Improving the Usability of a MAS DSML

Tomás Miranda, Moharram Challenger, Baris Tekin Tezel, Omer Faruk Alaca, Vasco Amaral, Miguel Goulão, and Geylani Kardas

NOVA LINCS, DI, FCT, Universidade NOVA de Lisboa
International Computer Institute, Ege University, Izmir-Turkey
tr.miranda@campus.fct.unl.pt, baris.tezel@deu.edu.tr, omerfarukalaca@gmail.com, {moharram.challenger, geylani.kardas}@ege.edu.tr, {vma, mgoul}@fct.unl.pt

Abstract. Context: A significant effort has been devoted to the design and implementation of various domain-specific modeling languages (DSMLs) for the software agents domain.
Problem: Language usability is often tackled in an ad-hoc way, with the collection of anecdotal evidence supporting the process. However, usability plays an important role in the productivity, learn-ability and, ultimately, in the adoption of a MAS DSML by agent developers.
Method: In this paper, we apply the principles of *The "Physics" of Notations* (PoN) to improve the visual notation of a MAS DSML, called SEA_ML and evaluate the result in terms of usability.
Results: The evolved version of the language, SEA_ML++, was perceived as significantly improved in terms of icons comprehensibility, adequacy and usability, as a direct result of employing the principles of PoN. However, users were not significantly more efficient and effective with SEA_ML++, suggesting these two properties were not chiefly constrained by the identified shortcomings of the SEA_ML concrete syntax.

Keywords: Usability · Multi-Agent Systems · Domain Specific Modeling Language · Physics of Notations · SEA_ML

1 Introduction

Software agents with the capability of both being autonomous and performing reactive/proactive behaviors, interact with each other in a Multi-agent system (MAS) to solve problems in a competitive or collaborative manner within an environment. To eliminate the complexity and difficulty encountered during MAS development, the researchers in agent-oriented software engineering (AOSE) field have significant efforts (e.g. DSML4MAS [17], FAML [3], SEA_ML [6], MAS-ML [9], JADEL [2]) on design and implementation of various domain-specific modeling languages (DSMLs). Those DSMLs are specific to the agent domain and provide appropriate integrated development environments (IDEs) in which both modelling and code generation for system-to-be-developed can be performed properly [22].

To be effective, the proposed agent DSMLs need to meet the various stakeholder concerns and the related quality criteria for the corresponding MASs. Unfortunately, very often the evaluation of the DSML, especially covering the language components and the use of the DSML during design and implementation of agent-based systems, is completely missing or has been carried out with an idiosyncratic approach [7]. Specifically, the usability, which plays an important role on adoption of a MAS DSML by agent developers, needs to be taken into consideration preferably during language design and improved to better align the DSML with developer expectations. Hence, in this study, we focus on the usability of DSMLs for MAS and propose an approach for promoting the usability of such languages by applying the principles of The "Physics" of Notations (PoN) [26]. For this purpose, the visual notation of a MAS DSML, called SEA_ML [6], is evaluated and its usability is improved by employing each principle of PoN. Hence, it is possible to enrich SEA_ML's visual notation and its correlation to the linked semantic constructs. A comparative assessment of the improved language is also performed with two different experiments using end-users that are defined by the domain experts. SEA_ML is an open source language and it is easy to achieve both abstract and concrete syntax specifications. Moreover, reflecting the changes according to the conducted PoN experiments and generating the new version of the language become much more easier since some of the developers of the language also participate in this study. These are the main reasons of selecting SEA_ML as the application language in our work.

The rest of the paper is organized as follows: Section 2 and Section 3 discuss SEA_ML and the principles of PoN respectively. The analysis of SEA_ML and improving its visual notation by using PoN principles are given in Section 4. Comparative evaluation of the new language is discussed in Section 5. Related work is given in Section 6 and Section 7 concludes the paper.

2 SEA_ML

SEA_ML [6] is a MAS modeling language which enables the developers to model agent systems in a platform independent level and then automatically generate codes and related documents required for the execution of the modeled MAS on target MAS implementation platforms. To support MAS experts when programming their systems, and to be able to fine-tune them visually, SEA_ML covers all aspects of an agent system from the internal view of a single agent to the complex MAS organization. To this end, it includes eight viewpoints, namely, Agent Internal, MAS, Interaction, Role, Environment, Plan, Ontology, and Agent-SWS. In addition to these capabilities, SEA_ML also supports the model-driven design and implementation of autonomous agents who can evaluate semantic data and collaborate with semantically-defined entities of the Semantic Web [30], like Semantic Web Services (SWS). That feature exactly differentiates SEA_ML and makes unique regarding any other MAS DSML currently available. Within this context, it includes new viewpoints which specifically pave the way for the development of software agents working on the Semantic Web environment. Modeling agents, agent knowledge-bases, platform ontologies, SWS and interactions between agents and SWS are all possible in SEA_ML.

SEA_ML can be used for both modeling MASs and generation of code from the defined models. SEA_ML instances are given as inputs to a series of M2M and M2T transformations to achieve executable artifacts of the system-to-be-built for JADEX [28] agent platform and semantic web service description documents conforming to Web Ontology Language for Services (OWL-S) ontology [25].

To demonstrate the modeling and implementation environment provided by SEA_ML, let us consider the development of a MAS for stock exchange software in which Investor (Buyer and/or Seller), Broker and Stock Trade Manager agents take role in a computerized stock trading system. All of the user agents including investors and brokers cooperate with stock trade manager agent to access the stock market. Also, the user agents interact with each other, for instance, investor A and investor B can cooperate with a broker in order to exchange the stock for which the broker is an expert. Figure 1 is a screenshot taken from SEA_ML modeling environment which shows the modeling of such a stock exchange MAS which is composed of six semantic web agent instances, one trade manager, two brokers, and three investors. The given model only considers the overview of the system from SEA_ML MAS viewpoint. However, it is also possible to model all specifications and components of the system considering the other SEA_ML viewpoints again inside the same IDE. Interested readers may refer to [6] for an extensive discussion on SEA_ML and [21] for complete design and implementation of this agent-based stock exchange system with SEA_ML.

3 Physics of Notations

The Physics of Notations (PoN) [26] is a set of principles, used to evaluate, compare and ultimately enhance the communication properties of a given language when designing its concrete syntax, see Table 1. Therefore, it addresses the often neglected perceptual aspects rather than their semantics. The ability of visual notations to communicate with stakeholders is called cognitive effectiveness, defined (and measured) as the speed, ease, and accuracy with which a representation can be processed by the human mind.

4 Applying Physics of Notations to SEA_ML

4.1 Applying the Principles

We proposed some symbol improvements for the SEA_ML visual notation by following The "Physics" of Notation principles. These improvements are employed in the development of the new version of SEA_ML called SEA_ML++. All viewpoints of SEA_ML are considered in this improvement.

Of the 43 symbols (44 including the symbol for arrows that relate each entity), 32 symbols were modified to follow the principles presented above.

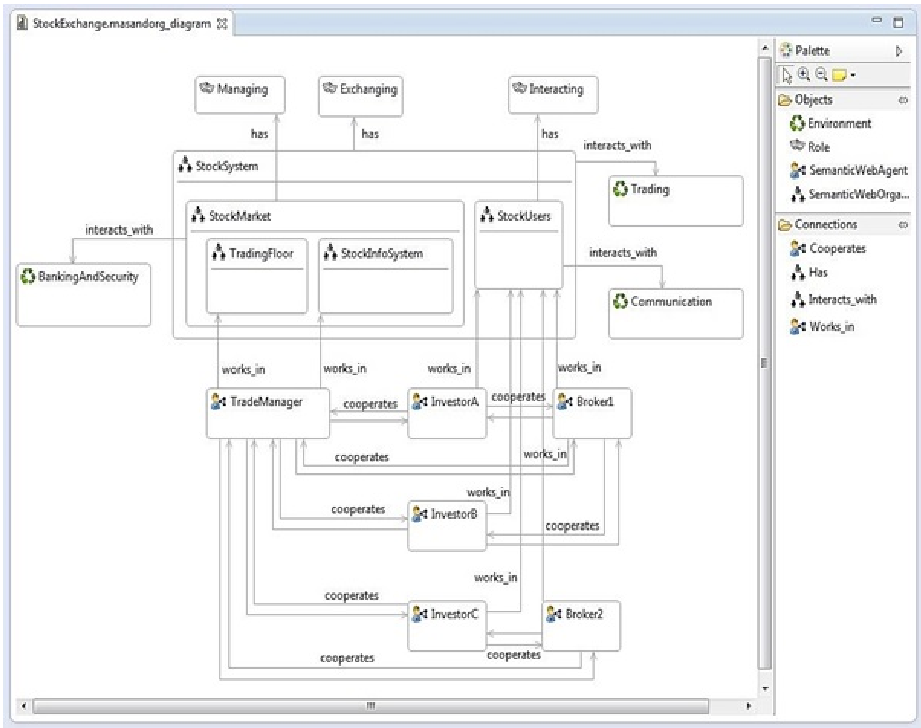


Fig. 1: MAS and Organization diagram for Stack Exchange System in SEA_ML

Table 2 synthesizes the information detailed on each sub-section of PoN shown in Table 1. Plus (+) refers that SEA_ML is currently according the presented principle, while a minus (-) refers that SEA_ML has room for improvement under that principle.

Some SEA_ML visual notations were not modified. The main reason is that these notations reflect correctly its semantic constructs and are according to *The "Physics" of Notations* principles. However, most of the notations (32 of them) are modified based on PoN principles, see Figure 2 for the current (SEA_ML) and new (SEA_ML) notations. With respect to the proposed SEA_ML++, the explanation for each new notation is defined below:

1. **Goal** - The new notation adds color to the target, making it more appropriate to be selected when using viewpoints that use this semantic construct;
2. **Capability** - The current visual notation may induce users wrong. The new notation reflects that users have a set of capabilities in order to solve their problems;
3. **Fact** - The current notation is similar to other notations present in SEA_ML. The new notation (check mark) reflects something that is correct and concrete;
4. **Plan** - The notation addresses a plan to reach a goal from X to Y;
5. **Semantic Service Register Plan (SSRP)** - The current notation has four similar symbols, being distinguished through different letters. The new notation adds the SWS notation and a person registering to a customer's list;

Table 1: PoN principles

Principle	Comment
Semiotic Clarity	There should be a 1:1 correspondence between semantic constructs and graphical symbols
Perceptual Discriminability	Different symbols should be clearly distinguishable from each other
Semantic transparency	The appearance of visual representations should suggest their meaning
Complexity management	Explicit mechanisms for dealing with complexity should be included
Cognitive integration	There should be explicit mechanisms to support the integration of information from different diagrams
Visual expressiveness	The full range of capacities and visual variables should be used
Dual coding	Text should be used to complement graphics
Graphic economy	The number of symbols presented in the notation may affect the handling of the tool
Cognitive fit	Different dialects should be used for different tasks and audiences.

6. **Semantic Service Finder Plan (SSFP)** - The current notation has four similar symbols, being distinguished through different letters. The new notation adds the "Semantic Web Services" notation and a magnifying glass;
7. **Semantic Service Agreement Plan (SSAP)** - The current notation has four similar symbols, being distinguished through different letters. The new notation adds the "Semantic Web Services" notation and a handshake between two people;
8. **Semantic Service Executor Plan (SSEP)** - The current notation has four similar symbols, being distinguished through different letters. The new notation adds the "Semantic Web Services" notation and a "Play" icon;
9. **Send** - It is not clear what the current notation is addressing. The new notation states clearly that the message is going to be sent elsewhere;

Table 2: Feedback of SEA_ML visual notation according to each principle of PoN

Principle	Room for Improvement
Semiotic Clarity	+
Perceptual Discriminability	-
Semantic Transparency	-
Complexity Management	+/-
Cognitive Integration	+
Visual Expressiveness	-
Dual Coding	-
Graphic Economy	+
Cognitive Fit	-
+ OK — - can be improved	

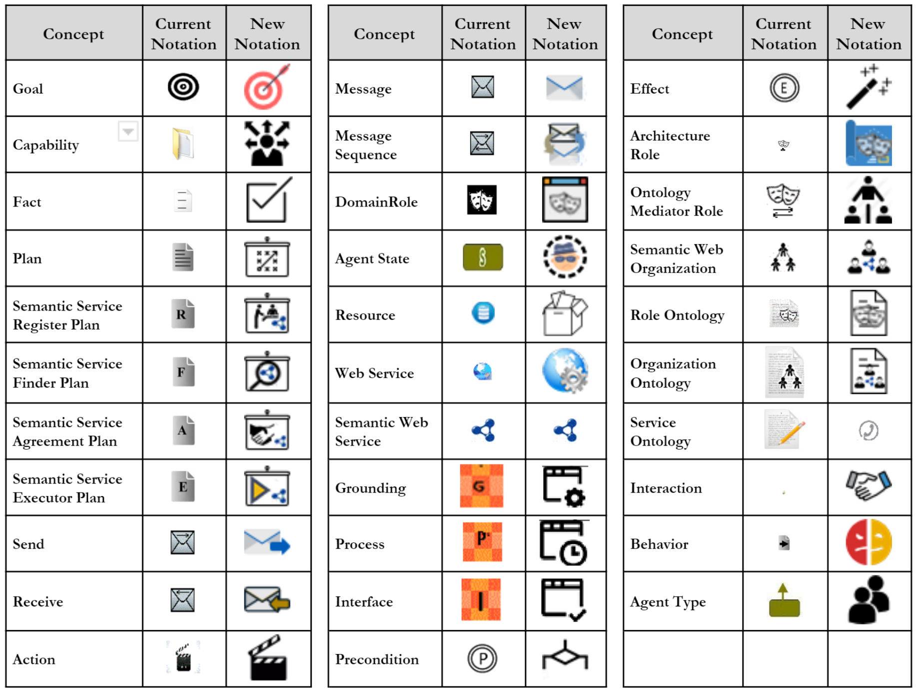


Fig. 2: SEA_ML vs SEA_ML++ notations

10. **Receive** - It is not clear what the current notation is addressing. The new notation states clearly that the message is going to be received;
11. **Action** - Removed the round border. The clapperboard is enough to understand the semantic construct;
12. **Message** - The new notation attempts to be similar to the new notations adopted in "Message Sequence", "Send" and "Receive";
13. **Message Sequence** - Similar to the notations presented in "Send" and "Receive", the new notation hints a sequence of message being transmitted by those parties;
14. **ODMOWLClass** - The new notation is similar to the previous "Plan" symbol. It tries to remove two similar from the visual notation (as the "Plan" symbol is totally different from the original one);
15. **DomainRole** - The current visual notation does not have any relation with a domain. The metaphor tried on the new notation aims at reflecting the web domains, inserting its roles on a web browser window;
16. **Agent State** - The current visual notation does not have any relation with an Agent State. The new notation attempts to add a "Secret Agent" to a typical rounded "State Icon" that appears on some loading screens;
17. **Resource** - The new notation reflects a box full of resources, which reflects more what the semantic construct is;
18. **Web Service** - The new notation adds a gear to an icon that relates to the web;
19. **Grounding** - Proposed by the developers having experience on MAS and SWS;
20. **Process** - Proposed by the developers having experience on MAS and SWS;

21. **Interface** - Proposed by the developers having experience on MAS and SWS;
22. **Precondition** - Proposed by the developers having experience on MAS and SWS;
23. **Effect** - The current visual notation does not have any direct relation with Effect. The new notation tries to adapt the "Magic" metaphor for an effect cause;
24. **Architecture Role** - The current visual notation does not have any direct relation with an "ArchitectureRole". The new icon adds the "Role" symbol to a common architecture plan;
25. **Ontology Mediator Role** - Proposed by the developers having agent programming experience;
26. **Semantic Web Organization (SWO)** - The current visual notation does not have any direct relation with a SWO. The new symbol adds that relation;
27. **Role Ontology** - The new visual notation adapts to the new ODMOWLClass proposed above;
28. **Organization Ontology** - The new visual notation adapts to the new ODMOWLClass and "Semantic Web Services" proposed above;
29. **Service Ontology** - The new notation adapts to the new ODMOWLClass proposed above;
30. **Interaction** - Although it is perceptible what the current visual notation proposes, there is room for improvement by adding a clearer symbol;
31. **Behavior** - The current visual notation does not have any relation with the "Behavior" semantic construct. The new symbol tries to apply a metaphor related to the human behavior;
32. **Agent Type** - Proposed by the developers having agent programming experience.

5 Evaluation

5.1 Experiment planning

Goals Broadly, we aim to compare the impact of using the evolved version of the MAS DSML (*SEA_ML++* when contrasted with the previous version (*SEA_ML), focusing, one at a time, in different quality criteria for the language assessment.* We present our evaluation goals following the GQM research goals template [1], which is shared among all our goals, with the exception of the term ***concrete quality criterium***, which varies from one goal to the next.

In general, our goal is to **analyse** the effect of evolving from *SEA_ML* to *SEA_ML++*, **for the purpose of** evaluation, **with respect to** the ***semantics transparency*** of the symbols used in the concrete syntax, **from the viewpoint of** researchers, **in the context of** an experiment conducted with participants with limited or no experience with MAS at Universidade Nova de Lisboa (UNL) in Portugal and Ege University in Turkey.

More specifically, our first goal is concerned about the *comprehensibility* of the symbols used on the concrete syntax, leading to the following formulation: Our first goal (G1) is to **analyse** the effect of evolving from *SEA_ML* to *SEA_ML++*, **for the purpose of** evaluation, **with respect to** the *comprehensibility* of the symbols used in the concrete syntax, **from the viewpoint of** researchers, **in the context of** an experiment conducted with participants with limited or no experience with MAS at UNL and Ege University. Our second goal

(G2) is concerned about the ***perceived usability*** of the concrete syntax. Our third goal (G3) is concerned about the ***effectiveness*** of the concrete syntax. Finally, our fourth goal (G4) is concerned about the ***efficiency***.

Tasks To achieve *(G1)*, (1) each participant read and signed a consent letter regarding the data collected in the experiment. This letter was only used for the purpose of this study. All participants remained anonymous. Then (2) each participant selected the symbol (s)he found more suitable for each of the 33 *SEA_ML++* concepts identified in the PoN assessment reported in Section 4. Finally, (3) participants filled in a background questionnaire.

We recruited a different, non-intersecting, group of participants for the remaining tasks. Again, (1) each participant read and signed a consent letter regarding the data collected, similar to the letter used in the other experiment. Then (2) each participant completed four exercises, two covering *SEA_ML* and two covering *SEA_ML++*. Each exercise ended with the user filling in a questionnaire about it. We had a crossover design with four possible sequences, as represented in Table 3. The goal was to mitigate any potential learning effects and balance the number of participants working with each example in each of the possible sequence positions. Finally, (3) participants filled in a background questionnaire.

Table 3: Experimental design. Key: MT = Music Trading; EF = Expert Finder

Sequence	**Task 1**	**Task 2**	**Task 3**	**Task 4**
Group 1	MT/SEA_ML++	MT/SEA_ML++	EF/SEA_ML	EF/SEA_ML
Group 2	EF/SEA_ML	EF/SEA_ML	MT/SEA_ML++	MT/SEA_ML++
Group 3	MT/SEA_ML	MT/SEA_ML	EF/SEA_ML++	EF/SEA_ML++
Group 4	EF/SEA_ML++	EF/SEA_ML++	MT/SEA_ML	MT/SEA_ML

Experimental material We provided each participant with a consent letter and a background questionnaire, which were the same for both experiments. In the symbol selection experiment, the participant also received a questionnaire where (s)he was asked to match each concept definition with the symbol that would best represent its concrete syntax. For the second experiment, the participants received four different scenarios with a corresponding challenge, each followed by a questionnaire about the notation they had just used. Two of those scenarios were related with music trading among software agents, while the other two involved an agent-based expert finding system. Each of these scenarios had two versions, one with *SEA_ML* and the other with *SEA_ML++*. Each participant received two different scenarios for each concrete syntax. All these materials are available in this paper's companion site [1].

[1] `https://doi.org/10.5281/zenodo.1288390`

Participants All the participants in our studies have formal University training in Informatics. For the symbol selection experiment, 25 participants (all undergraduate students) were involved. All of these participants are current or former students at Universidade Nova de Lisboa (UNL). 11 of those had some basic knowledge of MAS (in the context of a course), but not of SEA ML++. For the evaluation experiment, a total of 36 participants were included. That experiment was run in two replicas: The first one was conducted at UNL with 24 participants, including 12 with some basic knowledge of MAS. The second one was conducted at Ege University with 12 participants, all graduate students with some basic knowledge of MAS. All participants were selected through convenience sampling.

Hypotheses, parameters and variables Overall, we hypothesize that the proposed *SEA_ML++* has a *better* concrete syntax than *SEA_ML*. In order to make this more concrete, we anchor our formalized hypotheses on the research goals defined in Section 5.1, as presented in Table 4. For each of the high-level goals, we define the null (H_0Gi) and alternative (H_1Gi) hypotheses (where i denotes the specific goal).

Table 4: Hypotheses

H_{0G1}	The concrete syntax of *SEA_ML++* is **as comprehensible as** the one of *SEA_ML*.
H_{1G1}	The concrete syntax of *SEA_ML++* is **more comprehensible than** the one of *SEA_ML*.
H_{0G2}	The concrete syntax of *SEA_ML++* is **perceived as usable as** the one of *SEA_ML*.
H_{1G2}	The concrete syntax of *SEA_ML++* is **perceived as more usable** than the one of *SEA_ML*.
H_{0G3}	The concrete syntax of *SEA_ML++* is **as effective as** the one of *SEA_ML*.
H_{1G3}	The concrete syntax of *SEA_ML++* is **more effective than** the one of *SEA_ML*.
H_{0G4}	The concrete syntax of *SEA_ML++* is **as efficient as** the one of *SEA_ML*.
H_{1G4}	The concrete syntax of *SEA_ML++* is **more efficient than** the one of *SEA_ML*.

For all hypotheses, the independent variable is the *concrete syntax*, which can be *SEA_ML++* or *SEA_ML*. The dependent variables are different for each of the tested hypothesis.

Comprehensibility. Graphical symbols' *comprehensibility* can be assessed by measuring hit rates, *i.e.*, the percentage of correct responses [19, 20]. In this case, we measure the *hit rate* (percentage of answers where the correct symbol was chosen) for each concept in each of the concrete syntaxes.

Perceived Usability. In order to assess the *perceived usability* we asked our participants to fill in a *System Usability Scale* [4] questionnaire. This questionnaire consists of 10 questions, each with five response options, ranging from *"Strongly Disagree"* to *"Strongly Agree"*. The scores are then converted to a scale of 0-100. The threshold of 68 points is considered the *"average usability"* [4]. Lower scores indicate below average usability, while higher scores are considered above average. In addition, we asked our participants to classify the following three statements:

- ***S1:*** The symbols on the user interface (UI) were easy to **understand**.
- ***S2:*** The symbols on the UI are **adequate** to the MAS constructions they are linked to.
- ***S3:*** The symbols on the UI **helped** me solve the exercise in less time.

We deliberately used the term *"symbols on the UI"* (User Interface) rather than *"concrete syntax"*, as a simplification for our participants, who were not necessarily familiar with the notion of *"concrete syntax"*. For each of these sentences, the participants had to select from a five-point ordinal scale, ranging from one *"Strongly Disagree"* to five *"Strongly Agree"*.

Effectiveness. We use the *correctness* of the answers of our participants to measure how effectively they were able to solve the exercises.

Efficiency. We recorded the *duration* of the working sessions to measure how fast our participants were able to complete their assigned tasks.

5.2 Analysis

Descriptive statistics In this section, we present descriptive statistics for the metrics collected to answer our research questions (Table 5). For each data row, we identify the corresponding **goal** (ranging from *G1* to *G4*), the *dependent variable* (the quality focus for a particular goal), the *independent variable*, *i.e.* the *concrete syntax* followed by the descriptive statistics: the *mean*, *standard deviation* (*SD*), *skewness* (*Skew*), *kurtosis* (*Kurt*) and the *p-value* for the Shapiro-Wilk normality test (*S-W*). In most of these variables, the assumption of normality is **not** reasonable ($p-value < 0.05$), as confirmed by the visual inspection of boxplots, Q-Q plots and kernel density plots, omitted for the sake of brevity.

Hypotheses testing We now present the results of our hypotheses tests.

G1: *RQ1: Are participants more likely to select the correct elements from the PoN-based concrete syntax of SEA_ML++ or the baseline SEA_ML concrete syntax elements?* A Wilcoxon Signed-Ranks test was run and the output indicated that *SEA_ML++* scores ($Mdn = .44$) were statistically significantly higher than *SEA_ML* scores ($Mdn = .16$), $Z = 4.573$, $p < .001$, $r = .83$. This supports our hypothesis that participants were more likely to select the *SEA_ML++* elements.

G2: *RQ2: Do participants using SEA_ML++ perceive it as more usable than SEA_ML?* In order to answer this question, we look at this from two different

Table 5: Selection rate descriptive statistics

Goal	Dependent	Independent	Mdn.	Mean	S.Dev.	Skew.	Kurt.	S-W
G1	**Preference**	**SEA_ML++**	.44	.45	.14	-.10	.07	.457
		SEA_ML	.16	.19	.14	.20	-1.18	.018
G2	**SUS**	**SEA_ML++**	61.25	59.38	19.97	-.20	-.24	.409
		SEA_ML	57.50	54.17	20.62	-.20	.19	.268
	Understandability	**SEA_ML++**	4	3.96	1.09	-1.15	.83	.000
		SEA_ML	3	2.92	1.25	-.04	-.98	.001
	Adequacy	**SEA_ML++**	4	3.65	1.02	-.10	-1.10	.000
		SEA_ML	3	2.96	1.03	-.16	-.11	.001
	Speed	**SEA_ML++**	4	3.83	1.10	-.86	.23	.000
		SEA_ML	3	2.85	1.29	-.09	-.95	.000
G3	**Correctness**	**SEA_ML++**	1.00	.84	.32	-1.763	1.724	.000
		SEA_ML	1.00	.80	.32	-1.509	1.096	.000
G4	**Duration**	**SEA_ML++**	11:51	13:20	06:12	1.520	2.434	.000
		SEA_ML	12:24	14:48	09:32	2.784	8.463	.000

perspectives. We use a standard usability test – the System Usability Scale (SUS) – and a set of three questions to gather more detailed feedback.

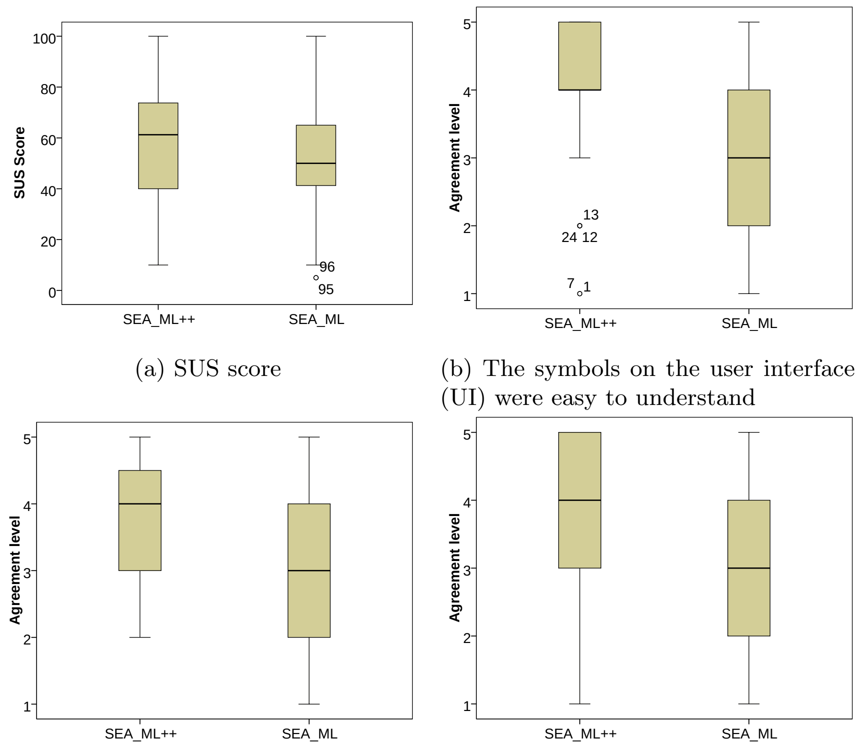
SUS: Is SEA_ML++ perceived as more usable than SEA_ML? The usability did not differ significantly, according to Welch's t test, $t(141.854) = 1.539$, $p = .126$ from SEA_ML++ ($M = 59.38$, $SD = 19.97$) to the usability of SEA_ML ($M = 54.17$, $SD = .20.62$).

Understandability: The symbols on the user interface (UI) were easy to understand. Because the data was skewed for both variables, a Wilcoxon Signed-Ranks Test was run and the output indicated that *SEA_ML++* scores ($Mdn = 4$), were statistically significantly higher than *SEA_ML* scores ($Mdn = 3$), $Z = 3.683$, $p < .001$, $r = .53$.

Adequacy: The symbols on the UI are adequate to the constructs they are linked to. Because the data was skewed for both variables, a Wilcoxon Signed-Ranks Test was run and the output indicated that *SEA_ML++* scores ($Mdn = 4$) were statistically significantly higher than *SEA_ML* scores ($Mdn = 3$), $Z = 2.939$, $p < .003$, $r = .42$. These results suggest that participants found *SEA_ML++* more adequate than *SEA_ML* to the constructs they were linked to.

Speed: The symbols on the UI helped me solve the exercise in less time. Because the data was skewed for both variables, a Wilcoxon Signed-Ranks Test was run and the output indicated that *SEA_ML++* scores ($Mdn = 4$) were statistically significantly higher than *SEA_ML* scores ($Mdn = 3$), $Z = 3.324$, $p < .001$, $r = .48$. These results suggest that participants perceived using *SEA_ML++* had helped them solving the exercise faster than using *SEA_ML*.

G3: We applied the Welch t-test, which is robust to deviations from normality within groups and when variance homogeneity among groups may not be assumed. The correctness does not differ significantly, according to Welch's



(a) SUS score

(b) The symbols on the user interface (UI) were easy to understand

(c) The symbols on the UI are adequate to the constructions they are linked to;

(d) The symbols on the UI helped me solve the exercise in less time

t-test, $t(141.968) = .417$, $p = .519$ from the SEA_ML ($M = .80$, $SD = .32$) to the SEA_ML++ ($M = .84$, $SD = .32$) concrete syntax. These results suggest that there was no difference between the two concrete syntaxes, in terms of *complexity*.

G4: As in *G3*, we applied the Welch t-test. The duration does not differ significantly, $t(122.030) = 1.180$, $p = .280$ from the SEA_ML ($M = 14 : 48$, $SD = 09 : 32$) to SEA_ML++ ($M = 13 : 20$, $SD = 06 : 12$) concrete syntax. These results suggest that there was no difference between the two concrete syntaxes, in terms of *duration*.

5.3 Discussion

Evaluation of the results and implications By using the PoN to guide a redesign of the concrete syntax of SEA_ML we proposed SEA_ML++. We found that (RQ1) the participants in our study were better at correctly identifying the symbols with SEA_ML++. They found the SEA_ML++ syntax (RQ2) easier to *understand*, more *adequate* to the MAS constructs it represents and helpful for performing *faster*, when compared to the the SEA_ML syntax. However, in practice, (RQ3) participants were neither significantly able to use the language more correctly, (RQ4) nor significantly faster using it. So, overall, although the perception of language usage has improved with the new concrete syntax (and, with it, the developer experience), its implications for the actual usage of the language in agent development did not translate into improved effectiveness or efficiency (the small improvements observed were not significant). While it was certainly the case that there was room for improvement of the concrete syntax, the PoN-based improvements only took us as far as improving the perceived developer experience. Other alternative techniques, such as the sign production technique used successfully with other languages, such as *i** [5], could potentially further improve the developer experience. That said, it seems more likely that the effectiveness and efficiency in using SEA_ML++ are mostly constrained by the semantics of the language. Further research is ongoing to explore this hypothesis.

Threats to validity The selection of participants is a potential threat. They are mostly representative of practitioners who are relatively inexperienced with MAS and, therefore, a good match for the main target population of this study. Most of the participants have less than one year experience on software agent development and only five participants in Ege University can be said experienced with having more than three years of MAS knowledge and implementation. As with many other languages, experts will cope better with the peculiarities of a given concrete syntax than newbies. The results obtained in the two replications were very similar, which increases our confidence on their external validity for other inexperienced MAS developers.

A second validity threat concerns the representativeness of the models used for this evaluation. While these models are good representatives of the complexity one would discuss with inexperienced MAS developers in the course of a training activity, further empirical evaluations with models of different complexities will increase the representativeness of this evaluation.

6 Related Work

In the last decade, several MAS modeling languages and DSMLs [3, 8, 12, 15] were proposed to support development of MASs. For example, DSML4MAS [17] introduces a general MAS metamodel with various viewpoints that enable the development of MAS for many application domains. As another example study, in [18], the authors develop a DSML and its supporting tool, called ERE_ML, for MAS working in emergency response environments. However, most of these DS(M)Ls proposed for MASs have been evaluated by just providing a case study demonstrating how the related language can be used for design and implementation of MAS. A quantitative analysis and/or qualitative evaluation considering e.g. the development time performance, generation performance, and/or the usability of the language are not considered in these studies.

In [7], an evaluation framework is proposed which provides the systematic assessment of both the language constructs and the use of agent DSMLs according to various dimensions and criteria. The study also provides an assessment of SEA_ML [6], however, it does not take into account the usability of the language, i.e. usefulness regarding the needs of language users. This evaluation framework is adopted in [23], [21] and [10] for the assessment of the proposed MAS DSMLs. Another MAS DSML evaluation feature exists in [2] for a textual DSL, called JADEL, providing four abstractions, namely agents, behaviours, communication ontologies, and interaction protocols to the well-known JADE agent development framework. However, the study evaluates solely JADEL's code generation performance.

The mentioned studies evaluate their MAS DS(M)Ls to some extent with or without using a structured evaluation framework. However, none of them addresses the usability of the MAS DS(M)Ls considering both the end-user perspective and the improvement of the visual language notation which, we argue that, is critical for the adoption of such languages in Agent-oriented Software Engineering. In this sense, this study contribute to the literature by assessing the usability of an available MAS DSML, namely SEA_ML, and improving its new version.

In general, despite the fact that it is usually claimed that DSLs are more usable and leading to productivity gains, in [11] it has been identified a generalized lack of practice of reporting their usability assessment. The Software Language Engineering community has been seeking for adequate and systematic approaches to evaluating the usability of DSLs. Work was reported [27] on how $i*$ concrete syntax was evaluated using PoN and a new symbol set was proposed for it. In the sequence of this, in [5], it is compared the proposed concrete syntax with alternatives produced by novices (a stereotype and a prototype concrete syntaxes) and the standard $i*$ concrete syntax.

Several modelling languages, for example, BPMN 2.0 [14], Use Case Maps [13], WebML [16], and misuse cases [29], use PoN to evaluate and identify improvement opportunities. It is possible to observe consistently similar conclusions concerning the challenges in most visual notations from a PoN perspective [26]. Other studies assess the $i*$ and KAOS modelling languages [24], using interviews, models creation, and evaluation of those models and the modelling language and found clarity problems in those languages semantics definition.

7 Conclusion and Future Work

There are many modeling languages and DSMLs for Multi-agent Systems. Although there are a few studies addressing the evaluation of MAS DSMLs and their performances, the usability of these DSMLs is not investigated in a systematic way. This study, the principles of The "Physics" of Notations are applied on a MAS DSML, called SEA_ML. By applying nine principles, 43 notations of SEA_ML are evaluated and 32 of them are modified which are used in the development of the new version of SEA_ML called SEA_ML++. In this way the notations in the graphical concrete syntax of the DSML are improved leading to the improvement of SEA_ML++. This hypothesis is examined under four research goals covering comprehensiveness, usability, effectiveness, and efficiency. The experiment conducted by the participants shows that the participants were more likely to select the SEA_ML elements and the symbols were easy to understand. However, the results show that there was no significant difference between the two concrete syntax, in terms of complexity and duration.

Acknowledgment

The authors would like to thank the followings: i) the Scientific and Technological Research Council of Turkey (TUBITAK) under grant 115E591, and ii) Portuguese grants NOVA LINCS Research Laboratory (Grant: FCT/MCTES PEst UID/ CEC/04516/2013) and DSML4MA Project (Grant: FCT/MCTES TUBITAK/0008/2014).

References

1. Basili, V., Caldiera, G., Rombach, H.: Goal Question Metric Paradigm. Encyclopedia of Software Eng. **1**, 528–532 (2001)
2. Bergenti, F., Iotti, E., Monica, S., Poggi, A.: Agent-oriented model-driven development for jade with the jadel programming language. Comput Lang Syst Str **50**, 142–158 (2017)
3. Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J.J., Pavon, J., Gonzalez-Perez, C.: Faml: a generic metamodel for mas development. IEEE T Software Eng **35**(6), 841–863 (2009)
4. Brooke, J., et al.: Sus-a quick and dirty usability scale. Usability evaluation in industry **189**(194), 4–7 (1996)
5. Caire, P., Genon, N., Heymans, P., Moody, D.L.: Visual notation design 2.0: Towards user comprehensible requirements engineering notations. In: RE'13. pp. 115–124. IEEE (2013)
6. Challenger, M., Demirkol, S., Getir, S., Mernik, M., Kardas, G., Kosar, T.: On the use of a domain-specific modeling language in the development of multiagent systems. Eng Appl Artif Intel **28**, 111–141 (2014)
7. Challenger, M., Kardas, G., Tekinerdogan, B.: A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems. Software Qual J **24**(3), 755–795 (Sep 2016)
8. Ciobanu, G., Juravle, C.: Flexible software architecture and language for mobile agents. Concurr Comp-Pract E **24**(6), 559–571 (2012)
9. Da Silva, V.T., Choren, R., De Lucena, C.J.: Mas-ml: a multiagent system modelling language. IJAOSE **2**(4), 382–421 (2008)
10. Faccin, J., Nunes, I.: A tool-supported development method for improved bdi plan selection. Eng Appl Artif Intel **62**, 195–213 (2017)
11. Gabriel, P., Goulão, M., Amaral, V.: Do software languages engineers evaluate their languages? In: Proceedings of the XIII Congreso Iberoamericano en "Software Engineering" (CIbSE'2010) (2011)
12. Gascueña, J.M., Navarro, E., Fernández-Caballero, A.: Model-driven engineering techniques for the development of multi-agent systems. Eng Appl Artif Intel **25**(1), 159–173 (2012)

13. Genon, N., Amyot, D., Heymans, P.: Analysing the cognitive effectiveness of the ucm visual notation. In: International Workshop on System Analysis and Modeling. pp. 221–240 (2010)
14. Genon, N., Heymans, P., Amyot, D.: Analysing the cognitive effectiveness of the bpmn 2.0 visual notation. In: Proceedings of the Third International Conference on Software Language Engineering. pp. 377–396 (2010)
15. Gonçalves, E.J.T., Cortés, M.I., Campos, G.A.L., Lopes, Y.S., Freire, E.S., da Silva, V.T., de Oliveira, K.S.F., de Oliveira, M.A.: Mas-ml 2.0: Supporting the modelling of multi-agent systems with different agent architectures. J Syst Software **108**, 77–109 (2015)
16. Granada, D., Vara, J.M., Brambilla, M., Bollati, V., Marcos, E.: Analysing the cognitive effectiveness of the webml visual notation. Softw Syst Model **16**(1), 195–227 (2017)
17. Hahn, C.: A domain specific modeling language for multiagent systems. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1. pp. 233–240 (2008)
18. Hosein Doost, S., Adamzadeh, T., Zamani, B., Fatemi, A.: A model-driven framework for developing multi-agent systems in emergency response environments. Softw Syst Model pp. 1–28 (2017)
19. ISO: Standard graphical symbols: Safety colours and safety signs–registered safety signs (iso 7010: 2003). International Standards Organisation (ISO): Geneva, Switzerland (2003)
20. ISO: Iso standard graphical symbols: Public information symbols (iso 7001:2007). International Standards Organisation (ISO): Geneva, Switzerland (2007)
21. Kardas, G., Bircan, E., Challenger, M.: Supporting the platform extensibility for the model-driven development of agent systems by the interoperability between domain-specific modeling languages of multi-agent systems. Comput Sci Inf Syst **14**(3), 875–912 (2017)
22. Kardas, G., Gomez-Sanz, J.J.: Special issue on model-driven engineering of multi-agent systems in theory and practice. Comput Lang Syst Str **50**, 140–141 (2017)
23. Kardas, G., Tezel, B.T., Challenger, M.: Domain-specific modelling language for belief-desire-intention software agents. IET Softw (2018). https://doi.org/10.1049/iet-sen.2017.0094
24. Matulevičius, R., Heymans, P.: Comparing goal modelling languages: An experiment. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. pp. 18–32 (2007)
25. McGuinness, D.L., van Harmelen, F.: Owl web ontology language overview. w3c (2004)
26. Moody, D.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE T Software Eng **35**(6), 756–779 (2009)
27. Moody, D.L., Heymans, P., Matulevičius, R.: Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation. Requir Eng **15**(2), 141–175 (2010)
28. Pokahr, A., Braubach, L., Walczak, A., Lamersdorf, W.: Jadex-engineering goal-oriented agents. Developing multi-agent systems with JADE pp. 254–258 (2007)
29. Saleh, F., El-Attar, M.: A scientific evaluation of the misuse case diagrams visual syntax. Inform Software Tech **66**, 73–96 (2015)
30. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intell Syst **21**(3), 96–101 (2006)