Evaluating GISMO

Usability Testing of the GISMO DSML – Results of a Pilot Study with Non Expert Scientists

Romuald Deshayes and Tom Mens
University of Mons

Abstract

In order to assess the "usability in use" of GISMO, a domain-specific modelling language for expressing gestural interaction, we have used the technique of "usability testing", well-known in user-centered interaction design to evaluate a product on its potential users. The experiment was carried out with 12 people, all students or researchers in computer science or mathematics. The results of the evaluation are reported upon in this technical report.

1 Introduction

GISMO is a domain-specific modelling language (DSML) that we have developed for specifying gestural interaction in interactive applications. The language has been described in [4], and is intended to be used on top of a gestural interaction framework described in [5]. The purpose of GISMO is to make the specification of gestural interaction models easier to domain experts. An example of a GISMO model, specifying the interaction behaviour of an archer, as part of some computer game, is shown in Figure 1.

2 Usability Testing

To evaluate the "usability in use" of GISMO, we decided to resort to the technique of "usability testing", a well-known technique in user-centered interaction design to evaluate a product on its potential users. More specifically, we resorted to "hallway testing": we asked a small group of randomly-selected people down the hall to test the usability of GISMO. Since we are working in a University, those random people were all students or researchers in computer science or mathematics.

We asked all these persons to conduct a short survey and to carry out the activity of developing a gestural interaction model with GISMO in order to get insight in the usability of the language and to get informed about any issues with the proposed language.

The evaluation experiment was structured in four steps.

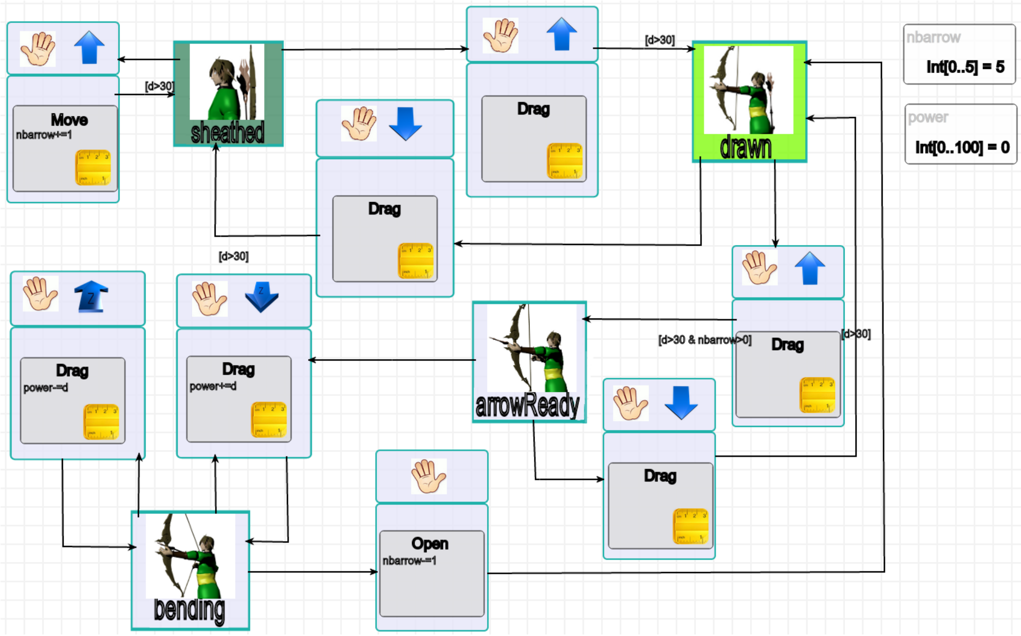


Figure 1: GISMO model specifying the gestural interaction to control an archer.

1. First, a live 15-minute presentation of GISMO was given to each participant with the aim to present the main features of GISMO, since none of the participants had previous experience with GISMO or even with gestural interaction applications. The abstract and concrete syntax of GISMO was presented, and a concrete example of a bow model (see Figure 1) was presented. This example was used to explain how to design interaction models, how to connect the model to a target application thanks to the automatic generation of an interface and the use of a client/server connection, and how to specify and verify domain-specific properties on a GISMO model.

2. After the presentation, the participants were required to respond to a short survey, containing a set of general questions to inform about their background knowledge and their initial perception about the usability of the GISMO language.

3. Next, each participant was asked to create a simple interaction model specifying the interaction of a user with a virtual book. The textual specification given to the participants was as follows:

   > "By default, the book is stored in a bookshelf. A drag of the right hand towards the user's body has the effect of retrieving the book and place it in closed position in front of the user. At this time, it is possible to store the book back in the bookshelf by performing the inverse gesture (dragging the right hand towards the screen) or to open the book by dragging both hands away from each other. Once opened, the book can be closed back by dragging both hands towards each other. When it is opened, pages can be turned by moving the right hand from right to left or from left to right, thus changing the currently visualised page."

   A correct version of the corresponding GISMO model is shown in Figure 2.

4. After this modelling task, a second short survey was given to the participants, in which they were required to respond to set of more precise questions about GISMO.

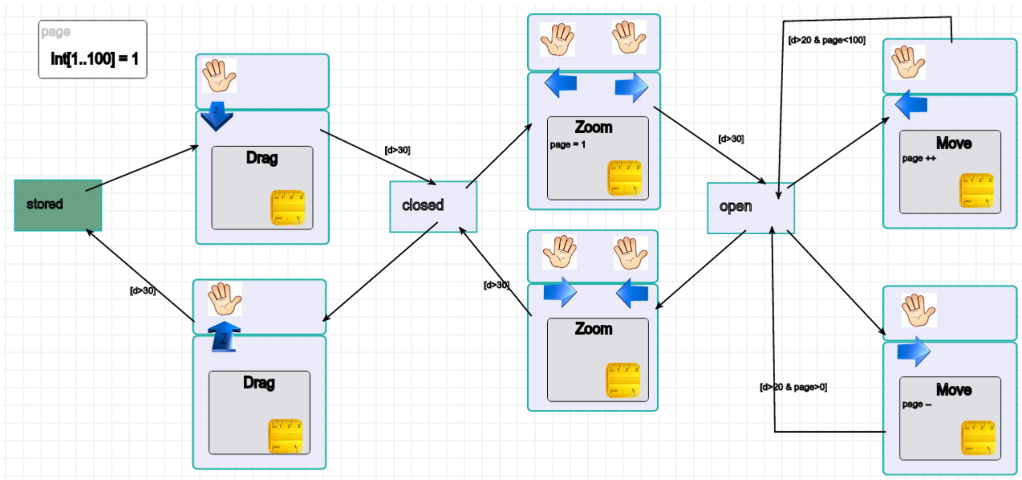


Figure 2: The book model which was asked to model by the assessors. Correct version

3 Desired characteristics of GISMO

Our main goal was to evaluate the "usability in use" of the GISMO language, as defined by the ISO 9241-11 standard. This quality characteristic is subdivided into three usability measures:

- *Efficiency*: "evaluate if the system as a whole can be used to retrieve information efficiently". It can be measured by how quickly the user performs a task (time taken and number of steps).
- *Effectiveness*: "evaluate if the system as a whole can provide information and functionality effectively". It can be measured by counting the amount of errors during the task and after task completion.
- *Satisfaction*: "look into the areas of ease of use, organization of information, clear labelling, visual appearance, contents, and error corrections". It can be measured through questionnaires using Likert scales.

The evaluation experiment enabled us to evaluate these three usability measures as follows:

- *Efficiency* was measured by recording the duration of the modelling task (step 3), in minutes.
- *Effectiveness* was measured by counting the amount of errors made during the process of creation of the GISMO book interaction model, as well as the number of errors remaining in the final version of this GISMO model produced by the participant. Errors can be syntactic (e.g., forgetting an arc linking a state and a gesture) or semantic (having a working model that does not correspond exactly to its specification). We considered both types of errors, except that for semantic errors, small conceptual differences were accepted (e.g., using a *drag* instead of a *move* gesture for turning the book's page was not considered as a mistake).
- *Satisfaction* was evaluated by means of the responses to the survey questions. Most

responses used Likert scales, with answers ranging from 1 to 5, 1 being the most negative and 5 the most positive answer. Likert scales are commonly used in satisfaction assessment, as they provide an easy way of categorising participants' answers.

4 Results of the evaluation

4.1 Profile of the participants

In total, the evaluation was conducted on 12 participants. To avoid bias, none of the participants was previously involved in any way in the development or evaluation of GISMO or related artefacts. Out of convenience, all selected participants were working or studying in the department of computer science or mathematics at the University of Mons, because this is where the GISMO language has been designed. 5 participants were researchers in computer science (either postdoc or PhD students), 4 participants were PhD researchers in mathematics, and 3 participants were master students in mathematics. Out of the 12 participants, 8 were male and 4 were female.

For an initial usability evaluation, a total of 12 participants is more than sufficient, since the goal is not to get any statistically relevant results, but rather to obtain a fast and early feedback on the main features of GISMO and on how its usability is perceived by non experts.

During the initial survey, participants were asked about their existing familiarity with three well-known visual formalisms: Finite State Machines (FSM), (UML) statecharts, and Petri nets. This question is relevant since GISMO's syntax is based on the notion of states and events, while its underlying semantics is based on formal languages. Participants with some knowledge on state-based or event-based formalisms and visual languages are therefore more likely to achieve good results while designing GISMO models than participants who do not. The results are summarised in Figure 3. As could be expected given their education background, most respondents were familiar with FSM, since it was part of at least one course in their current or past course programme. Only one participant had no previous knowledge in any of the proposed formalisms.

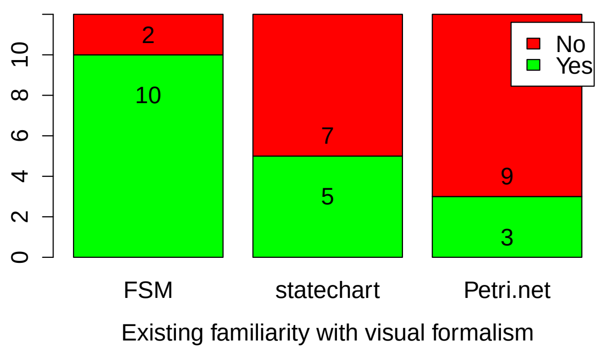


Figure 3: Familiarity with visual formalisms

Another initial question was “Do you have a clear idea of the complexity involved in creating gesture-based applications ?” The responses are summarised in Figure 4 using a Likert scale. Only one participant gave a maximum score of 5, since he was fully aware of the complexity involved in developing gesture-based applications, having developed an interactive gestural application (unrelated to GISMO) in the context of his Masters thesis. The majority of the other participants did not have a clear idea since they have never been involved in gestural interaction applications. Globally, computer scientists claimed to have a better idea of the complexity involved than mathematicians (with a mean value of 2.8/5 for computer scientists and 2/5 for mathematicians). All participants have learned at least one programming language, but we can clearly claim that they are mostly non expert in developing interactive applications.

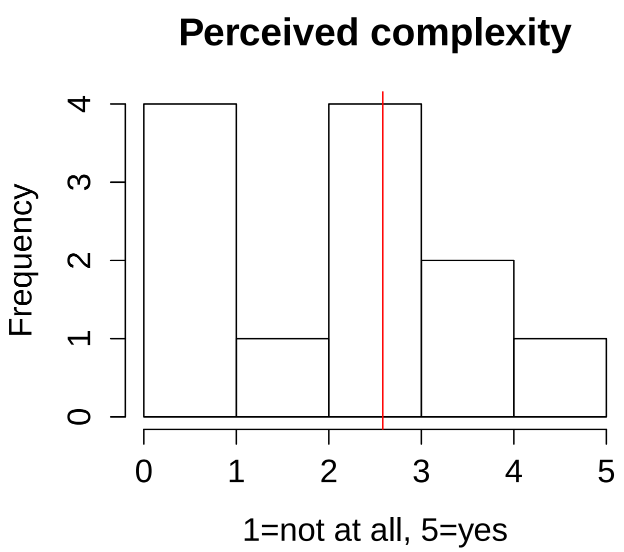


Figure 4: Answers to question“Do you have a clear idea of the complexity involved in creating gesture-based applications?” using a Likert scale. The red vertical line represents the global mean value of 2.6/5.

4.2 Assessing “efficiency”

To assess efficiency, we measured the time taken by the participants to create the virtual book interaction model. Table 1 shows, among others, the time (in minutes) taken by the participants to create the book model. A median time of 12 minutes was needed. No significant differences were observed base on the user’s profile (e.g., based on gender, occupation or scientific discipline). The modelling task being rather small, the time taken by every participant was very consistent, as shown by the boxplot in Figure 5. Most participants took between 10 and 13 minutes to complete the task, which is fairly low.

As a matter of comparison of the time it would take to carry out the same task with a traditional general-purpose programming language, the first author has implemented the virtual book model in Java. To this extent, he made use of the SwingStates API [2], which adds the notion of state machines to the Java Swing user interface toolkit. This significantly facilitates the coding of executable statecharts in Java, without the need to

Discipline	CS	CS	CS	CS	CS	MA	MA	MA	MA	MA	MA	MA
Occupation	RES	RES	RES	RES	RES	STU	RES	RES	RES	STU	RES	STU
Time taken (in minutes)	9	13	14	12	12	12	12	12	11	10	13	13
Errors during	1	0	1	1	1	0	0	1	0	0	1	2
Errors after	0	0	0	0	0	0	0	0	0	0	0	0

Table 1: Time taken by participants to model the book example, number of errors during modelling, and remaining errors by the end of the modelling task. Legend: CS = Computer Scientist, MA = Mathematician, RES= Researcher, STU = Student.
FOR THE SAKE OF COMPLETENESS ADD A *GENDER* LINE AS WELL

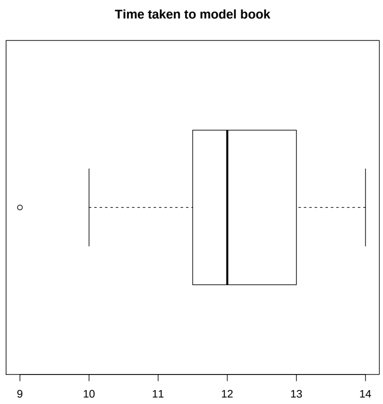


Figure 5: Boxplot of the time (expressed in minutes) taken for carrying out the modelling task in GISMO (virtual book interaction model).

resort to a visual modelling language. Being expert in the development of interactive applications involving gestures, and being used to SwingStates, it took the first author 40 minutes and about 110 lines of code to create a Java program equivalent to the book model created in GISMO. This is more than 3 times longer (for an expert developer) than the average time it took (for non experts in gestural interaction modelling) with GISMO to carry out the same task.

Of course, comparing a visual DSML approach with a classical programming language should be done with care, to avoid comparing apples with oranges. The above experiment only intends to give a rough idea of the difference in complexity between having to code the interaction using a classical textual programming language as opposed to using the GISMO DSML. To make the comparison fair, we did not require for the Java-coded approach to rebuild the entire GMOD framework. We made the assumption that a Java version of GMOD was available and that high-level gestures were available as an API in Java. In this way, we only need to code an interaction model (equivalent to the one in GISMO) in Java, and the comparison between both approaches makes more sense. The Java code for the virtual book interaction model is shown in Appendix B. In addition to the time taken to write the Java code, the resulting Java code is also harder to read than the equivalent GISMO model. Even when developed in a high-level textual programming language such as Java, with a complete available API for implementing statecharts, it is harder for non experts to read and understand the models compared to GISMO. **TOM:**

This last sentence is an unsubstantiated claim. In fact, you should ask all non experts with of the variants, Java or GISMO is harder to understand. Since all survey respondents now know Java AND GISMO, you could ask them and report on the results here...

4.3 Assessing "effectiveness"

Effectiveness can be assessed by counting the number of errors while designing a model, and once the model is considered finalised by the designer. We only considered syntactic errors, and errors that would lead a semantically incorrect model or would simply prevent the model from running (such as not connecting the states to gestures). Possible such errors can be, among others, errors in the precondition of a gesture (e.g. a minimum distance threshold to allow the state change), missing variables, and inconsistencies w.r.t. the specification (such as associating the wrong direction to a gesture, which results in an interaction model that does not behave as expected according to the specification).

In a few cases, we accepted semantic errors as being conceptually alternative design choices. For example, while designing the bow model, some participants chose a move gesture to turn the pages, while other chose a drag gesture. Both solutions were considered as being correct.

After looking at the results in Table 1, we notice that the number of errors in the final model were very low (sometimes 1 small error). While the modeling task was fairly simple, thus limiting the amount of possible errors, the requested model has the size of a typical object to be modelled in a gaming environment.

All identified errors occurred during the modelling task (see Figure 6 on the left, depicting the total amount of errors committed *during* modelling). Five participants did not make any error, 6 participants committed only one error and one participant committed 2 errors. The most frequently occurring error (made by 6 out of 7 participants) was forgetting to add a global variable representing the current book page.

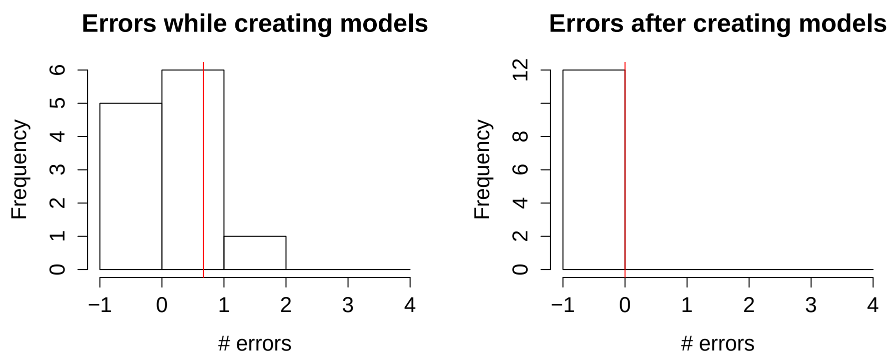


Figure 6: Number of errors while designing the book interaction model (left), and after finishing the model (right).

Interestingly, none of the final models created by the participants contained any remaining errors (see Figure 6 on the right). For example, during the modelling phase, some participants forgot to add a state or gesture, but corrected their mistake when reading again the requirements specification of the modelling task.

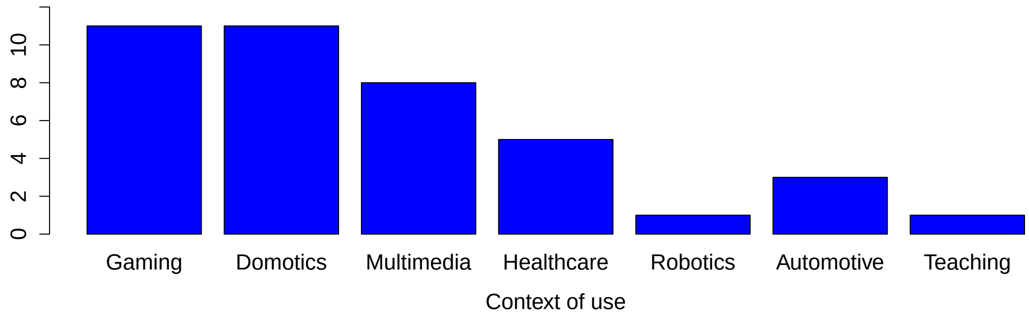


Figure 7: In what type of context do you think GISMO might be useful ?

Gaming and domotics are, according to nearly all participants (see Figure 7), adequate contexts for applying GISMO. In these contexts, we would model rather simple interaction models, with a fairly low amount of states and gestures. In both contexts, too many states and possible actions would make it difficult for the user to have a complete mental representation of what are the possible ways of interacting with a certain object (e.g. locked chest or door in a gaming context, shutters or lights in a domotics context).

4.4 Assessing "satisfaction"

To evaluate the global satisfaction, we processed the questionnaires and derived histograms to show the repartition of answers among assessors. Let us recall from Section 2 that the evaluation of GISMO was done in four steps. Step 2 involved a survey with series of general questions, after having presented the GISMO language to the participants. These questions are listed in Appendix A.1. Step 4 involved more specific questions about GISMO, after each participant created the virtual book interaction model in Step 3. Some questions asked in Step 2 were asked again, considering that the users' experience was likely to have changed after creating their own models. Twelve additional questions are asked to the participants, as described in Appendix A.2.

Assessing global satisfaction First we asked the participants whether they felt comfortable with GISMO, and if they could easily understand the presented models. We can clearly see (Figure 8-Left) that most participants have a very good understanding of GISMO models. As another measure, the global satisfaction about using GISMO is depicted in Figure 8-Right. This question was asked after a presentation of the language and its features, and after realising the modelling task of the book interaction model. We can clearly see that everyone was globally satisfied about her experience with GISMO.

Making links between data **TOM: I HAVE READ THIS ENTIRE PARAGRAPH AND IT IS MORE CONFUSING THAN ANYTHING ELSE I SUGGEST TO COMMENT IT OUT COMPLETELY!** In order to try to establish a

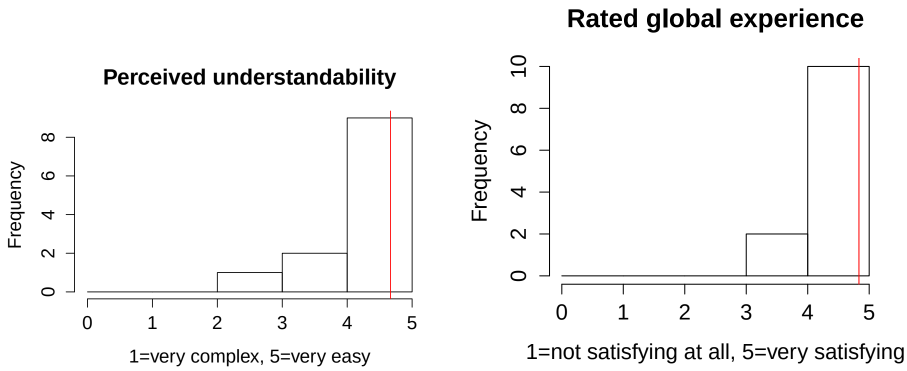


Figure 8: After the presentation of the language and of some examples, do you think GISMO models are easy to understand ? (Left) How would you rate your global experience with GISMO to model gestural interaction ? (Right)

link between the participants' profile and some other specific questions, we compared the answers to question 5 (see figure 4) with more specific questions. As we will see with the histogram related to all questions**TOM:WHICH HISTOGRAM?**, the results are mostly homogeneous and good. In this case, it is not necessary to try to separate the participants into different profiles, since they all seemed to give similar answers. Histograms are thus sufficient to show the distribution of answers for each questions.

Although we cannot say much about scatter plots, the most relevant one is depicted in Figure 9. It shows on the abscissa the perceived complexity, related to the perceived ease of integrating GISMO in software applications. While all but one participant gave a score of 4 or 5 to the easiness question, the perceived complexity is much more scattered. We can see in this graph that people with a good idea about the complexity involved in creating gesture-based application also found it very easy to integrate GISMO with target application. On the other hand, the only participant that gave a note of 3/5 on the easiness to integrate is also not so aware of the actual complexity involved in creating gesture-based applications (he also gave a note of 3/5 to this question).

Separating computer scientists and mathematicians, or students and researchers mostly did not have an important impact. Since we only had 12 participants, the data are insufficient to assess if computer scientists are more or less comfortable using GISMO. This additional reason makes it even more straightforward that only simple histograms should be used to display the answers given by the participants.

GISMO's use and adequacy The next question relates to GISMO's adequacy for expressing gestural interaction. After the presentation of GISMO, but before creating their own models, participants already agreed that GISMO is adequate, with an average score of 4.08/5.**AVERAGES ARE NOT MEANINGFUL WITH DISCRETE SCORES, USE MEDIAN INSTEAD** After the exercise, the same question was asked. The results became even better with an average answer of 4.42/5.**AVERAGES ARE NOT MEANINGFUL WITH DISCRETE SCORES, USE MEDIAN INSTEAD** We can thus say that after a small training session, participants were mostly thinking that GISMO is adequate for expressing gestural interaction.

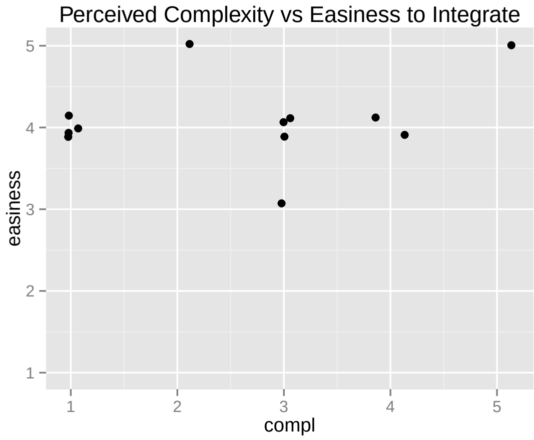


Figure 9: Scatter Plot of “Do you have a clear idea of the complexity involved in creating gesture-based applications ?” vs “Do you think it is easy to integrate GISMO in software applications (in particular thanks to the automatically generated interface)”

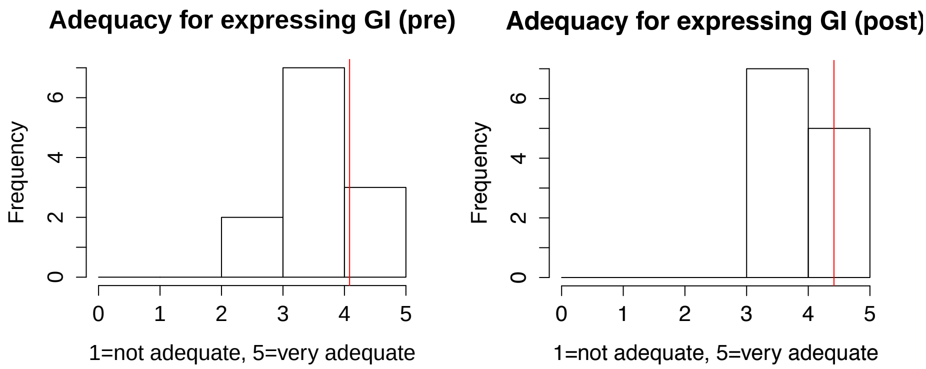
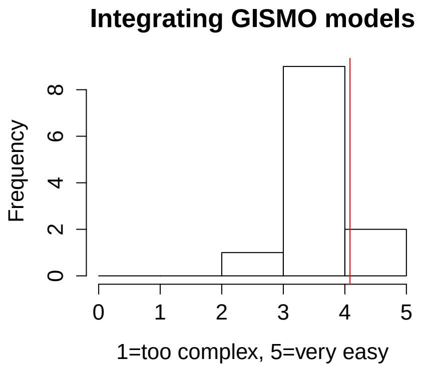


Figure 10: Question : Do you think GISMO is adequate for expressing gestural interaction with objects ? (before and after the exercise of modelling the book model).

Already depicted in the scatter plot of Figure 9, the histogram in Figure 11 (left) shows how easy do participants think that it is to integrate GISMO in a software application. With an average answer rated at 4.08/5 **USE MEDIAN INSTEAD**, we can fairly say that most participants agreed on their answer. Whether they would actually use it if they had to develop gesture-based application is the next question that was asked. Its results are displayed in Figure 11 (right). With an average answer of 4.25/5 **USE MEDIAN**, many people are convinced enough by the GISMO approach that they would actually use it, because they find it easy and straightforward to use. Two participants gave a score of 3/5 and justified their answers by saying that they would first have a look at the existing possibilities to develop gesture-based applications before making a decision. These two participants are both researcher in mathematics that do not have a clear idea of the complexity involved in creating gesture-based applications.

When asking how much GISMO models are easy to use and create (Figure 12), before the modeling task participants agreed with an average rating of 4,08/5 **USE MEDIAN** that models are fairly easy to create. After the modelling task, the rating became even



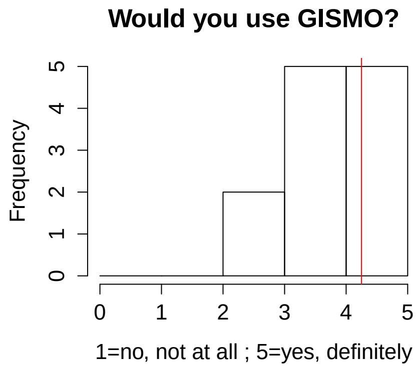
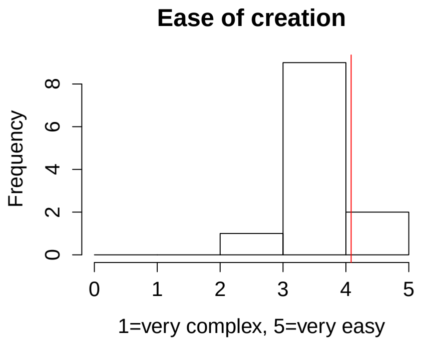


Figure 11: Do you think it is easy to integrate GISMO in software applications ? (Left) Would you use GISMO if you have to integrate gestural interaction in your applications ? (Right)



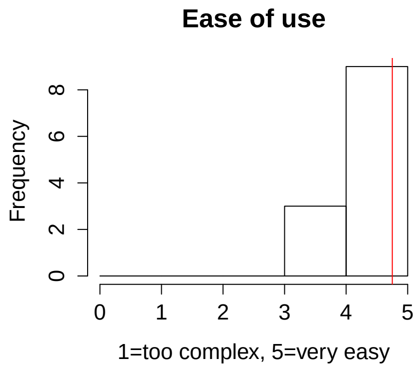


Figure 12: Do you think it is simple to create GISMO models to specify 3D gestural interaction with objects ? (Left, before exercise) Do you think GISMO is easy to use ? (Right, after exercise).

better, as 9 participants over 12 gave a 5/5 rating to the question. The average response rating is 4,75/5 **USE MEDIAN** which is very good and comforts us in our belief that GISMO is promising and well received by non-experts in gesture-based development.

Figure 13 depicts the participants' perception about visual syntax. Most participants found the visual syntax to be adequate with a mean value of 4,41/5 **USE MEDIAN**. Participants also agreed that the visual syntax is easily changeable (mean value of 4,08/5) **USE MEDIAN**. The lowest notes (2 participants gave a 2/5 note to the second question) were given by two mathematicians, one student with almost no background in software development and one researcher with low background in software development. Since changing the concrete syntax may require modifying the GISMO metamodel, it appeared more complex for them to achieve. Changing the icons of GISMO is achievable by anyone, but more complex modifications such as replacing the precondition from text to a more visual representation would indeed involve some expertise in the development of DSML and should thus be left to language developers.

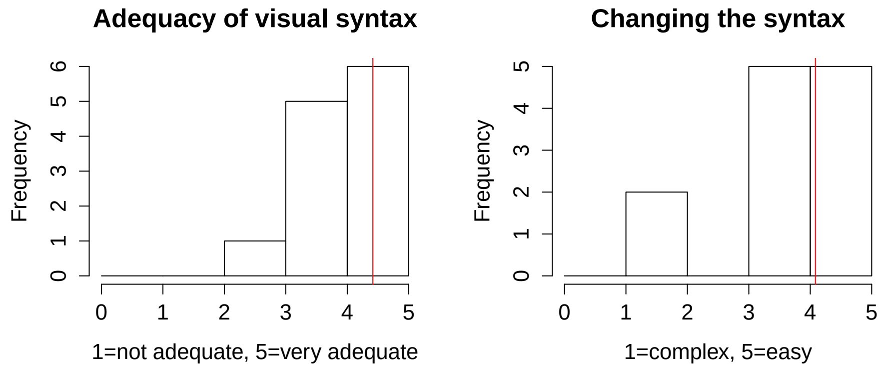


Figure 13: Do you think the visual syntax is adequate ? (Left) Do you think changing the concrete syntax of GISMO is easily achievable ? (Right)

Properties in GISMO During the presentation of the GISMO language, we explained to the participants how specification and verification of domain-specific properties can be achieved on GISMO models, thanks to the ProMoBox approach [6]. Properties can be expressed at the DSML level, and potenial counter-examples can be viewed as a trace of execution of a GISMO model. Most participants were really satisfied with this approach as they gave an average rating of 4.25/5 to the question**USE MEDIAN**. Researcher in mathematics give a higher rating to the question (4.5/5 on average)**USE MEDIAN** since most of them have some knowledge about model checking and found it very convenient to express properties at a higher level of abstraction than by using logic-based languages such as LTL.**TOM: LTL is not a "language"**

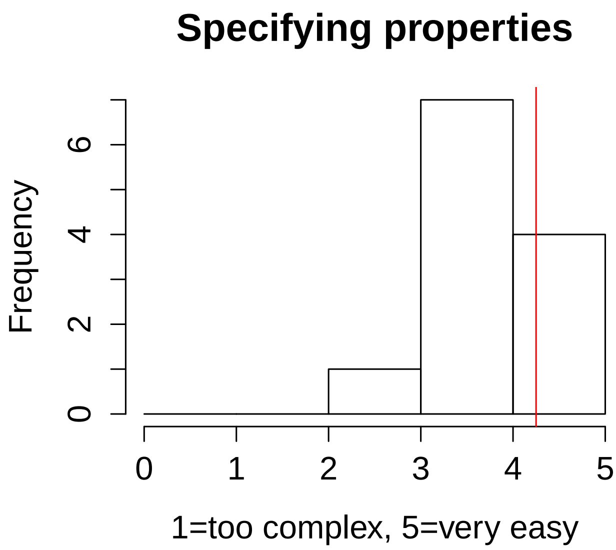


Figure 14: Accoding to you, how easy is it to specify properties in GISMO ?

State-based formalism All but one participants were familiar with state-based formalisms (see Figure 15). With an average score of 4.5/5**USE MEDIAN**, they seemed convinced that using a state-based formalism for modelling gestural interaction is adequate. This supports the claims defended by Buxton [3] that state-based and event-based approaches are adequate for expressing interaction.

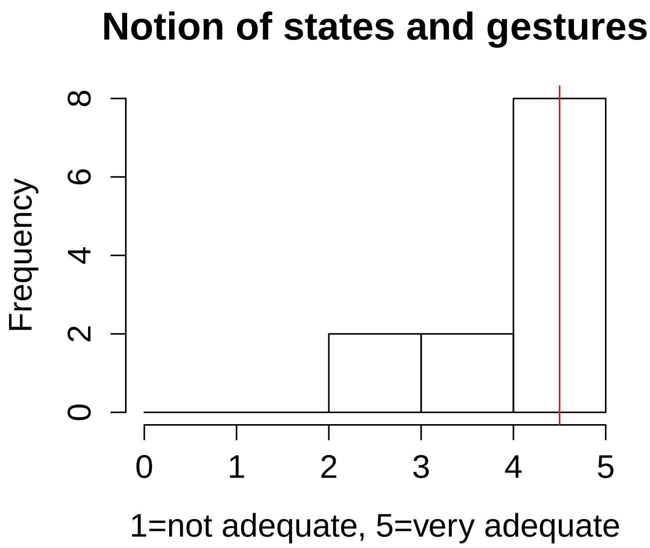


Figure 15: According to you, is the notion of states and gestures adequate ?

Particular participants Now that all questions of the survey has been reviewed, let us have a closed look to two particular cases.

The first one is a mathematics student with no knowledge of any state-based formalism (she only had 1 year of Python programming, then took a physics orientation). She also has no idea of the complexity involved in creating gesture-based applications (she gave a 1/5 answer). When looking at her answers, she gave a 4/5 or 5/5 to every question except for the easiness to specify properties (3/5). She added in free comments that GISMO is a really interesting approach and that it could be used to teach programming to secondary school students.

The second case is also a student in mathematics who, as opposed to the previous case, is involved in the creation of gesture-based application in the context of his Masters thesis. His actual work does not involve any DSML nor any formal models. Everything in his project is coded in C++. He reviewed GISMO with an important interest and gave a 5/5 note to all questions, arguing that everything was simpler with GISMO than with traditional techniques based on coding and using classical code-based APIs. Of course, we cannot generalise any conclusions from these two specific participants, but it still provides us some very interesting feedback on how people react to GISMO, depending on their technical background.

Analysing free comments We received some textual feedback by users who answered the open questions 13 and 15. Let us review and argue the different comments that were given.

Question 13 asked how could GISMO be improved:

- A first feedback indicated that the notion of guard could not be intuitive for people that are not familiar with formal languages. A more visual representation could help reducing the difficulty to express such preconditions.
  A: Having a more visual representation for precondition is indeed an interesting remark and could be done as future work.

- Being able to combine gestures could reduce the number of transitions, especially if different gestures can lead to the same destination state.
  A: The notion of "or" is actually expressed by creating multiple gestures sharing the same source and destination states. Improving GISMO's syntax in such a way that different gestures could be expressed in a unique box is a good idea, although it could also reduce readability if we have more than 2 possible gestures.

- Creating composite gestures in GISMO to manipulate higher-level symbols, by combining a sequence of gestures and associate it to a new high-level gesture with a particular icon syntax.
  A: This is something we have already thought about and that is planned as future work.

- The notion of state could not always be adequate, especially for more complex interaction, such as controlling a robot.
  A: State-based languages have multiple advantages and also have their limitations. GISMO does not target complex and really precise gestural interaction which is typically needed in robotics.

- GISMO should be integrated as part of the modelling framework so that we can create an IDE that uses GISMO itself to create GISMO models.
  A: This is a really interesting feedback. We have made some research on this topic which lead to an exploratory research paper [1].

- Being able to automatically propose a layout for organising the GISMO model that is being created.
  A: Some state of the art techniques exist to provide such automatic layout. This could be a future feature of GISMO in a later version.

Question 15 asked about the important elements which are not present in GISMO and that participants would like to find in such language. Less comments were made here since most participants are not expert in the development of gesture-based applications:

- One participant realised that independent models could run in parallel but made a point about the impossibility to communicate between the different models.
  A: At this moment, we do not have communicating models, and we are fully aware that this is an issue for scalability. However, our underlying framework (GMOD) already has such feature. It has just not been integrated in GISMO yet.

- Another participant wondered about the ability to cast actions when in a certain state.
  A: With 15 minutes of presentation, it was not possible to present all existing features

of GISMO. This is however possible by using the exit action code, which can refer to an object of any type, on which methods can be called. This mechanism can be used to cast messages.

- The last remark is related to GISMO's syntax and is concerned about being able to have many outgoing arrows from a single gestures to different states, according to different preconditions.
  A: Once again this is an interesting feature which could be added, but can also lead to limit the readability of GISMO models. These kind of features have to be assessed and reviewed before being integrated.

5 Conclusion

In this technical report, we have assessed the usability in use of GISMO, a DSML for expressing gestural interaction. By studying efficiency, effectiveness and satisfaction on 12 participants that had no experience in the development of interactive applications (except for 1 participant), we can conclude that GISMO has been well received by the participants, and that they are globally very satisfied of their experience with the language.

At the level of efficiency, we measured that, in average**TOM: NO, MEDIAN**, only 12 minutes are needed to create a fully functional model of a simple book in a virtual application. Coding the same model in Java took 40 minutes by the first author, who is an expert in the development of gesture-based applications. Models created in GISMO are also more easy to read and understand because the proposed visual approach helps the developers to have a global view over the possible states and gestures of the model, which is not the case of a coded approach, which requires more time and effort to process and understand.**TOM:THAT IT REQUIRES MORE TIME TO UNDERSTAND HAS NOT BEEN SHOWN OR VERIFIED**

Considering effectiveness, the amount of errors during the task of designing a virtual book interaction model and after finishing its creation have been recorded. By the end of the exercise, none of the 12 participants validated an incorrect model, thus leading to a perfect score. During the completion of the task, 6 participants forgot to model the active page of the book with a variable. They all discovered the mistake before validating their model by proofreading the exercise statement. These results are very good, but they must be mitigated by the fact that only one simple exercise was given to the participants. However, in a gaming context, the possible interaction with virtual objects are generally limited, thus resulting in small and easy to design models. One should also consider that only 15 minutes of training was given before asking the participants to create their own model, implying a very low learning curve. During the creation of the models, a certain freedom was given when it comes to the gesture allowing to turn the pages. The exercise statement stated that "when the user moves his left hand from left to right (resp. from right to left) the next (resp. previous) page of the book should be displayed. Some participants used a *move* gesture while others used a *drag* gesture to turn the pages. We did not consider this as an error since it is more a design choice due to an incomplete requirements specification. It is also one of the goals of designing executable models to give the domain expert the ability to execute or simulate his designed models and to facilitate exploration of different design alternatives.

Assessing the satisfaction of participants was done through questionnaires, mostly using Likert scales. Global satisfaction is very high, with an average value of 4,83/5.**TOM: USE MEDIAN** Participants also believed that GISMO is adequate for expressing gestural interaction with an average value of 4.42/5 **TOM: USE MEDIAN** after creating their own GISMO model. Trying to separate participants according to their profile did not give any interesting results since everyone mostly seemed to give good notes for all the questions they were asked.

The low number of participants makes it difficult to make any generalised conclusion. All we can say is that GISMO seems to raise the interest of all the involved participants. Based on that conclusion, we believe that continuing its implementation towards a full-fledged tool for prototyping gesture-based applications is the way to go.

Some participants gave some pieces of advice that were interesting for the future versions of GISMO. It was interesting to see that these participants seemed to propose improvements which we were already planning to work on as future work. This is especially the case for features including creation of composite gestures and communication between models.

In conclusion, thanks to this usability testing of GISMO, we believe that using a DSML approach for developing gesture-based application is a good approach. In particular, non experts were convinced that it is easy to create GISMO models and to use them for developing gesture-based applications.

Appendices

A List of survey questions

A.1 List of questions asked after presentation of GISMO

1. What is your discipline?
2. What is your occupation ?
3. Are you familiar with at least one of these formal languages ? (FSM, UML Statecharts, Petri Nets, Other)
4. Have you ever developed interactive applications involving gestures or touch ?
5. Do you have a clear idea of the complexity involved in creating gesture-based applications ?
6. After the presentation of the language and of some examples, do you think GISMO models are easy to understand ?
7. After the presentation of the language and of some examples, do you think it is simple to create GISMO models to specify 3D gestural interaction with objects ?
8. Do you think GISMO is adequate for expressing gestural interaction with objects ?

A.2 List of questions asked after designing the virtual book interaction model

9. How would you rate your global experience with GISMO to model gestural interaction ?
10. Assuming that GISMO is fully developed and part of an integrated development environment and you are developing multimedia application, would you use GISMO if you would have to integrate gestural interaction in your applications ?
11. Do you think GISMO is adequate for specifying gestural interaction with objects ?
12. Do you think GISMO is easy to use ?
13. According to you, how could GISMO be improved ? (open answer)
14. Do you think the visual syntax is adequate ?
15. What are, according to you, the important elements that are not present and that you would like to find in such language ? (open answer)
16. Do you think changing the concrete syntax of GISMO is easily achievable ?
17. According to you, is the notion of states and gestures adequate ?
18. According to you, how easy is it to specify properties in GISMO ?

19. Do you think it is easy to integrate GISMO in software applications (in particular thanks to the automatically generated interface) ?
20. In what type of context do you think GISMO might be useful ? (Gaming, Domotics, Multimedia, Healthcare, Automotive, Other)

B Java code of the virtual book interaction model

The following Java code represents an implementation of the virtual book interaction model. The SwingStates API is used for implementing state charts in Java. We also assume that a Java version of GMOD is available and that high-level gestures are available as a Java API.

Transitions are triggered when high-level gestures are received by the statechart model. For example, in the *stored* state, going to *closed* state is possible when a *drag* event is received. Preconditions can be specified by defining a guard on the transition. The active context is an instance of the GMOD framework, on which a method invocation can be performed when an event occurs. This allows to ask for the parameters of the occurred gesture (distance, body limb used, direction).

```java
import fr.lri.swingstates.sm.State;
import fr.lri.swingstates.sm.StateMachine;
import fr.lri.swingstates.sm.Transition;
import fr.lri.swingstates.sm.transitions.Event;

public class SwingstatesGISMO {

    private static StateMachine sm;
    private static Context context = new Context();

    private static int pmin = 1;
    private static int pmax = 100;
    private static int page = 1;

    public static void main(String[] args) {
        sm = new StateMachine() {

            public State stored = new State() {

                Transition retrieve = new Event("drag", ">> closed") {
                    public boolean guard() {
                        return context.getDist() > 30.
                                && context.getHand().equals("left")
                                && context.getDirection().equals("BW");
                    }
                };
            };

            public State closed = new State() {

                Transition store = new Event("drag", ">> stored") {
                    public boolean guard() {
                        return context.getDist() > 30
                                && context.getHand().equals("left")
                                && context.getDirection().equals("FW");
                    }
                };

                Transition openb = new Event("zoom", ">> open") {
                    public boolean guard() {
```

```
                return context.getDist() > 30
                        && context.getDirection().equals("AW");
            }
            public void action() {
                page = 1;
            }
        };
    };

    public State open = new State() {

        Transition turnleft = new Event("move", ">> open") {
            public boolean guard() {
                return context.getDist() > 30
                        && context.getHand().equals("left")
                        && context.getDirection().equals("LF");
            }
            public void action() {
                page++;
                if (page > pmax) {
                    page = 100;
                }
            }
        };

        Transition turnright = new Event("move", ">> open") {
            public boolean guard() {
                return context.getDist() > 30
                        && context.getHand().equals("left")
                        && context.getDirection().equals("RG");
            }
            public void action() {
                page--;
                if (page < pmin) {
                    page = 1;
                }
            }
        };

        Transition closeb = new Event("zoom", ">> closed") {
            public boolean guard() {
                return context.getDist() > 30
                        && context.getDirection().equals("TW");
            }
        };
    };
  };
 }
}
```

References

[1] V. Amaral, A. Cicchetti, and R. Deshayes. A multiparadigm approach to integrate gestures and sound in the modeling framework. In *Proceedings of the 7th Workshop on Multi-Paradigm Modeling co-located with the 16th International Conference on Model Driven Engineering Languages and Systems, MPM@MoDELS 2013, Miami, Florida, September 30, 2013.*, pages 57–66, 2013.

[2] C. Appert and M. Beaudouin-Lafon. SwingStates: Adding state machines to Java and the Swing toolkit. *Software Practice and Experience*, 38(11):1149–1182, September 2008.

[3] W. Buxton. A three-state model of graphical input. In D. Diaper, D. J. Gilmore, G. Cockton, and B. Shackel, editors, *Human-Computer Interaction, INTERACT '90, Proceedings of the IFIP TC13 Third Interantional Conference on Human-Computer Interaction, Cambridge, UK, 27-31 August, 1990*, pages 449–456. North-Holland, 1990.

[4] R. Deshayes and T. Mens. Towards a domain-specific language for executable modelling and verification of gestural interaction. In *ACM SIGCHI Symposium on Engineering Interactive Computing*, 2015.

[5] R. Deshayes, T. Mens, and P. Palanque. A generic framework for executable gestural interaction models. In *Visual Languages / Human Centric Computing*, pages 35–38, 2013.

[6] R. Deshayes, B. Meyers, T. Mens, and H. Vangheluwe. Promobox in practice : A case study on the GISMO domain-specific modelling language. In *Int'l Workshop on Multi-Paradigm Modeling (MPM)*, pages 21–30, 2014.