# Using Formal Methods to Validate Research Hypotheses: the Duolingo Case Study

Antonio Cerone and Aiym Zhexenbayeva

Department of Computer Science, Nazarbayev University, Astana, Kazakhstan
{antonio.cerone, aiym.zhexenbayeva}@nu.edu.kz

**Abstract.** In this paper we present a methodology that combines formal methods and informal research methods to validate research hypotheses. We use the CSP (Communicating Sequential Processes) process algebra to model the system as well as user profiles, and PAT (Process Analysis Toolkit) to perform formal verification. We illustrate our methodology on Duolingo, a very popular application for language learning. Two kinds of data are considered: a log of the interaction of the user with the application and the assessment of the user's level of proficiency in the language to be learned (learner profile). The goal is to validate research hypotheses that relate the learner profile to the user behaviour during interaction (user profile). To this purpose, two CSP processes, one modelling the user profile that is associated by the considered research hypothesis to the learner profile and one modelling the interaction log are composed in parallel with the system model. Thus, for each user with the given learner profile and specific interaction log, the verification of the functional correctness of the overall system validates the correlation between user profile and learner profile.

**Keywords:** Formal Methods; CSP Process Algebra; Process Analysis Toolkit (PAT); Multimodal Interaction; Language Learning Application.

## 1 Introduction

Almost all people are nowadays routinely running heaps of applications on their mobile devices. There is a large variability both of users, e.g. in terms of age, education and cultural background, and applications, which cover entertainment, learning, personal monitoring, accounting, internet banking, booking and many other domains. This global variability makes it impossible to develop interactive systems appropriate to all users. It is then essential to understand the different ways users may potentially interact with the application and try to address them. However, in order to better adapt the application interface to the user, it is also needed to understand how the user's knowledge and activity within the domain for which the application is created drive a specific interacting approach.

In this paper, we consider a language-learning application, which uses two modalities to present exercises to the user, i.e. audio and printed text, and we observe that the combination of the two modalities within the same question

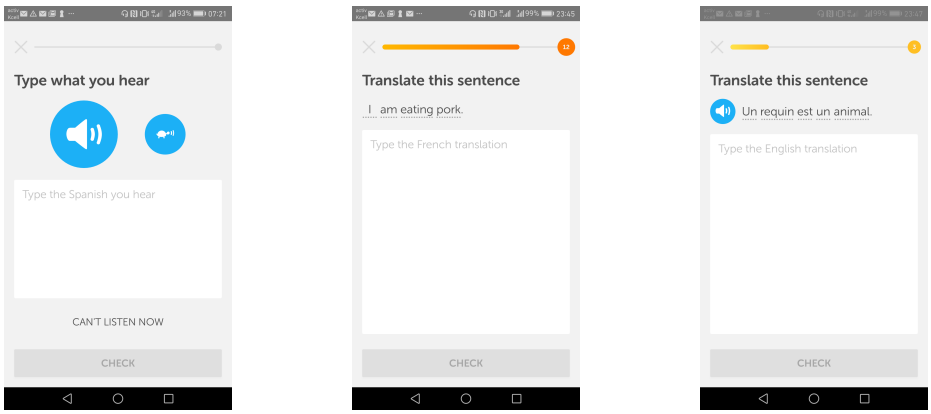
may induce some users to make errors. In order to understand what drives the observable user behaviour in interacting with the application, in the specific learning context of our example, we distinguish between the *user profile*, characterising the way the user interact with the application, and the *learner profile*, characterising the level of proficiency of the user in the foreign language.

We use formal methods, specifically the CSP process algebra [5], to model the application and the user profile and to formally represent the log of the interaction of the user with the application [4, 6]. The learner profile is instead defined using social science research methods: tests, questionnaires, interviews. Our approach aims to consider a hypothesis on the relation between given user profile and learner profile and validate it by carrying out formal verification on the model that comprises both that user profile and a formal representation of the interaction log of users with that learner profile. We use the model-checking capabilities of the Process Analysis Toolkit (PAT) [2] to perform formal verification.

## 2 The Problem: Duolingo Application Case Study

Duolingo [1] is the most popular language learning platform. It includes a website and mobile applications. It offers a large number of language courses for both English and non-English speakers.

A lesson is structured as a sequence of exercises, each featuring a different kind of question. After the user answers the question, the application provides an assessment as correct or wrong before proceeding to the next exercise or completing the lesson. In this paper we consider the three kinds of questions illustrated in Fig. 1:




**(a)** **(b)** **(c)**

**Fig. 1.** Duolingo screenshots.

**(a)** the user hears a sentence in the foreign language and has to type it;
**(b)** the user reads a sentence in the native language and has to translate it in writing to the foreign language;
**(c)** the user reads and hear a sentence in the foreign language and has to translate it in writing to the native language.

These three questions are representative of the three possible situations in which audio and visual presentation modalities are used separately and in combination.

We carried out some experiments using the Duolingo application and we realised that a common error consists in giving the answer in the wrong language. Typically, the user will tend to ignore the information on the type of exercise and focus instead on the question itself. Furthermore, since the question may be proposed using two modalities, audio and printed text, the user may focus on just one of such modalities. For example, when the question involves a translation to the native language, the sentence to translate is proposed in the foreign language using both audio and print modalities. However, the user may actually focus on just one modality. If the user, in general, tends to focus on the audio modality, several repetitions of this kind of exercise will create an automatism whereby the user always tends to translate an audio perception to the foreign language. Therefore, when the exercise requests to type what is heard, a user affected by such acquired automatism would instead translate to the native language, thus giving the wrong answer. We analyse this kind of error in Sect. 4 and 5.

## 3 CSP Model

In this section we use CSP to model the three kinds of questions illustrated in Fig. 1. In our abstract model, the only parameter used to discriminate between correct and wrong answer is the language in which the answer is given: native or foreign language. The model is presented in Fig. 2.

```
DuolingoExercise() = question -> ( typeWhatYouHear -> CheckTypeWhatYouHear() []
                                   translateToForeign -> CheckTranslationToForeign() []
                                   translateToNative -> CheckTranslationToNative() );

CheckTypeWhatYouHear() = foreignLang -> correct -> DuolingoExercise() []
                         nativeLang -> wrong -> DuolingoExercise();
CheckTranslationToForeign() = foreignLang -> correct -> DuolingoExercise() []
                              nativeLang -> wrong -> DuolingoExercise();
CheckTranslationToNative() = nativeLang -> correct -> DuolingoExercise() []
                             foreignLang -> wrong -> DuolingoExercise();
```

**Fig. 2.** System Model: Exercises and Assessment

The `DuolingoExercise` process presents the three possible kinds of questions: `typeWhatYouHear`, `translateToForeign` and `translateToNative`. A request to type what is heard in the foreign language (`typeWhatYouHear`) is checked by the `CheckTypeWhatYouHear` process, which returns `correct` if the

```
DuolingoAudio() = question -> ( typeWhatYouHear -> audio -> DuolingoAudio() []
                                translateToNative -> audio -> DuolingoAudio() []
                                translateToForeign -> noAudio -> DuolingoAudio() );

DuolingoPrint() = question -> ( typeWhatYouHear -> noPrint -> DuolingoPrint() []
                                translateToNative -> printForeign -> DuolingoPrint() []
                                translateToForeign -> printNative -> DuolingoPrint() );

SessionSystem() = DuolingoExercise() || DuolingoAudio() || DuolingoPrint();
```

**Fig. 3.** System Model: Modalities.

```
UserData() = question -> typeWhatYouHear -> foreignLang ->
             question -> translateToForeign -> foreignLang ->
             question -> translateToNative ->  nativeLang ->
             question -> typeWhatYouHear -> nativeLang -> Stop();

SessionUserData() = SessionSystem() || UserData();
```

**Fig. 4.** Example of User Data.

answer is given in the foreign language (`foreignLang`) and `wrong` if it is given in the native language (`nativeLang`). The other two kinds of questions are checked analogously.

Processes `DuolingoAudio` and `DuolingoPrint` in Fig. 3 model the two output modalities used by Duolingo. The requests to translate to the native language (`translateToNative`) are presented using both audio and printed text, whereas the other two requests are presented using just one modality, printed text for the translation to foreign language (`translateToForeign`) and audio for request to type what is heard in the foreign language (`typeWhatYouHear`).

The overall system is given by process `SessionSystem`, which is the parallel composition of the three components illustrated above.

Fig. 4 shows an example of data, consisting of a sequence of four questions proposed by Duolingo and the corresponding answers given by the user. We can note that, in the last question, the user gives the wrong answer by using the native language instead of the foreign language, that is, by translating rather than just typing what is heard.

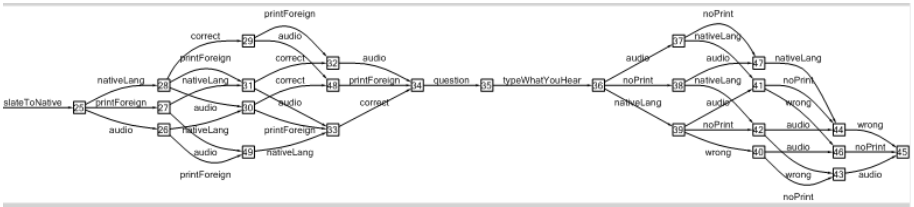



**Fig. 5.** Simulation.

We may compose the overall system `SessionSystem` with this specific dataset. Fig. 5 shows the fragment of graph generated by PAT for the part of behaviour of process `SessionUserData` corresponding to the last two questions. Note that for each question the user behaviour starts with an external choice among perception of audio, perception of printing text and user's decision to answer in native or foreign language. In fact, nothing prevents the user from deciding to answer in a language independently of the actual request by the application.

## 4 Formal Verification

In this section we present how to verify the functional correctness of the model defined in Sect. 3, how to constrain the model with specific user profiles and how to verify whether such user profiles are prone to incur in the error considered at the end of Sect. 2. Functional correctness is characterised by the ability of the system to provide the user with the proper assessment of the answer as correct or wrong for each question. We may say that "*always*, if there is a question, then there will *not* be any *further* question *until* the user's answer is assessed as correct *or* wrong". This statement may be refined towards a low-level temporal logical formula as:

> "*always*, if there is a `question`, then, starting from the *next state* of the system, there will *not* be any `question` *until* the user's answer is assessed as `correct` *or* `wrong`"

The temporal logic counterpart of this statement is the formula of the first assertion in Fig. 6. Using PAT we can see that this first assertion is verified as valid. The second assertion, which states that the user gives a correct answer to each question, is, instead, verified as not valid. This is obviously due to the fact that, correctly, our model leaves the option that the user may give wrong answers open. We can say that this second assertion formalises a usability property, since it states that the user will not be induced by the system to provide a wrong answer.

In order to analyse the error illustrated at the end of Sect. 2, we consider two profiles: a user who always focuses on the print modality and a user who always focuses on the audio modality. These two kinds of users, after repeatedly using the application, will be driven towards two different forms of automatism. Fig. 7 shows the models for such profiles in terms of the acquired automatism. A user who focuses on the print modality (process `UserFocusPrint`) realises that:

- if the question is not printed, then the answer has to be in the foreign language;
- if the question is printed in the native language, then the answer has to be in the foreign language;
- if the question is printed in the foreign language, then the answer has to be in the native language.

A user who focuses on the audio modality (process `UserFocusAudio`) realises that:

```
#assert SessionSystem() |= [] ( question -> X (! question U ( correct || wrong)) );
#assert SessionSystem() |= [] ( question -> (! wrong U (correct)) );
```

**Fig. 6.** Assertions for Functional and Usability Properties.

```
UserFocusPrint() = noPrint -> foreignLang -> UserFocusPrint() []
                   printNative -> foreignLang -> UserFocusPrint() []
                   printForeign -> nativeLang -> UserFocusPrint();

UserFocusAudio() = audio -> ( foreignLang -> UserFocusAudio() []
                              nativeLang -> UserFocusAudio() ) []
                   noAudio -> foreignLang -> UserFocusAudio();

SessionFocusPrint() = SessionSystem || UserFocusPrint();
SessionFocusAudio() = SessionSystem || UserFocusAudio();

#assert SessionFocusPrint() |= [] ( question -> X (! question U correct) );
#assert SessionFocusAudio() |= [] ( question -> X (! question U correct) );

#assert SessionUserData() |= [] ( question -> X (! question U ( correct || wrong)) );
#assert SessionUserData() |= [] ( question -> (! wrong U (correct)) );
```

**Fig. 7.** User Profile Model and Analysis.

– if the question is not heard, then the answer has to be in the foreign language.

Therefore the audio modality is less informative than the print modality and gives space to two possible, conflicting forms of automatism. As we have discussed in Sect. 2, the user may interpret the audio either as a request to answer in the foreign language or as request to answer in the native language. The usability property in the first two assertion in Fig. 7 is verified by PAT as valid on process `SessionFocusPrint` (first assertion) and not valid on process `SessionFocusAudio` (second assertion). This is consistent with the fact that the automatism developed by the user who focuses on the print modality always leads to the correct answer, but this is not the case for the user who focuses on the audio modality.

Finally, we may also verify properties of the system behaviour on a specific data set. For example, considering the last two assertion in Fig. 7 with the dataset `UserData` in Fig. 4, which is consistent with focusing on the audio modality, as component of process `SessionUserData`, PAT verifies the first assertion (functional property) as valid and the second assertion (usability property) as not valid. If we remove the last question from `UserData`, which is the one causing the error, then the usability property is verified as valid, since the data is now consistent with focusing on the print modality too.

## 5 Hypothesis Formulation and Validation

We formulate two hypotheses to relate a user profile, i.e. which modality the user focus on, to a learner profile, i.e. which level of proficiency the user has in the foreign language.

**Hypothesis** [**H1**] *A learner at a beginner level in the foreign language always focuses on the print modality.*

**Hypothesis** [**H2**] *A learner at an advanced level in the foreign language always focuses on the audio modality.*

These two hypotheses are suggested by the observation that beginners have difficulty in listening comprehension and need the support of a written text, whereas advanced learners are able to quickly go through the exercises reacting immediately to the audio without reading the written text.

In order to validate these hypotheses, an extensive user experience evaluation should be conducted at the following two levels:

1. the creation of a log of the interaction of the user with the application, through either natural observation or by using an instrumented version of the application;
2. the assessment of the user's level of proficiency in the foreign language (learner profile), through either a language test or a questionnaire or interviews.

Then, for each subject user, the log is converted into a `UserData` process to be combined with the `UserFocusPrint` or `UserFocusAudio` process depending on whether the user is assessed at the beginner or advanced level of proficiency, respectively.

Formal verification is finally carried out as shown in Fig. 8. The two processes, `DataModelFocusPrint` and `DataModelFocusAudio`, combine the data collection at item 1 above, represented by process `SessionUserData`, with the user's assessment at item 2 above, which is associated by our research hypotheses to process `SessionFocusPrint`, if the user is assessed as a beginner ([**H1**]) or process `SessionFocusAudio`, if the user is assessed as an advanced learner ([**H2**]).

The assertion corresponding to the user profile that one of the research hypotheses associates with the assessed learner profile of the considered user is valid when the behaviour of process `SessionUserData` is consistent with the process that models the user profile, i.e. it does not invalidate the functional correctness. In fact, a mismatch between the considered user profile and the real user data would result in a conflict in some answer assessment as correct or wrong, with a resultant deadlock after the occurrence of `question` but before either `correct` or `wrong` may occur, thus invalidating the temporal logic formula for functional correctness. This is what happens if we verify the first assertion in Fig. 8 on the user data given in Fig. 4, due to the mismatch between a user whose real data corresponds to a focus on the audio modality and a user profile model of

```
DataModelFocusPrint() = SessionUserData() || SessionFocusPrint();
DataModelFocusAudio() = SessionUserData() || SessionFocusAudio();

#assert DataModelFocusPrint() |= [] ( question -> X (! question U ( correct || wrong)) );
#assert DataModelFocusAudio() |= [] ( question -> X (! question U ( correct || wrong)) );
```

**Fig. 8.** Formal Verification for Hypothesis Validation.

a focus on the print modality. Therefore, a research hypothesis is satisfied by a specific user when the assertion on functional correctness is valid. Finally, we can conclude that a research hypothesis is validated when it is satisfied by a statistically significant number of users with the appropriate learner profile.

## 6 Conclusion and Future Work

The analysis carried out on the Duolingo case study shows that multimodal interaction is not always effective and it is essential to take the user's profile into account while choosing whether and how to combine modalities. Furthermore, if our research question is validated, then we may claim that although the Duolingo application is appropriate for learners at the beginner level, in its current state it is not equally effective for learners at the advanced level. In this case, a possible improvement could be the introduction of a learner level, either explicitly set by the user or inferred by the the system in some intelligent way. The learner level would then drive the choice of modalities to use for question presentation: multimodality audio and print for a beginner learner and unimodality, either audio or print, for an advanced user.

There are three directions for our future work. First, we would like to validate our research hypotheses for the Duolingo case study on real data as discussed in Sect. 5. Second, we plan to apply our methodology to further, more challenging case studies. In fact, the case study and abstraction level considered in this paper result in very straightforward user profiles and system model, with properties easily observable on the data model. The purpose of our choice was to test the feasibility of our methodology and easily illustrate it. We now intend to combine this work with our work on cognitive errors [3] and consider system models in which human errors are not directly observable on the data model, but emerge from the interaction. Finally, we would like to extend our methodology by developing methods for synthesising users profiles from the data. In order to achieve such a challenging goal we may need to consider other formal methods instead of process algebras, and integrate some form of data mining or process mining within our approach.

## References

1. Duolingo. https://www.duolingo.com.
2. PAT: Process Analysis Toolkit. pat.comp.nus.edu.sg.
3. A. Cerone. Towards a cognitive architecture for the formal analysis of human behaviour and learning. In *STAF 2018 Workshops (FMIS)*, Lecture Notes in Computer Science. Springer, to appear, 2018.
4. A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human Computer Interaction*. Prentice Hall, 2004.
5. C. A. R. Hoare. *Communication Sequential Processes*. Prentice Hall, 2004.
6. B. van Schooten, O. Donk, and J. Zwiers. Modelling interaction in virtual environment using process algebras. In *TWLT15: Interaction in Virtual Worlds*, pages 195–212, 1999.