Consistently Formalizing a Business Process and its Properties for Verification: A Case Study

Michael Rathmair, Ralph Hoch, Hermann Kaindl, Roman Popp

Consistently Formalizing a Business Process and its Properties for Verification: A Case Study

Michael Rathmair, Ralph Hoch, Hermann Kaindl, and Roman Popp

TU Wien, Institute of Computer Technology
Vienna, Austria
{rathmair,hoch,kaindl,popp}@ict.tuwien.ac.at

Abstract. Formal verification of business process models can be done through *model checking* (also known as *property checking*), where a model checker tool may automatically find violations of properties in a process model. This approach obviously has formal representations as a prerequisite. However, a key challenge for applying this approach in practice is to consistently formalize the process and its properties, which clearly cannot be done automatically. We studied this challenge in a case study of formally verifying an informally given business process against a guideline written like a legal text. Major lessons learned from this case study are that formalizing is key to success and that in its course a semi-formal representation of properties is useful. In the course of such a step-wise and incremental formalization, problems with the given process model have been found already, apart from those found with a model checker tool that used the formal property specification. In total, our approach revealed five problems not found by the official review. In summary, this paper investigates in a case study consistently formalizing a business process and its properties for verification through model checking.

1 Introduction

Hardware and software should be free of errors, and the same applies to business processes. Usual quality assurance techniques for hardware and software in practice are reviews and tests. These have the purpose of finding errors but they can, in general, not show that there are no errors. Research both related to hardware and software investigates *model checking* for formal verification, which can, in principle, show that there are no errors with regard to certain properties. This requires both a formally specified behavioral model and formulas specifying properties to verify them against. Neither of them are usually available in practice, unfortunately.

Roughly speaking, a formal representation is one that allows (automatic) reasoning purely based on its form, which has defined semantics. In particular, illustrative diagrams or natural language are *not* formal representations. That is why formalization is necessary when something is given informally, e.g., in such diagrams or in natural language. Business process models typically are behavioral models, but usually not (really) formally defined in practice. So, their formalization is important but even more so the consistent formalization of properties to check them against. These properties are sometimes hard to get in practice and at best, informally described. The properties

typically refer to the behavioral models, but this means some coupling. And if the person who formalizes properties has the behavioral model available, then there will be some influence on the properties. This is reminiscent of someone writing test cases for his own software. So, we argue for more or less independent formalization of a behavioral model and of the properties to check it against. Unfortunately, this may lead to inconsistent formalizations that do not fit together for the purpose of model checking. Therefore, consistent formalization of behavioral model and properties is a challenge, which we address in this paper.

We investigated this challenge in the context of a case study where it became apparent. The task was verifying a high-level business process of our university against a corresponding guideline, both of which were given informally. More precisely, this is a real process enacted on a regular basis (for searching and appointing a full professor). For the case study, a process diagram in informal notation was used, as given in the course of preparations for an official Quality Audit. The guideline is an official document of our university, derived from the Austrian law for universities. In particular, its writing style is like that of this legal text, i.e., a special kind of text in natural language.

So, there was informal input, but formal representations of both process and properties are needed for formal verification using model checking. Of course, there is neither an automated transformation nor a defined sequence of steps available for such a task of formalization, and we did not attempt to define something like that, either. Still, we suggest, based on the case study, to use a semi-formal representation in the course of the formalization. Its usefulness is a major lesson learned, generalized from the case at hand.

The remainder of this paper is organized in the following manner. First, we present some background material on model checking in order to make this paper self-contained. Then we discuss related work. The core part presents a case study of model checking an informally given high-level process against a guideline based on legal text, where consistent formalization was particularly studied, and concludes with lessons learned from it. After that, we discuss threats to validity. Finally, we derive general conclusions on consistently formalizing properties used for model checking business processes.

2 Background on Model Checking

Model checking (or property checking) is a formal verification technique based on models of system behavior and properties, specified unambiguously in formal languages (see, e.g., [1]). The behavioral model of the system under verification is often specified using a Finite State Machine (FSM), in our case using synchronized FSMs. Their expressiveness is sufficient for our case, but Petri nets, e.g., could be used as well, if needed (depending on the tool used). The properties to be checked on the behavioral model are formulated in a specific property specification language, usually based on a temporal language . Several tools (such as SPIN [2] or NuSMV [3]) exist for performing these checks by systematically exploring the state-space of the system. When such a tool finds a property violation, it reports it in the form of a counterexample.

In this work, we make use of *Linear Temporal Logic*, or *Linear-time Temporal Logic*, (LTL) and *Computational Tree Logic* (CTL) for property specification. More

precisely, we use PLTL (LTL with past). Since a rough understanding of some of their operators is needed for understanding our formalization approach, let us briefly sketch these here. PLTL provides expressions of relations between states (path formulas) using operators referring to behavior over time. In PLTL, the set of traditional propositional logic operators is extended by time operators such as:

- G (Globally): an expression p is true at time t if p is true at all times $t' \geq t$.
- F (Future): an expression p is true at time t if p is true at some time $t' \geq t$.
- O (Once): an expression p is true at time t if p is true at some previous time $t' \leq t$.

CTL features the specification of branching time properties. While PLTL allows the specification of properties to hold for all computation paths related to a given point in time, CTL provides operators for specifying whether there exists (eventually) a computation path where a specific state property holds. In this work, we use the following CTL operators:

- EF (Eventually in the Future): an expression p is true in the initial state s_0 and there exists a state sequence $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n$ such that p is true in s_n.
- AG (Always Globally): an expression p is true in the initial state s_0 and in each state of all transitions $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots \rightarrow s_n$.

3 Related Work

Previous related work made it absolutely clear that some representation with defined semantics is a prerequisite for formal verification, also of business processes. Given such a representation, checking correctness properties inherent in the business process itself is possible. Since we rather focus on formalizing properties given in addition to a business process, we cite only a few references here. Wynn et al. [4] verify business processes against four defined properties (soundness, weak soundness, irreducible cancellation regions and immutable OR-joins). Sbai et al. [5] show how a model checker can be used to identify problems with a specification of a business process to be automated as a workflow, and how a verification of certain correctness properties can be accomplished. Kherbouche et al. [6] propose an approach for using model checking as a mechanism to detect errors such as deadlocks or lifelocks.

Some previous work addressed the question of what to verify a business process model against, to determine possible violations of certain properties given in addition to the process model itself. Fisteus et al. [7] propose a framework for integrating BPEL4WS and the SPIN and SMV verification tools. This framework can verify a process specification against properties such as invariants and goals through model checking. Armando et al. [8] show how model checking can be used for automatic analysis of security-sensitive business processes. They propose a system that allows the separate specification of the business process workflow and of corresponding security requirements. In more recent work [9], they show how model checking can be specifically used to check authorization requirements that are implemented in parts of business processes. Barros et al. [10] propose to check business processes against execution rules incorporated in workflows with model checking techniques.

Mrasek et al. [11] point out that formalizing properties in CTL is a difficult task and strive for making it easier through so-called patterns based on textual fragments in natural language. This approach can work in a given context for entering properties, and it helped in a case study. In general, however, the interpretation of these textual patterns is subtle and error-prone. So, they have to be prepared specifically for a given problem by CTL specialists, anyway. In particular, for our case study with given legal text, such an approach would most likely require a variety of different patterns and still be hard to validate.

The focus of our work as presented in this paper is, however, consistently formalizing the business process and its properties as required for automatic verification through model checking. Still, no previous work in the context of model checking of business process models addressed it to our best knowledge, including model-based business process compliance-checking approaches [12]. Apart from [11], which addresses formalizing properties (but not formalizing the process), all the publications on model checking of business processes already assume the availability of formal representations.

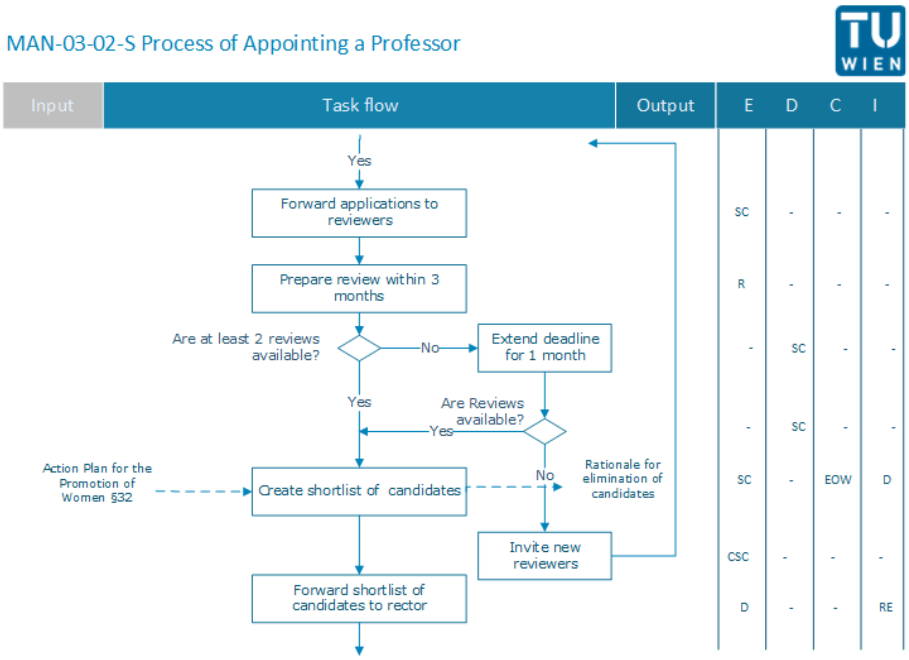
4 A Case Study of Model Checking a High-level Process Against a Guideline

We performed a case study, where we verified a high-level business process of our university against a corresponding guideline, both of which were given informally. More precisely, this is a real process enacted on a regular basis (for searching and appointing a full professor), but its 'as-is' process *diagram* has been yet under development at this time (in the course of preparations for an official Quality Audit). We used a version of this diagram that was under official review at about the same time as the case study, but in order to keep pace with the tight schedule of the overall endeavor, we only dealt with the core part of this process where the search committee is active. Figure 1 shows a selected part of this core part to be used below for illustration purposes, in the informal notation officially used. (Note, that the arrow from the task "Invite new reviewers" leads outside of this selected part.) The guideline is an official document of our university, and its text (in German) can be found at `http://www.tuwien.ac.at/dle/universitaetskanzlei/satzung/berufungsverfahren/`. This guideline is derived from the Austrian law for universities. In particular, its writing style is like the one of this legal text.

4.1 Stakeholders

The stakeholders involved in this case study, directly or indirectly, are the following:

– Central group responsible
  A dedicated group directly assigned to the rectorate was responsible for creating business models of several high-level processes of our university, and for their review.



E = Executes; D = Decides; C = Contributes; I = Informed.
SC = Search Committee; R = Reviewer; EOW = Equal opportunities working party; D = Dean; CSC = Chair Person of the Search Committee; RE = Rector.

Fig. 1: Diagram of a Part of the Process for Appointing a Professor

– Working groups
  In order to acquire knowledge on these processes, several dedicated working groups were assembled, whose output was fed into the process models created by the central group responsible for that.
– Reviewers of process models
  In order to get as much feedback as possible on these models, every employee of our university has been invited to participate in the review (via email).
– Case study team
  The people having performed this case study are actually the same as the authors of this paper. Two of them had enacted this very process in key roles in 2013.
– Verification engineer
  The first author of this paper was the verification engineer in this team and brought in know-how and experience from applying such techniques in hardware design, more precisely circuit verification, see, e.g., [13].

4.2 Formalization of Guideline and Process Model

Initially, the verification engineer only worked on formalizing the guideline, as he did not know the process at all. Therefore, he could not formalize the process model yet,

and also not completely formalize the properties to check it against, since he did not know the states of the process model. So, he created a *semi-formal* representation of properties according to the guideline first, where he used PLTL/CTL operators already, but still text fragments from the guideline to indicate, e.g., sequences of tasks. This property representation was subject to an informal manual inspection by the complete case study team. Only after that, the verification engineer was given the part of the process model, which he formalized as an FSM. Once having the FSM available, he formalized the properties based on the previously prepared *semi-formal* representations. Finally, he applied the model checking tool for checking whether the process model is inconsistent with the given guideline.

For illustrating the creation of a semi-formal representation, let us start with an example excerpt from the guideline. §7(1) of the guideline is given in the second column of Table 1: "The chairperson of the search committee forwards …". First, the verification engineer identified all actors and artefacts mentioned in the partial sentence and treated them as individual objects, e.g., CSC and list (of candidates). After that, he identified actions and mapped them to expressions including temporal operators, e.g., F as specifications of allowed sequences. Since this example contains two different actions with different actors, the full paragraph is actually mapped to two properties. The second of these properties in its semi-formal representation, i.e., "G (list = TRUE $\rightarrow$ F Dean.state = forward candidate list to rector)", expresses that globally (PLTL operator G), if the list exists, then the Dean must reach a future state (PLTL operator F), where the list of candidates is forwarded to the rector.

Especially with respect to 'time', it is interesting to give an example of what we did *not* formalize from the given guideline, even though temporal logics are employed. The guideline says, for instance, "reviews should be prepared within 3 months" (as translated from the German text). The verification engineer did not include this statement into the list of properties (not even their semi-formal representation), since he had his focus on *sequences in time*, although the so-called X operator of LTL could be used in an attempt to model this statement. In hindsight, we briefly discussed this possibility, where time units would be defined, e.g., a day. Based on that, the checker tool could simulate time slots through loops for a defined number of time units corresponding to 3 months. Obviously, there would a minor intricacy involved, since not every month has the same number of days, but this could be approximated. Another, more serious issue with this approach would be that a day would be the minimum duration of each task. All of these intricacies of this thought experiment are, however, beyond the scope of the given guideline text and the law it is based upon.

A related example of what we clearly did not formalize about 'time' is the phrase "as soon as possible" (in the original German text the word "ehestmöglich") in §7(1), see Table 1 in the last row. Without modeling time units, this cannot even be approximated.

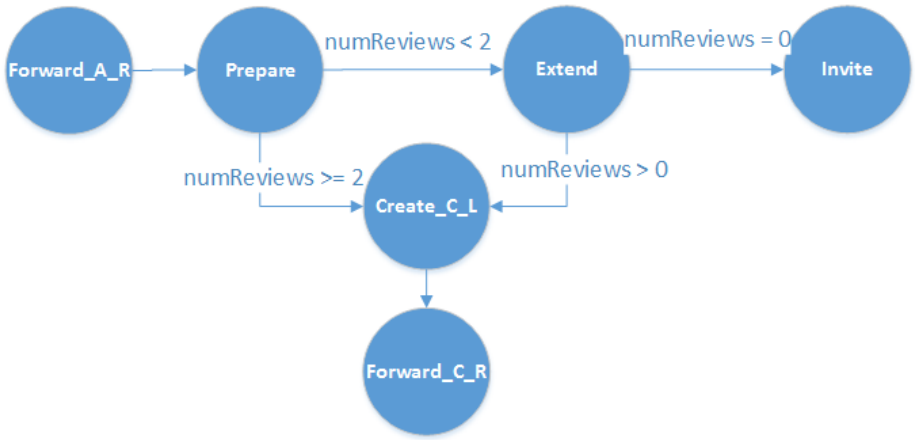
The effort for the manual translation of the given part of the guideline to a set of semi-formally represented properties was approximately six hours. It included eleven paragraphs of the guideline, resulting in 25 semi-formally represented properties, and a meeting of the team for their informal inspection.

The informal inspection of the semi-formal property formulations was an essential part of the full verification process. The translation of a natural language text fragment

Table 1: A Selection of Properties as Derived from the Guideline, where Violated Properties are Highlighted through Various Shades of Gray as Background

§	Guideline Text	Semi-formal Property Representation	Property Formula
§6(2)	If according to "§3 Absatz 1" of this guideline more than 2 reviews have been requested, and are 2 reviews available after 3 months, the search committee makes its decision based on these available reviews, otherwise the search committee may extend the deadline for one month ; after this deadline, the search committee makes its decision based on the available reviews	G (SC has received 2 or more reviews $\rightarrow$ F SC.state = create list with candidates)	G ((inst_process.state = Prepare & inst_review_list.i_numReviews $= 2$) $\rightarrow$ F inst_process.state=Create_C_L)
		AG (SC has received less than 2 reviews $\rightarrow$ (!EF SC.state = Extend deadline for 1 month & EF SC.state = Extend deadline for 1 month)	AG (((inst_process.state = Prepare) & (inst_review_list.i_numReviews $<$ 2)) $\rightarrow$ (!(EF inst_process.state = Extend) & (EF inst_process.state = Extend))
		G (SC.state = Extend $\rightarrow$ F SC.state = create list with candidates)	G (inst_process.state = Extend $\rightarrow$ F inst_process.state=Create_C_L)
§7(1)	The chairperson of the search committee forwards this list as soon as possible to the responsible dean, who forwards it to the rector.	G (list = TRUE $\rightarrow$ F CSC.state = forward candidate list to dean)	No refined formalization possible, since no such state exists in the FSM.
		G (list = TRUE $\rightarrow$ F Dean.state = forward candidate list to rector)	G (inst_process.state = Forward_C_R $\rightarrow$ inst_process.E = Dean)

inst_process = Process of appointing a professor;
Extend = Extend deadline for 1 month;
Create_C_L = Create shortlist of candidates;
Prepare = Prepare review within 3 months;
Forward_C_R = Forward shortlist of candidates to rector.



Extend = Extend deadline for 1 month; numReviews = signal number of reviews; Create_C_L = Create shortlist of candidates; Prepare = Prepare review within 3 months; Forward_A_R = Forward Applications to Reviewers; Forward_C_R = Forward shortlist of candidates to rector; Invite = Invite new reviewers.

Fig. 2: FSM of the Part of the Process Model shown in Figure 1

as given by the guideline is an error-prone process. One or more formulated properties have to cover the textually given facts and characteristics of the described process "Appointing a Professor". At the review meeting, all 25 properties were discussed and analyzed whether they are not in conflict, covering the guideline and adequately representing it.

Based on the results of this review, the verification engineer made a few changes to the semi-formal representation of the properties. After that, he was given the process representation in the form of the excerpt shown in Figure 1, essentially an annotated flow diagram. He constructed an FSM for the control flow, which is shown in Figure 2 for the same excerpt. In essence, he mapped each chart element to an FSM state each. State transitions were derived from the task flow of the given process model. Since the arrow from the state "Invite new reviewers" in Figure 1 is outside of the part selected for presentation here, it is not included in the FSM.

Constructing this FSM may even look straight-forward, especially for the excerpt used in this paper. However, the notation of diagrams like those shown in Figure 1 is not formally specified. In fact, forks of lines can either mean procedural or concurrent flows, as we found in such process representations. So, there is essential ambiguity also in such process diagrams, which make formalization hard in general.

Data objects were extracted from the given process model and may influence control flow decisions: reviewer list, review list, application list, and list of candidates. As illustrated in Figure 2, i_numReviews is an internal variable of the data object review list. The value of this variable (either > 0 or $= 0$) directly influences the state transition in the control flow FSM.

Roles of this process as given through the columns with the header "E / D / C / I" in the process diagram of Figure 1 are modeled through a variable each, all of them of type enumeration, with possible values SC, CSC, etc. The values assigned depend on the given FSM state, e.g., for the state *Create_C_L* of this FSM corresponding to the task "Create shortlist of candidates", variables to be used by the model checking tool are assigned as follows: E := SC, C := EOW and I := D.

Once the FSM is defined, based on the given process diagram, it is useful to reflect again on the representation of 'time' apart from sequences. There is a task with the text "Prepare review within 3 months" in the process diagram in Figure 1. In the FSM, it is simply a state with a corresponding identifier. So, the semantics of this text is obviously not represented. The verification engineer saw the correspondence of this text with the corresponding text of the guideline (as discussed above) in passing, but there was no formal verification based on a temporal logic.

For the creation of the FSM and the definition of related variables as derived from (the selected part of) the officially given process diagram, a time effort of approximately three hours was used.

According to the given FSM, the verification engineer manually reformulated the semi-formally represented property statements to corresponding PLTL or CTL formulas, respectively. Informal parts of the semi-formally represented property had to be replaced by expressions referencing states of the FSM and its related variables. In the example used above, the formula "G (inst_process.state = Forward_C_R $\rightarrow$ inst_process.E = Dean)" refers to the process state Forward_C_R. While the list object is not used here, the state Create_C_L before Forward_C_R (in the FSM) creates it.

If a semi-formal representation of a property could not be translated to an adapted PLTL or CTL formula, two cases were to be distinguished:

1. The granularity of the given model was partially not compatible with the level of detail specified in the guideline. Hence, if the verification engineer was unable to redefine some semi-formally represented property, this is not necessarily a violation of the given guideline.
2. An error in the model was identified. A subsequent detailed manual inspection of the model uncovered errors like missing or improper states, undefined variables, etc.

An example of a missing state is the first semi-formally represented property of §7(1). It cannot be translated into a refined formula because there is no state in the given model denoting that "The chairperson of the search committee forwards this list as soon as possible to the responsible dean", i.e., the sentence highlighted in light-gray of the guideline in Table 1 and the corresponding semi-formal property representation. More precisely, there is also a confusion in the process model about the actors SC and CSC involved here, which means a second violation of this property. Yet another task was identified to be missing (with respect to preparing and submitting the final report, which were mixed up in the process model), similarly to the one indicated above. So, a total of three violations were found by the verification engineer already in the course of modeling.

After this final formalization effort of approximately two hours, both the model in the form of an FSM and the set of formalized properties were defined and ready for model checking with the tool.

4.3 Tool-supported Model Checking and its Results

For the tool-supported model checking, the formalized process model (in the form of the FSM and its associated variables) and the set of refined properties from the guideline (in the form listed in the fourth column of Table 1) were input to the model checker tool NuSMV. For any property violation found by the model checker tool, it returns a *counterexample* listing. This is presented as a possible execution sequence violating a specific property formulation. These results had to be manually analyzed by the verification engineer to locate the violations in detail and, finally, to interpret them in terms of the given process model and the guideline it has been verified against.

Table 1 contains examples of such violations found by the NuSMV tool, indicated in dark-gray and mid-gray, respectively. Let us explain the violation shown in mid-gray first. Listing 1 shows a selected part of the counterexample report for this violated property formula. Such listings refer to states, but these are different from the states of the FSM. In fact, NuSMV enumerates its *execution* states, whose sequence forms a trace. In this example, State 1.1 denotes that the variable i_numReviews, which is a local variable of the data object review_list, has the value 0. The control flow FSM reaches its FSM state Prepare at (execution) State 1.13. At State 1.14, the control flow FSM reaches its state Extend because the condition "Are at least 2 reviews available" is not fulfilled. The violation of this property is indicated by the following unspecified transition from the FSM state Extend (State 1.14) to Invite (State 1.16). This FSM state sequence is in conflict with the partial sentence "after this deadline, the search committee makes its decision based on the available reviews" (textual part highlighted in mid-gray of column one). In fact, the guideline does not define what to do if i_numReviews $= 0$. As a consequence, the model checking tool automatically indicates that the given formal property is not satisfied on the formalized input model, and hence the guideline is not consistent with the process model.

```
-- specification  G (inst_process.state = Extend -> F
    inst_process.state = Create_C_L)  is false
-- as demonstrated by the following execution sequence
    Trace Type: Counterexample
-> State: 1.1 <-
...
inst_review_list.i_numReviews = 0
...
-> State: 1.13 <-
inst_process.state = Prepare
-> State: 1.14 <-
inst_process.state = Extend
-> State: 1.16 <-
inst_process.state = Invite
...
```

Listing 1: Counterexample Tool Output Reporting a Property Violation

For the second property violation found by the tool, highlighted in dark-gray in Table 1, the same counterexample as shown in Listing 1 is returned (possibly with different execution states, but this does not matter). In fact, the value 0 is the cause of the contradictions of both properties. In addition, the property highlighted in dark-gray is also violated if the variable i_numReviews is assigned to 1. This has actually been checked with the tool by the verification engineer by forcing it to evaluate an example with the value 1, where the tool tells that this is a counterexample as well.

This second property violation is especially interesting, since it expresses that shifting of the deadline for the submission of Reviews is optional and not mandatory if i_numReviews < 2. This is formalized as a CTL formula, which enables the combination of path and state operators. Since the property states that the task "Extend the deadline for 1 month" is optional, it has to be checked whether both paths, one including the task and one not, are reachable. In terms of formalization, simply using the EF operator in one direction is not sufficient, therefore, since it defines that a path *exists*. This would also include the case of a mandatory extension as given in the process model. So, it is necessary to have a conjunction with the same part of the formula negated, see Table 1.

For the dedicated final adaptation and debugging of the formalized model and properties, a final interpretation, and location of the two violations found by the tool, a time effort of approximately two hours was used.

4.4 Summary of Results and Lessons Learned

In total, five problems were revealed in the selected part of the process for appointing a professor. In fact, no problems at all were found in this part by the official review, for which all employees of the university had been invited.

Now let us briefly generalize from the case at hand and try to indicate lessons learned that may be useful for similar endeavors:

- *Possibly conflicting roles*
  Specifying the process model and the properties to be used for verifying it, should be done by different people. The process model should also not be known in advance by the one(s) formalizing the properties. Having it available, however, imposes the risk that knowing the process already may influence the verification engineer in the semantic interpretation of properties. We think that this is reminiscent of a test-case writer who knows the internal details of the program to be tested. In our case study, this helped the verification engineer to avoid related pitfalls.
- *Usefulness of a semi-formal representation of properties*
  Introducing a semi-formal representation of properties appears to be helpful, when the final formulas cannot be directly stated because the process model is not (yet) available to the verification engineer. In addition, the semi-formal representation was useful for the informal inspection, since not the whole team had to be familiar with the actual modeling language.
- *Finding violations in the course of formalization*
  While usually the emphasis is on finding violations through a model checker tool, already in the course of formalizing properties, certain violations can be revealed. In

our case study, even three out of five violations were found in the course of trying to translate the semi-formal representation of properties to formulas fitting the FSM of the process model.

– *Mismatch of the level of abstraction of the business process model and properties to be checked*
  Especially when the model and the properties are formalized separately, a mismatch of their respective levels of abstraction may occur. In our case study, several statements in the official guideline that the given process diagram was verified against were much more detailed than this diagram with its abstractions. Analyzing such cases helped to determine missing tasks in the process model.
– *Formalizing time*
  It may come as a surprise first that in spite of the use of temporal logics certain aspects of 'time' were not formally represented and, therefore, not verified by the model checker tool. One issue in this regard is, again, the level of abstraction to be used for consistent formalization of the process and its properties, another the expressiveness of the given formalisms.
– *Ambiguity of legal text*
  It does not come as a surprise, however, that natural language is, in principle, ambiguous. Strictly speaking, we had to deal with a kind of legal text in our case study, which is supposed to be less ambiguous. We believe that we made a reasonable formalization of the text "the search committee may extend the deadline for one month", as given in Table 1 in dark-gray, as discussed in relation to a property violation found by the checker tool. The text is, however, "otherwise the search committee may" ("andernfalls kann die Berufungskommission" in the original German text), which could even deserve a legal interpretation by an educated expert in the given context.

While some of these lessons learned have most likely been observed before and in other areas, we think that especially finding violations already in the course of formalization, as well as mismatch of the level of abstraction of the business process model and properties to be checked are new. Some of our lessons learned may even be more generally relevant for other formalization efforts, e.g., usefulness of a semi-formal representation. In fact, it was also observed in the context of requirements engineering [14].

5 Threats to Validity

Of course, there are threats to the validity, both external and internal. Regarding threats to the external validity, having performed just a single case study yet is obviously relevant. In addition, this was a relatively small case study. However, the third author is well informed about the spectrum of business processes subject to the official Quality Audit, and we can state that the chosen one is highly representative. So, for gaining first insights into issues and benefits of applying model checking to the verification of such high-level business process, this case study seems to have been appropriate, especially with regard to the formalization challenge. Still, more cases studies along these lines will be required, also in industry, to substantiate the results.

Regarding threats to the internal validity, let us consider the fact that the verification engineer (the first author) comes from the field of hardware design and verification. The work in this case study may have been easier to tackle with more background on business processes and their modeling. However, our experience shows that especially the core task of formalization was possible to be done well even without. The most important background was the know-how for model checking and its tool support as well as the prior experience with behavioral and temporal modeling, even gained in a completely different domain. In addition, a case study with inductively generalized lessons learned is clearly a weak method for gaining trustable results. However, it is a recognized method for empirical studies and for gaining experience, and we applied it according to the usual way of doing a case study according to the current state of the art. Still, for getting more reliable results for specific hypotheses (to be formulated based on our lessons learned), reproducible experiments will be necessary.

6 Conclusion

In this paper, we investigate formalizing as required for formal model checking. We present a case study of formalizing an excerpt of a high-level business process of our university and of properties derived from a guideline derived from an Austrian law. A fundamental issue involved was formalizing properties consistently with a process model (yet) unknown to the verification engineer. This has been addressed and solved by using a semi-formal intermediary specification. In this case study, problems in the process model were found already while formalizing, and later by the model checker tool. Overall, several problems were revealed by this approach to model checking that were uncovered in the official review.

In this course, we investigated a new verification option for business process owners whose processes and associated properties are given informally. There is quite some effort involved in such a formalization task, and a case study like ours helps to get an idea of the amount. It just depends on the criticality of the business process, whether the results are worth this effort. The process of our case study is one of those most critical for a university, and none of the problems found by our formalization effort and by the model checker has been found in a review by all members of the university, in principle. So, it is up to the people responsible for a given business process to judge whether this effort may pay off. Our case study shows the feasibility of such an approach and provides data on this trade-off.

Based on all that, we conclude that consistently formalizing is key for successful formal and automated verification of business processes. This has not been addressed yet in the literature. Still, much is left for future work, such as systematically extracting data objects and their life cycles from process models and additional sources, and machine support for incremental formalization (possibly building on [15, 16]).

Acknowledgment

Part of this research has been carried out in the ProREUSE project (No. 834167), funded by the Austrian FFG.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
2. SPIN: SPIN Verifying Multi-threaded Software with Spin. `http://spinroot.com/spin/whatispin.html` [Online; accessed 01-December-2014].
3. NuSMV: NuSMV: a new symbolic model checker. `http://nusmv.fbk.eu/` [Online; accessed 01-December-2014].
4. Wynn, M., Verbeek, H., van der Aalst, W., ter Hofstede, A., Edmond, D.: Business process verification – finally a reality! Business Process Management Journal **15**(1) (2009) 74–92
5. Sbai, Z., Missaoui, A., Barkaoui, K., Ben Ayed, R.: On the verification of business processes by model checking techniques. In: Software Technology and Engineering (ICSTE), 2010 2nd International Conference on. Volume 1. (Oct 2010) V1–97–V1–103
6. Kherbouche, O., Ahmad, A., Basson, H.: Using model checking to control the structural errors in bpmn models. In: Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on. (May 2013) 1–12
7. Fisteus, J.A., Fernández, L.S., Kloos, C.D.: Applying Model Checking to BPEL4WS Business Collaborations. In: Proceedings of the 2005 ACM Symposium on Applied Computing. SAC '05, New York, NY, USA, ACM (2005) 826–830
8. Armando, A., Ponta, S.: Model checking of security-sensitive business processes. In Degano, P., Guttman, J., eds.: Formal Aspects in Security and Trust. Volume 5983 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2010) 66–80
9. Armando, A., Ponta, S.E.: Model checking authorization requirements in business processes. Computers & Security **40**(0) (2014) 1 – 22
10. Barros, C., Song, M.: Automatized checking of business rules for activity execution sequence in workflows. Journal of Software **7**(2) (2012)
11. Mrasek, R., Mülle, J.A., Böhm, K., Becker, M., Allmann, C.: User-friendly property specification and process verification - a case study with vehicle-commissioning processes. In: Proceedings of BPM 2014. (2014) 301–316
12. Becker, J., Delfmann, P., Eggert, M., Schwittay, S.: Generalizability and applicability of model-based business process compliance-checking approaches — a state-of-the-art analysis and research roadmap. BuR — Business Research **5**(2) (2012) 221–247
13. Rathmair, M., Schupfer, F., Krieg, C.: Applied formal methods for hardware Trojan detection. In: Circuits and Systems (ISCAS), 2014 IEEE International Symposium on. (June 2014) 169–172
14. Kaindl, H.: Using hypertext for semiformal representation in requirements engineering practice. The New Review of Hypermedia and Multimedia **2** (1996) 149–173
15. Kaindl, H.: How to identify binary relations for domain models. In: Proceedings of the Eighteenth International Conference on Software Engineering (ICSE-18), Berlin, Germany, IEEE (March 1996) 28–36
16. Kaindl, H., Kramer, S., Diallo, P.S.N.: Semiautomatic generation of glossary links: A practical solution. In: Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia (Hypertext '99), Darmstadt, Germany (February 1999) 3–12