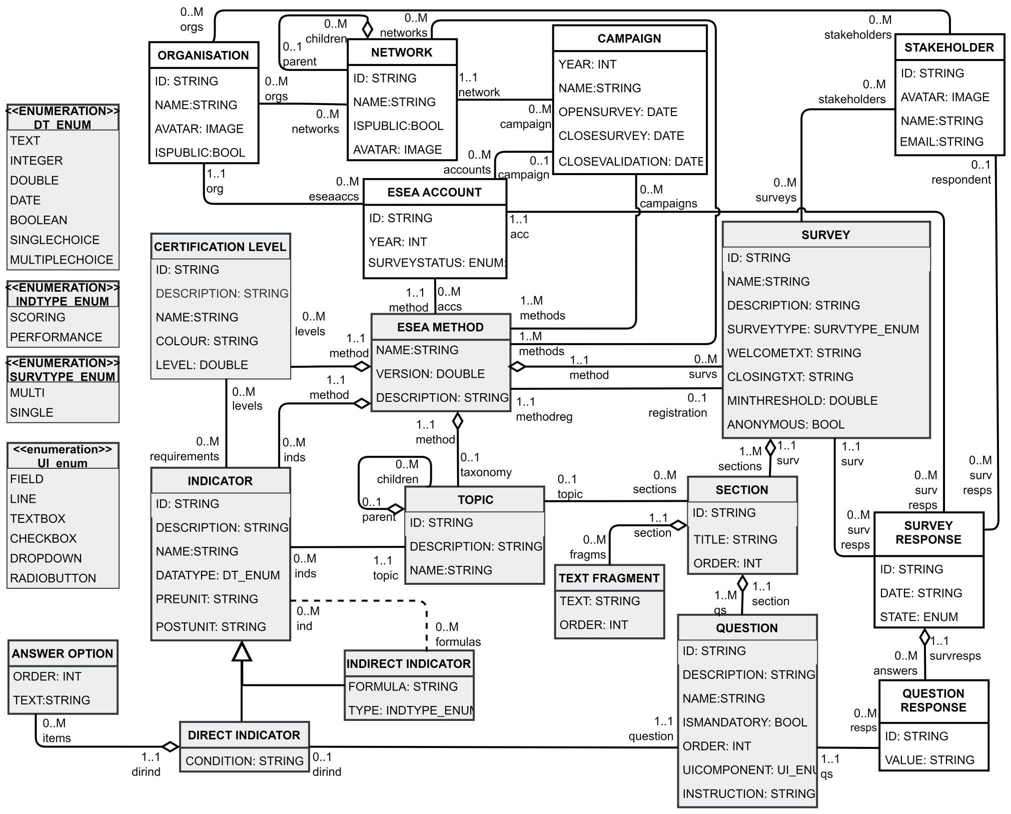

Figure 2: The openESEA metamodel contains the most relevant primitives needed to model ESEA methods

questions are asked, during execution time, to the people involved in the ESEA data collection (e.g. the ESE accountant or sustainability officer, staff members); however, they are specified during the method engineering. Examples of classes that are instantiated when applying or executing the method are *Organisation* (which represent entities that apply the method), and *Question response* (which stores the responses of questions for one specific application of an ESEA method, by a given organisation, in a given year). The grey metaclasses provide a proper abstract syntax for the grammar.

5.2. Textual grammar for creating ESEA models

We have implemented the DSL as a textual grammar that allows method engineers to create textual models of ESEA methods. The resulting textual models can then be parsed and interpreted by openESEA, and the tool reacts by offering the proper interfaces and features to support the modelled method. For every grey metaclass in the metamodel, we define a grammar primitive. While there is a multitude of ways of designing grammar rules that operationalise the metaclasses, we have opted for an approach that ensures human readability. For instance, every attribute is written on a new line and we try to choose commonly used, intuitive names for concepts (e.g. for UI components we used common names such as radio button, check box,

Table 2
On the left, Xtext grammar excerpt containing the rule that structures the ESEA method model. On the right, a model fragment according to the grammar rules on the left. The model fragments for the lists are omitted since they reference other rules of which the models contain several lines each.

Grammar rule	Model fragment
`ESEA method:`	
`"Name:" STRING`	`Name: "B Impact Assessment"`
`"Version:" DOUBLE`	`Version: 6.0`
`"isPublic:" BOOLEAN`	`isPublic: True`
`"Description:" STRING`	`Description:"First step towards B Corp Certification"`
`listTopics+=ListTopics`	`...`
`listIndicators+=ListIndicators`	
`listSurveys+=ListSurveys`	
`listCertificationLevels+=ListCertificationLevels`	
`;`	

text field, etc.). An example of a grammar primitive can be found in the left column of Table 2. Here the metaclass *ESEA method* is supported by an Xtext grammar rule. The rule states that the method engineer should specify the name of the method, version, description, and whether the method is publicly available for any organisation to access and apply it. The lists refer to lists of topics, indicators, surveys and certification levels. Each list contains a set of items. For instance, the topics list contains a set of ESE topics as specified in the ESEA method. In the case of the B Impact Assessment the topic names are “Governance”, “Community”, “Workers”, “Environment”, and “Customers”. The topics are then further specified in indicators. An example of a model fragment according to this grammar rule is depicted in the right column in Table 2. For the sake of brevity, we omitted the model fragments that correspond to the lists. For the full Xtext grammar, see the technical report[28] or find it on Github [12].

6. Validation of the ESEA grammar

6.1. User test design

To assess the grammar’s performance, we run a user test, supported by an e-assessment tool. Fig. 3 shows the test procedure and variables. We use the reporting guidelines from [30]. The **object of study** is the grammar. We leave the Xtext editor out of the scope, since it might interfere with the results. The **main objective**, assessing the grammar, is refined into two sub-objectives: (i) determine to what extent users are able to successfully create ESEA method models, using the DSL, (ii) discover potential improvements of the grammar by performing qualitative analyses on the user test results. The **test participants** are 75 Information Science bachelor students from Utrecht University, with little to no professional experience, little programming knowledge, and no knowledge of model-driven architectures, textual grammars, and ESEA prior to the user test. The expected future users of the grammar are ESEA method engineers, having similar experience with ICT, but greater knowledge of ESEA.

[12]https://github.com/sergioespana/openESEA

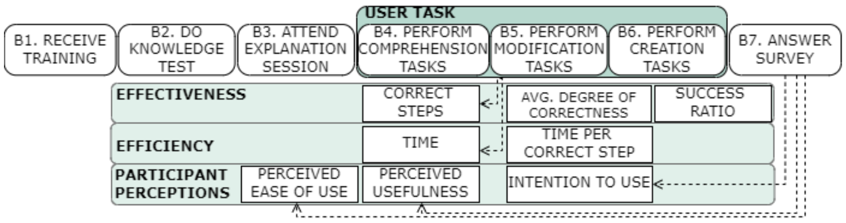

Figure 3: Overview of the user test. Participants first receive training. While carrying out the tasks, users spend some time taking a set of steps. We then elicit their perceptions.

The **test structure** is as follows. There are three types of tasks: comprehension, modification, and creation. For each task, we have formulated questions, each consisting of a number of steps. The comprehension questions are the easiest, the modification questions ask the participants to make a change in a given model, and the creation questions are the most challenging; they require the participants to create a model from scratch, based on a textual description of an ESEA method or a screenshot of a real ESEA tool. For comprehension questions, a step typically refers to answering a multiple-choice question. For modification questions, a step refers to making an alteration in a model or filling in a text field. For creation questions, a step refers to writing a line of a model fragment. For instance, the model in Table 2 consists of four lines, thus a question that asks to create such a model has four steps.

The **variables** we measure based on the MEM [13] are the effectiveness of using the DSL grammar, the efficiency in the proposed tasks, and the participant perceptions. These variables refer to the environment and structure system dimensions according to the hierarchy of criteria for information system artifact evaluation [31]. For each of these constructs, we define response variables [32]. **Effectiveness** refers to how well the DSL achieves its objectives. While assessing the test responses, we produce the values of the *correct steps* variable. With these values, we can calculate the following variables: *average degree of correctness* (see formula 1) measures to what extent the participant correctly conducted the steps of the modelling tasks in the user task, *success* indicates that the participant did not make any mistakes and thus answered the entire question correctly, and *success ratio* (formula 2) reflects the normalised percentage of successful responses.

$$average\ degree\ of\ correctness = \frac{\sum_{q=1}^{|tquestions|} \frac{\sum_{qr=1}^{|responses|} \frac{correct\ steps_{qr}}{steps_{qr}}}{|responses|}}{|tquestions|} \tag{1}$$

$$success\ ratio = \frac{\sum_{q=1}^{|tquestions|} \frac{\sum_{qr=1}^{|responses|} success}{|responses|}}{|tquestions|} \tag{2}$$

Efficiency refers to the effort required to apply the DSL. For each question, the e-assessment tool automatically measures the time that the participant spent on it. As a better variable for efficiency, we define *time per correct step* (formula 3).

$$time\ per\ correct\ step = \frac{\sum_{q=1}^{|tquestions|} \frac{\sum_{qr=1}^{|responses|} \frac{time_{qr}}{correct\ steps_{qr}}}{|responses|}}{|tquestions|} \tag{3}$$

The formulas are aggregating the results per task (comprehension, modification or creation), where $|tquestions|$ represents the total number of questions per task. When aggregating the average degree of correctness and success ratio per grammar primitive, $|tquestions|$ should be changed to $|pquestions|$ which represents the total number of questions related to each grammar primitive (per task). Similarly, *responses* represents the set of responses. To assess **participant perceptions**, the MEM offers an adaptable questionnaire that allows measuring the *perceived usefulness*, the *perceived ease of use*, and the *intention to use* the DSL in the future, if confronted with similar tasks during their profession.

In accordance with the **test procedure** shown in Fig. 3, the participants receive a ninety-minute training (B1) where we introduce them to ESEA methods. We also train them in the DSL grammar. After the training, the participants perform a knowledge test (B2). The knowledge test also consists of three types of questions (comprehension, modification, and creation). The purpose of the knowledge test is to have the participants apply their newly acquired knowledge and receive feedback on their performance. This resembles the training that the future users of the grammar (i.e. ESEA method engineers) will receive. After the participants have finished the knowledge test, they attend an explanation session (B3), where we discuss the answers to the knowledge test. Hereafter, the participants start the user task; i.e. a test where we actually measure their effectiveness and efficiency. They answer comprehension (B4), modification (B5) and creation questions (B6), in that order. The questions are similar to the ones in the knowledge test, but the overall test is longer and more challenging. After the test, the participants are asked to fill in the MEM questionnaire (B7).

6.2. User test results

Table 3 shows the average degree of correctness and the success ratio for each grammar primitive per task, as well as the time spent per correct step, the average degree of correctness, and the success ratio aggregated per task. The average degree of correctness per task is quite high, ranging from 86% to 89%. The success ratios range from 72% for the creation task to 83% for the modification task. Overall, a positive sign, indicating that many participants were able to execute questions flawlessly. When evaluating the participant's answers we have chosen to be very strict since, strictly speaking, every deviation from the grammar rules is a syntactic error. In practice, most syntactic mistakes will be prevented by the usage of the editor, since the editor ensures that the models comply with the syntax.

To put efficiency results in the context of industrial practice, we have estimated the size of a real ESEA method and the total time it would take to author its model using the grammar. We have taken the basic variant of the XES Social Balance as a reference, since we have full access to its internal documentation. That method has five topics, 203 indicators, one survey, five survey sections, 86 questions, and five text fragments. The language primitive *Topic* requires three lines, so modelling all five topics of the XES Social Balance implies writing 15 lines. An *Indirect indicator* has eight lines, *Direct indicator* has seven lines, *Answer option* has three lines,

Table 3
The average degree of correctness and success ratio per primitive per task. Additionally, the aggregated values per task and time per correct step per task.

	Avg degree of correctness	Success ratio	Time per correct step (s)
Comprehension	0.89	0.81	57.18
Section	0.86	0.63	
Text Fragment	0.89	0.89	
Indicator	0.91	0.79	
Certification level	0.97	0.97	
ESEA method	0.80	0.80	
Modification	0.86	0.83	55.63
Indicator	0.75	0.75	
Certification level	1.00	1.00	
Answer option	1.00	1.00	
Question	0.82	0.80	
Survey	1.00	0.99	
Creation	0.87	0.72	50.00
Survey	0.97	0.85	
Section	1.00	0.99	
Text Fragment	0.91	0.87	
ESEA method	0.98	0.93	
Topic	0.99	0.98	
Indicator	0.78	0.15	
Answer option	0.39	0.39	
Question	0.86	0.37	

Survey has eight lines, *Survey section* has four lines, *Question* has eight lines, and *Text fragment* has three lines. As a result, the total size of the model would be 2,916 lines. Given that our participants spent 50 seconds per correct line, creating a fully correct method model without the editor would take them $50 * 2,916 = 145,800$ seconds (40 hours and 30 minutes). Most probably, this is an overestimation. We suspect that users of the grammar will become more efficient while producing the model because there are many repetitive actions. For instance, we expect that the ESEA method engineers will spend more time on the first few indicators and become quicker as they progress. They can also copy, paste, and tweak method fragments. Furthermore, in a real-life setting, grammar users will use the editor, which should further improve their efficiency. On the other hand, the real source of complexity in ESEA method engineering is the participatory design of the method, especially in the case of bottom-up, democratic design processes, as is the case of the XES Social Balance. But this falls out of the scope of the DSL.

The results per grammar primitive help us pinpoint where we can improve the grammar and the training. The grammar primitive *Answer option* in the creation task has the lowest values for the effectiveness variables. The reason for this is that most participants forgot to define the answer options altogether. Perhaps users find it counter-intuitive to define the answer options when modelling a direct indicator (see metamodel, Figure 2). A more intuitive approach could be to define the answer options when modelling its corresponding question. However, a more probable reason for participants forgetting to define the answer options is that users have to select the correct data type for the answer option rule to be triggered. Upon further inspection, we found that in many cases the selected data type was incorrect, making the forgotten answer options an unpreventable follow-up error. The autocompletion feature of

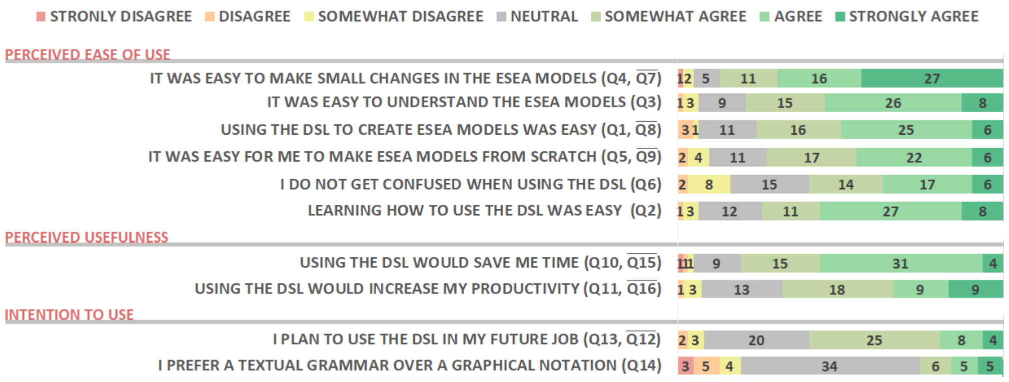

Figure 4: The MEM questionnaire results (n=62). After each statement, the question IDs are listed between parentheses (*Qx*). Note that some statements had a negated counter question (depicted with $\overline{Qx}$) to minimise and correct for acquiescence.

the editor will show users exactly which grammar rules are triggered. This will, for instance, prevent the users from forgetting to define answer options. The *Indicator* primitive scores fairly well in the comprehension and modification task. In the creation task, on the other hand, *Indicator* has the lowest success ratio. Most syntactic mistakes in the creation questions related to *Indicator* were non-critical (e.g. capitalisation mistakes). Most semantic mistakes were made in the data type. This compromises the utility of the indicator for measuring the intended organisational sustainability performance. In the indicator creation task 65% of the subjects chose the wrong data type. We suspect this is caused by a lack of experience with ESEA and insufficient knowledge about ESEA methods. This can probably be solved by providing a longer, more detailed training session. The success ratio of the *Question* primitive in the creation task is rather low, but most mistakes are non-critical. However, one frequently appearing serious mistake is made in the *order* attribute of *Question*. The order in which questions should be displayed in a survey is indicated with a numeric value in the attribute *order*. The question with the lowest order value is displayed first, followed by the question with the consecutive numeric value, and so on. Overlapping order numbers are not allowed, since multiple questions cannot be in the same place in the survey. Nonetheless, users frequently gave questions the same order number. To tackle this problem, we should add a constraint to the grammar by extending the validator [33]. The grammar primitives that have proven to cause confusion will be reassessed and possibly updated in next versions of the modelling language.

The results of the perceptions and intentions questionnaire yield Fig. 4. It shows whether participants found the grammar easy to use and useful. Additionally, it gives some indication of their intention to use the grammar in combination with a model-driven interpreter, if they become a sustainability officer after their studies. The majority of the participants have indicated that they found it easy to make small changes in the models, understand the models, and create models. Just 24% of the participants found that creating ESEA models required a lot of mental effort. For perceived usefulness, 67% of the participants felt like using the grammar would save them time when engineering ESEA methods and 48% said that using the grammar would increase their productivity. The majority of the participants have indicated that they are willing to use openESEA in combination with the grammar if they ever become an ESE accountant.

7. Discussion

The modelling language allows us to specify any ESEA methods and combinations of ESEA methods. The benefit of this is that organisations can create a model combining all the methods that they wish to apply (e.g. B Impact Assessment, Common Good Balance Sheet, and GRI Standards). By uploading the textual model (which contains three methods) in openESEA, the tool parses and interprets the model. It displays all surveys, topics, indicators and other necessary elements to execute the three methods specified in the textual model. This way organisations will no longer have to use multiple tools to support their ESEA.

To increase the expressiveness of the DSL we added new concepts to the metamodel. This, of course, increases the complexity of the DSL. However, by means of a user test, we found that users can manage this complexity, since they can create models successfully. The user test allowed us to identify four major points of improvement. At least one of these points can be resolved by introducing the editor. Two points can be addressed with additional training and one point requires an extension of the Xtext validator. We consider the user test overall a success and deem that, overall, users can effectively create ESEA models. The success rate and degree of correctness are high and the observed mistakes are mostly non-critical.

The only model-driven approach for ESEA we found is our own earlier work [5]. The modelling language presented herein builds upon the artefacts from that article, therewith all extensions of these artefacts are new contributions to the model-driven ESEA approach. For testing an Xtext DSL we only found one article that also performed a user test. Jonathan et al., developed a DSL and syntax checker in Xtext for producing mapping configuration files for ASML's Twinscan machine. To test the artefacts, a component test, code review, and user test were performed [34]. Similar to our user test, test subjects were asked to create configuration files. The number of scientific works that describe how to test Xtext DSLs is limited. The test procedure in this article can contribute to an approach for testing Xtext DSLs.

A limitation is that the group of students is not a representative group to measure the intention to use, since they do not (yet) work in the domain of ESEA. Consequently, they might not be able to imagine whether they would use such a tool. Anyhow, in earlier (and ongoing) expert assessment interviews, ESEA experts show appreciation and interest in our approach [5].

Future work includes testing how the Xtext model editor influences the grammar usage and whether the proposed improvements result in a more successful application of the grammar. Also, we plan an experiment to evaluate the learning curve of grammar users. We opted not to do this for the current version of the grammar, since we still have a planned roadmap with substantial updates. Additionally, we could extend the Xtext validator to implement constraints and provide domain-specific errors [33]. In terms of new versions of the modelling language for specifying ESEA methods, we plan to extend the language so that ESE accounts can be audited. Auditing and assurance of sustainability reports are critical for building trust in their content. On top of that, we envision that openESEA will, in the future, support other impact measurement families, such as social impact assessment and life-cycle assessment. Finally, we have so far opted for engineering a textual grammar to support the DSL. We actually envision a combination of textual and graphical modelling of ESEA methods, since a graphical specification of the process dimension of methods may be more intuitive and we already have experience in modelling ESEA methods with PDDs.

8. Conclusion

We consider that business informatics will play a role in making organisations more sustainable and managing the complexity that these practices entail. In this paper, we contribute an updated modelling language for ethical, social and environmental accounting methods. The modelling language is interpreted by an open-source, model-driven tool for producing ESE assessments and reports. The modelling language consists of a metamodel and a textual grammar. To design the grammar we have used improvement points from [5], and analysed additional ESEA methods. We have tested our approach with a user test. The results are promising and lay the basis for further development. We intend that the textual grammar can be used to express any ESEA method, although this remains to be proven. The ESEA models can then be uploaded to openESEA, which will be capable of supporting the method applications. Therewith, we expect to simplify the ESEA ICT tool support and hopefully lower the barrier for performing ESEA. With this change in the tool support landscape, we encourage organisations to start measuring, reporting and monitoring ESE performance and impacts, so they discover how they can start building sustainable communities and formulate their sustainability strategy.

References

[1] V. Smith, J. Lau, J. Dumay, Shareholder use of CSR reports: an accountability perspective, Meditari Accountancy Research (2021).

[2] S. España, V. Ramautar, Q. T. Le, Assessing the ethical, social and environmental performance of conferences, in: RCIS proc., Springer, 2022, pp. 752–760.

[3] R. Gray, C. A. Adams, D. Owen, Accountability, social responsibility and sustainability, Pearson, 2014.

[4] C. A. Adams, C. Larrinaga-González, Engaging with organisations in pursuit of improved sustainability accounting and performance, Account. Audit. Account. J. (2007).

[5] S. España, N. Bik, S. Overbeek, Model-driven engineering support for social and environmental accounting, in: RCIS, IEEE, 2019, pp. 1–12.

[6] H. Behrens, M. Clay, S. Efftinge, M. Eysholdt, P. Friese, J. Köhlein, K. Wannheden, S. Zarnekow, contributors, Xtext user guide, Eclipse Fundation (2010).

[7] R. J. Wieringa, Design science methodology for information systems and software engineering, Springer, 2014.

[8] M. Mernik, J. Heering, A. M. Sloane, When and how to develop domain-specific languages, ACM Comput Surv 37 (2005) 316–344.

[9] I. van de Weerd, S. Brinkkemper, Meta-modeling for situational analysis and design methods, in: Handbook of research on modern systems analysis and design technologies and applications, IGI Global, 2009, pp. 35–54.

[10] I. van de Weerd, S. de Weerd, S. Brinkkemper, Developing a reference method for game

production by method comparison, in: Working Conf. on Method Engineering, Springer, 2007, pp. 313–327.

[11] OMG, Unified Modeling Language (OMG UML), Version 2.5.1, 2017.

[12] ISO/IEC, International vocabulary of metrology – Basic and general concepts and associated terms (VIM), Standard, ISO/IEC Guide 99:2007, 2007.

[13] D. L. Moody, The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods, in: ECIS Proc., 2003.

[14] R. Gray, J. Bebbington, Accounting for the environment, Sage (1993) 13.

[15] G. Lamberton, Sustainability accounting—a brief history and conceptual framework, in: Accounting forum, volume 29, Elsevier, 2005, pp. 7–26.

[16] S. Sisaye, The influence of non-governmental organizations (ngos) on the development of voluntary sustainability accounting reporting rules, JBSED (2021).

[17] S. S. Gao, J. J. Zhang, Stakeholder engagement, social auditing and corporate sustainability, BPMJ (2006).

[18] P. Castka, M. A. Balzarova, A critical look on quality through CSR lenses: Key challenges stemming from the development of ISO 26000, IJQRM (2007).

[19] V. Paelman, P. Van Cauwenberge, H. Vander Bauwhede, Effect of B Corp certification on short-term growth: European evidence, Sustainability 12 (2020) 8459.

[20] A. Singh, M. Chakraborty, Does CSR disclosure influence financial performance of firms? Evidence from an emerging economy, SAMPJ (2021).

[21] R. Küchler, C. Herzig, Connectivity is key: holistic sustainability assessment and reporting from the perspective of food manufacturers, British Food Journal (2021).

[22] G. Figueiredo, A. Duchardt, M. M. Hedblom, G. Guizzardi, Breaking into pieces: An ontological approach to conceptual model complexity management, in: RCIS, 2018, pp. 1–10.

[23] D. L. Moody, Complexity effects on end user understanding of data models: An experimental comparison of large data model representation methods, ECIS (2002) 10.

[24] M. E. Manso, M. Genero, M. Piattini, No-redundant metrics for UML class diagram structural complexity, in: CaiSE, Springer, 2003, pp. 127–142.

[25] R. Petrusel, J. Mendling, Eye-tracking the factors of process model comprehension tasks, in: CaiSE, Springer, 2013, pp. 224–239.

[26] R. A. Buchmann, D. Karagiannis, Modelling mobile app requirements for semantic traceability, Requir. Eng. 22 (2017) 41–75.

[27] V. Ramautar, S. España, Domain Analysis of Ethical, Social and Environmental Accounting Methods, Technical Report, Utrecht University, 2022.

[28] V. Ramautar, S. España, The openESEA Modelling Language for Ethical, Social and Environmental, Technical Report, Utrecht University, 2022.

[29] S. Brinkkemper, Method engineering: engineering of information systems development methods and tools, Inf Softw Technol 38 (1996) 275–280.

[30] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Science, 2012.

[31] N. Prat, I. Comyn-Wattiau, J. Akoka, Artifact evaluation in information systems design-science research–a holistic view, in: PACIS proc. 23, 2014.

[32] N. Juristo, A. M. Moreno, Basics of software engineering experimentation, Springer, 2013.

[33] L. Bettini, Implementing domain-specific languages with Xtext and Xtend, Packt, 2016.
[34] B. Jonathan, R. Avetyan, S. Abeln, Create Domain-Specific Language and Syntax Checker Using Xtext, Int. j. eng. res. ind. appl. 4 (2020) 26–32.