The Analysis of a Real Life Declarative Process

Søren Debois
IT University of Copenhagen
debois@itu.dk

Tijs Slaats
IT University of Copenhagen & Exformatics A/S
tslaats@itu.dk

***Abstract*—This paper reports on a qualitative study of the use of declarative process notations used in a commercial setting. Specifically, we investigate the actual use of a system implemented in terms of DCR graphs for the Danish "Dreyer Foundation" by our industry partner Exformatics A/S. The study is performed by analysing logs of actual use of the system.**

By the study we seek to illuminate the currently open—and heavily debated!—research question of whether the claimed advantages in conciseness and flexibility of declarative models over more traditional flow-based notations are significant in practice. Studies investigating this question have typically focused on understandability for users, driven by lab experiments and workshops with practitioners unfamiliar with the paradigm.

In the present study, we (1) attempt to assess qualitatively whether users employed the flexibility granted them by the declarative model, and (2) use process discovery techniques to examine if a perfect-fitness flow-based model representing the main business constraints is in fact easy to come by. For (1), we find evidence in various forms, most notably an apparent change in best practices by end-users allowed by the model. For (2), we find no such model. We leave as a challenge to the community the construction of a flow-based model adequately representing the business constraints and supporting all observed behaviour by the users, whether by hand or by mining.

I. Introduction

Modern developed economies are experiencing a transition from production work to knowledge work [1], [2]. Knowledge workers, who are experts in their field and often deal with highly variable problems, need a high degree of flexibility from the IT systems that support their processes [3]–[5]. The workers should be able of applying customised solutions to the unique problems that they face and therefore many different executions of the underlying processes should be supported. Researchers have proposed that declarative, or constraint-based, notations are better at supporting such a high degree of flexibility than more traditional flow-based notations [6]–[8].

However, declarative notations have seen little adoption by industry, and so the question naturally arises whether these notations are in fact solutions to practical needs. This has lead to a number of studies into the usability of declarative notations, but such studies have been limited to lab experiments and short workshops with industry partners being introduced to the new notations [9]–[13]. In essence researchers have been faced with a chicken-and-the-egg dilemma: with little industrial adoption of declarative notations it is difficult to test their usability in practice, but without such experiments it is difficult to determine what is limiting their industrial adoption.

Recently this situation has changed: Exformatics, a Danish vendor of Electronic Case Management (ECM) Systems, has invested heavily in declarative notations. The company started by developing a declarative process engine for their products [14] and more recently created a declarative process modelling portal [13]. As part of a solution developed for the Danish foundation Dreyers fond [15], Exformatics employed a declarative model to describe the process of the customer. By now this solution has been in operation for over a year and the process model and logs from the first half year have been made available for research.

In this paper we attempt to illuminate two important research questions regarding the usability of declarative models by applying process mining techniques [16] to these logs:

1) Do users exploit the flexibility provided by the declarative model?
2) Can automated approaches find a reasonable flow-based replacement for the declarative model?

The first question gives insight into the usefulness of the declarative approach for real-life processes. If we were to discover that the logs exhibit only very structured behaviour, we would have a strong indication that Dreyer in fact does not use the flexibility offered by a declarative model. On the other hand, if the logs do show variable and unstructured behaviour, we can make the qualitative argument that (1) the declarative model does apparently enables flexibility and freedom for end-users, and (2) that end-users do in fact use this flexibility.

The second question helps us understand whether the declarative notation really provides a more concise model for the Dreyers process. The state-space and transition system of the declarative model are so big that constructing a flow-based model allowing all and only the runs of the declarative model would very likely be practically impossible. However, the declarative model may allow for behaviour which is practically irrelevant; the flexibility might be useless. In this case, perhaps there exists a simple, flow-based model which adequately models the processes at Dreyer. We tackle this question by applying automated mining techniques.

For the first question, we find ample evidence of the use of flexibility, in the form of widely varying traces; a long tail of special-cases; and an apparent change in practice by users as time progresses. For the second question, we find that mainstream miners emphatically do not produce helpful

Authors listed alphabetically. This paper would not have been possible without the vision of and encouragement from our industry partner, Exformatics A/S; in particular CEO Morten Marquaard. The research reported on here is supported by ITU, Exformatics A/S and the Velux Foundation through the Computational Artefacts (CompArt) project (http://www.compart.ku.dk).

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.196

models; in particular, the mined models are either hopelessly complex, omit important business constraints, or both.

We emphasise that both questions are treated qualitatively, and that we do not pretend to present a firm conclusion on the relative merits of flow-based and declarative notations. In essence, this paper reports on a failure to disprove the superiority of declarative notations for the present case.

Related work. The relative merits of declarative and imperative workflow notations have been the subject of considerable study and debate. In [10], the authors propose, based on existing research on cognitive aspects of programming language design to business process notations, that "circumstantial" changes are easier to apply in declarative notations, whereas "sequential" changes are more easily applied to imperative notations. Subsequent empirical studies found that imperative notations are easier understood[1] [17]; respectively that imperative and declarative notations exhibit *no* difference in understandability [18]. Other important works include [9], [12], [13], which investigates in various ways understandability of declarative notations.

The present work differs in that we are not interested in understandability of the entire-model, but rather in seeing (a) whether end-users, regardless of their understanding the underlying declarative model or not, do exploit the flexibility ostensibly afforded by declarative models; and (b) whether the declarative model in question could have been easily machine-discovered.

For want of space, figures in this paper are somewhat small; please find full-size copies electronically at [19].

II. The case

The Dreyer Foundation awards grants to projects and activities aimed at promoting the development of the lawyer and architect professions, and their interaction with society. In 2013 the Dreyer Foundation awarded grants amounting to a total of DKK 15.9M/EUR 2.1M.

Towards implementing an ECM solution for the Foundation, Exformatics developed a process model of the Foundation's internal processes regarding both application assessment, rejections, grants and payouts. Roughly, an application is processed as follows. Applications are accepted in rounds. In each round, first, a caseworker pre-screens applications, weeding out obvious rejects. The remaining applications are independently reviewed by 2-4 reviewers, at least one of which must be an architect or a lawyer, depending on the type of application. Once all reviews are in, the Foundation's board decides on which applications to accept on a board meeting. Successful applications then have a running payout, until the grant period expires and an end-report is produced. In practice, board members and reviewers overlap, and reviews might happen or be amended *at* the board meeting.

III. The model

Exformatics modelled the Dreyers process using Dynamic Condition Response (DCR) Graphs, a declarative process notation developed at the IT University of Copenhagen [7], [8], [20]; the model is visualised in Figure 1. We will explain the DCR notation only very briefly here, and refer the reader to further sources for details on the semantics [20]–[22] and supporting tools [13], [14].

A DCR graph comprises *activities* and *relations* between activities. In the diagram, activities are rounded boxes, with the label of the activity on top and the role executing the activity at the bottom. Activities are arranged in a *nesting hierarchy* indicated by dashed boxes. Relations are arrows between activities or nestings; the latter is a short-hand, indicating that the relation applies to all entities within the nesting box. There are five kinds of arrows:

- Conditions, drawn as arrows with dots at their head, indicate that the source activity must be executed at least once before the target activity can be. E.g., in Figure 1, *Register decision* must be executed before *Approve* can be.
- Responses, drawn as arrows with dots at their tail, indicate that if the source activity is executed, the target activity must subsequently also be executed. We say then that the target activity is *pending*. E.g., in Figure 1, *Register decision* requires the subsequent execution of *Change phase to board meeting*.
- Milestones, drawn as arrows with diamond heads, indicate that the target activity cannot execute if the source activity is pending.
- Inclusions and exclusions, drawn with respectively '+' and '%' as head, indicate that when the source activity is executed, the target activity is either included or excluded from the workflow. Conditions and pending state are ignored for excluded activities, and excluded activities cannot execute. E.g., in Figure 1, *Approve* excludes *Reject*.

We emphasise that, when included and not restricted by relations, a DCR activity can execute at any time and any number of times.

Each activity of a DCR model can be said to have a state (has it been executed? is it pending? is it included?); the sum of these states is the state of the graph, known as the *marking* of the graph. Executing an activity can change the state of one or more activities, thus taking us from one marking to another; the set of all possible markings equipped with transitions labelled by executable activities forms a *transition system* for the DCR model. From this transition system we derive the formal semantics of a DCR model, namely the set of (finite and infinite) sequences of activities in which every pending activity is eventually executed. We call such sequences "runs". For finite runs, this means that no activity is pending at the end.

Metrics of the model. The DCR model in Figure 1 contains 37 activities, 59 relations, and 11 nesting boxes with a maximum nesting depth of 3. Writing out all the relations implied by nesting, one obtains instead 71 relations.

The transition system or state space associated with this model is too large to practically enumerate on the authors' available hardware; on a Mid-2012 Macbook Pro, enumeration

[1]However, study participants were subsequently discovered to be predisposed towards imperative notations; and evidence of learning of declarative notations in the experiment was uncovered.

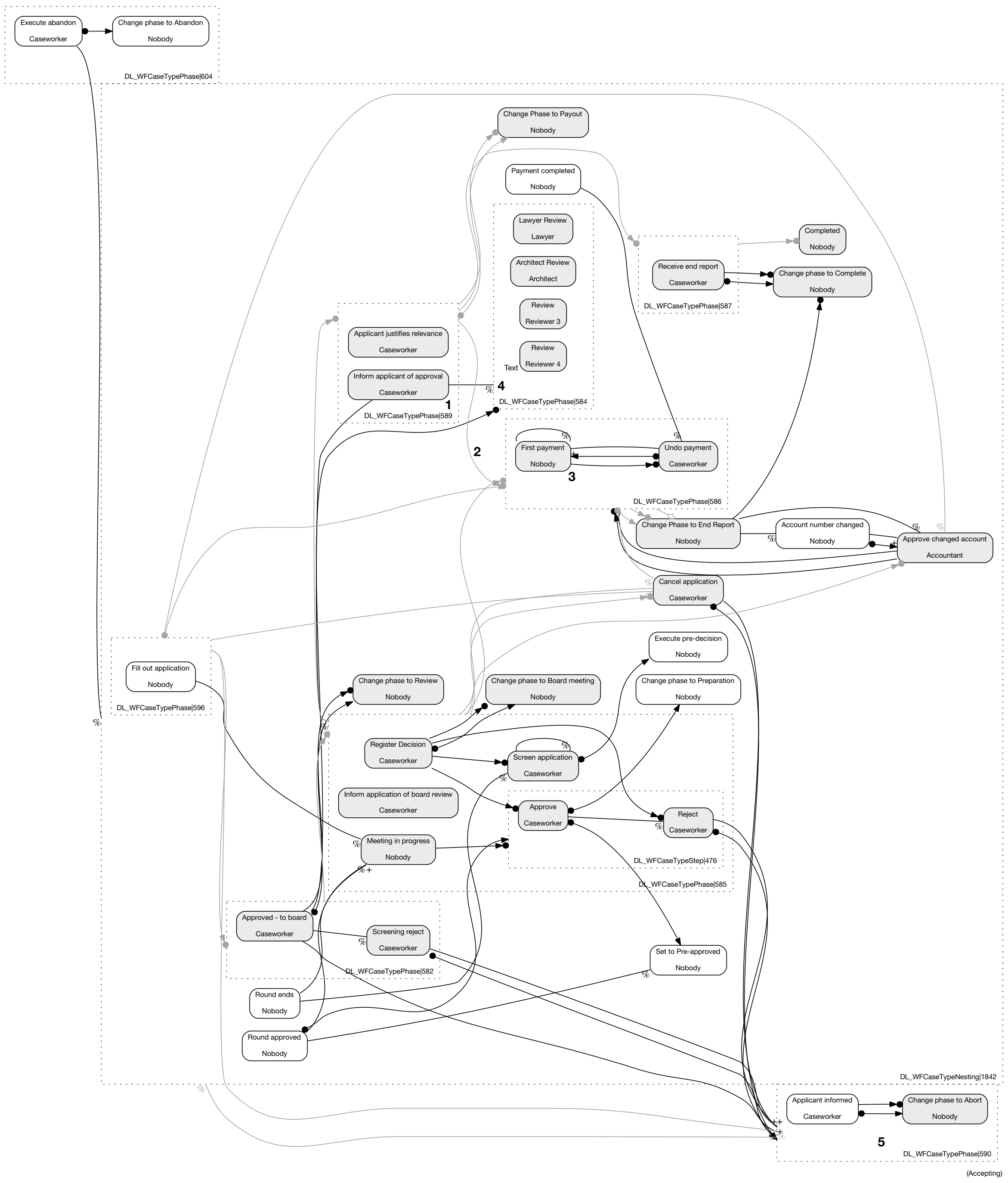

Fig. 1. Dreyer DCR Model

of the transition system exhausts 3.2GB of available memory after having visited 2,977,623 distinct markings by following 19,504,669 transitions (activity executions).

Selection of business rules. Instead of describing every detail in the graph, we select three rules that are both particularly relevant and that can be straightforwardly observed from Figure 1.

1) Before the first payment in a successful application, the applicant must have been informed of the approval.
2) Once an applicant is informed of approval, that decision is final and no changes can be made to the reviews.
3) Once an applicant is informed of rejection, the entire process should stop and the "phase" (a rough indication of where in the process we are) should be changed to abort.

The first rule can be observed in the DCR Graph as a condition relation from the two nestings containing *Inform applicant of approval* to *First payment* ("1", "2", and "3" in the Figure). The second rule is included in the DCR Graph through the exclusion relation from *Inform applicant of approval* to the nesting that contains all review activities ("1" and "4" in the Figure). The last rule is modelled by the exclusion from the *Applicant informed* activity to the nesting surrounding most of the graph ("5"). In addition it requires as a response the activity *Change phase to Abort*, thereby ensuring that the phase is changed.

IV. The Log

The logs supplied to us by Exformatics consisted of an event log and an XML database of DCR models for each process instance[2].

The event log initially contained 12,151 events, but not all of these were interesting for our purposes. Some of the events were meta-events, representing the initialisation (start) of a process, restart of a process and ad-hoc changes to the model. Ad-hoc changes came in two categories; most changes simply added additional tasks to the process, but some changes also affected the rules of the process. Since we are investigating the flexibility of the basic declarative model rather than its support for ad-hoc changes, we chose to remove both meta-event and ad-hoc events from the log. In the case of ad-hoc changes that affected the rules of a process instance, we removed the entire instance from the log. Most importantly this allowed for a fair comparison between mined models and the declarative model.

Removing such events we retained 11,044 events, corresponding to 587 cases or *traces*. However, because the event log contained restart meta-events not present in the DCR formalism, we split the traces in the log across such events, considering the parts before and after distinct cases. This resulted in 797 traces. It's important to note that already the original 587 cases contained incomplete traces; we do not believe we have made mining substantially harder by adding additional incomplete traces.

[2]The original log is available at
http://www.dropbox.com/s/kh7i7ykbj10nc3w/originallog.csv?dl=0

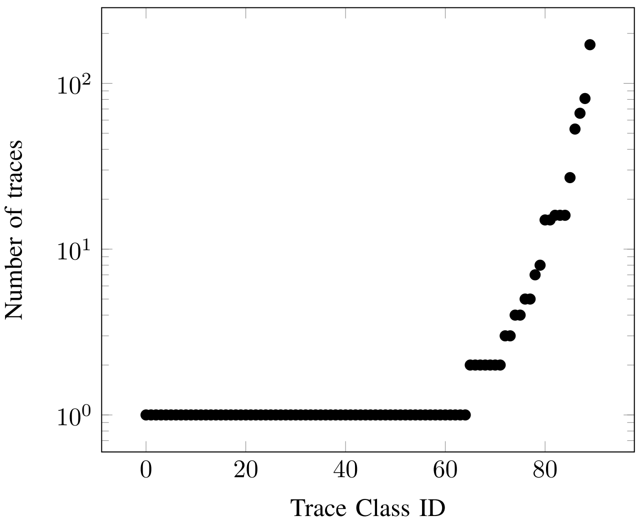

Fig. 2. Trace density. (Space considerations preclude a bar-chart.)

Exformatics took advantage of the declarative process engine and implemented post-deployment updates and bug-fixes by simply adjusting the rules of the process through updates to the underlying declarative model. Therefore the actual models used varied slightly, representing incremental updates to the original model.

However, for our comparison of the declarative model and mined models, it is of course important that there is a single declarative model. If we keep all traces, knowing that they arise from slightly different declarative models, we would be violating a usual assumption of mining, namely that the input log represents some number of process instances *of the same process*. We therefore reduced the set of traces further, by retaining only the traces that conform to the final model used by Exformatics. We replayed all traces against this model and kept only the compliant 594 traces, which had a total of 6470 events[3].

V. Variability of Traces

In this section, we qualitatively assess the amount of variability in the log, with the aim of clarifying research question 1: "Do users exploit the flexibility provided by the declarative model?".

We start by grouping the traces into trace classes that contain traces that are exactly the same. Figure 2 shows the result as a plot with on the x-axis each class and on the y-axis the number of traces in that class. As one can see the 594 traces in the log can be divided in a small number of relatively large classes and a large number of classes with few or unique traces. On first sight this seems to indicate that while there are a number of trace classes that represent "best practice" executions that are commonly used, there is also a large amount of unique traces representing various variations on the best practice.

[3]The processed log is available at
http://www.dropbox.com/s/1l0ur2pr2bpqyix/log.csv?dl=0

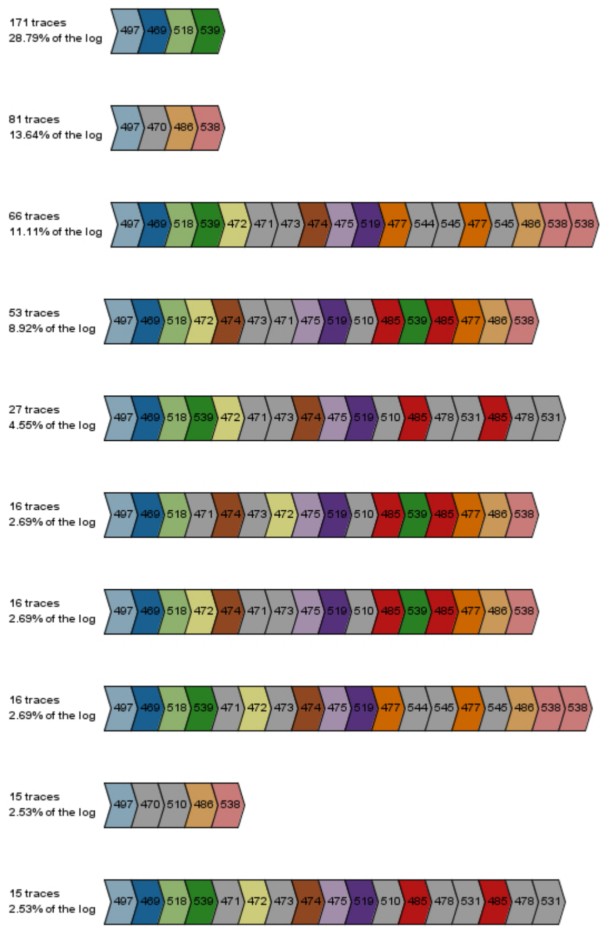

Fig. 3. 10 Most Common Trace Classes

We further investigate this assumption by looking at what the top 10 most populated trace classes actually look like in Figure 3. Here we see that the most common trace class, with 171 traces of only 4 events, represents a set of process instances from the 2nd round of applications which have been approved for review by the board, but are still awaiting the aforementioned reviews. Since these process instances are still far from completion they say little about variability except for that apparently the first steps in the process are often the same. The second most common trace class also contains only 4 events and represents 81 traces. In this case it turns out that we are dealing with process instances where the application was rejected on formal grounds, before being considered by the board. It is perhaps not surprising that these cases can be handled straightforwardly by the case worker and require little variation. The next two trace classes, with respectively 66 and 53 traces and 18 and 16 events are more interesting. They both represent applications that were rejected after evaluation by the board, but whereas the first class contains only applications from the second round, the second class contains only applications from the first round. The fifth trace class, containing 27 traces of 17 events, appears to represent approved applications in the second round. Judging by the final event the traces in this class have not been completed yet. The sixth and seventh trace class, each containing 16 traces of 16 events represent two other variations on rejections in the first round of applications. The eighth trace class, containing 16 traces of 18 events represents another variant on rejections in the second round of applications. The ninth trace class, containing 15 traces of 5 events represents a variant on early rejections in the first round. The tenth trace class, containing 15 traces of 17 events represents approved applications in the second round.

It is interesting that the larger trace classes often contain traces from exclusively one round or the other. We hypothesise that this is because the habits and best practice employed by the participants of the process changes over time, changing the way most applications are handled between the two rounds. This is facilitated by employing a flexible declarative model, which allows all behaviour falling within the rules instead of just a limited number of common scenarios and therefore allows the methods of the users to evolve over time without requiring changes to the process definition and underlying IT systems.

Further investigating the smaller trace classes also shows that while there are clear best practice runs of the process, many variations on the best practice exist and have been used over the half-year run of the process. Figure 4 shows an example of ten arbitrarily chosen variations.

Parallelism. Of course variability in a log does not necessarily indicate declarative flexibility, but can also be a result of parallelism in the process leading to many different possible interleavings. We do not believe that this is the case for the Dreyers process: the variability in, e.g., the traces shown in Figures 3 and 4 arises not just from re-ordering events, but also from varying degrees of repetition and plain absence of some events in some traces. This kind of variability in the number of occurrence (and possible absence) of events is commonly offered by declarative models and is not straightforwardly mimicked by the use of parallelism in flow-based models. For example we observe that the fourth, sixth and seventh trace class, each representing rejections in the first application round, all have a very similar structure: their only variation appears to come from the ordering of the reviews. While this can easily be handled by parallelism in flow-based notations, we observe that many of the single-instance trace classes also represent rejections in the first round, but have larger differences from these four base cases, that can not be handled with parallelism alone.

Looping. Another common source of variability in logs is repeatable behaviour, which can be modelled by loops in flow-based notations. Flow-based loops however need to be highly structured, or they easily lead to hard to read "spaghetti-models". On the other hand most declarative notations, including DCR Graphs, assume repeatability of activities. Activities can be repeated freely unless inhibited by constraints or relations. A good example of this kind of repetition best supported by declarative notations is shown in Figure 4. The activities 475 and 519 always appear as a pair. This pair can appear in different contexts (preceded and succeeded by various other activities) and can be absent, occur once or occur multiple times in a single trace. The pair can be repeated in direct succession, or repetitions can be separated by other activities.

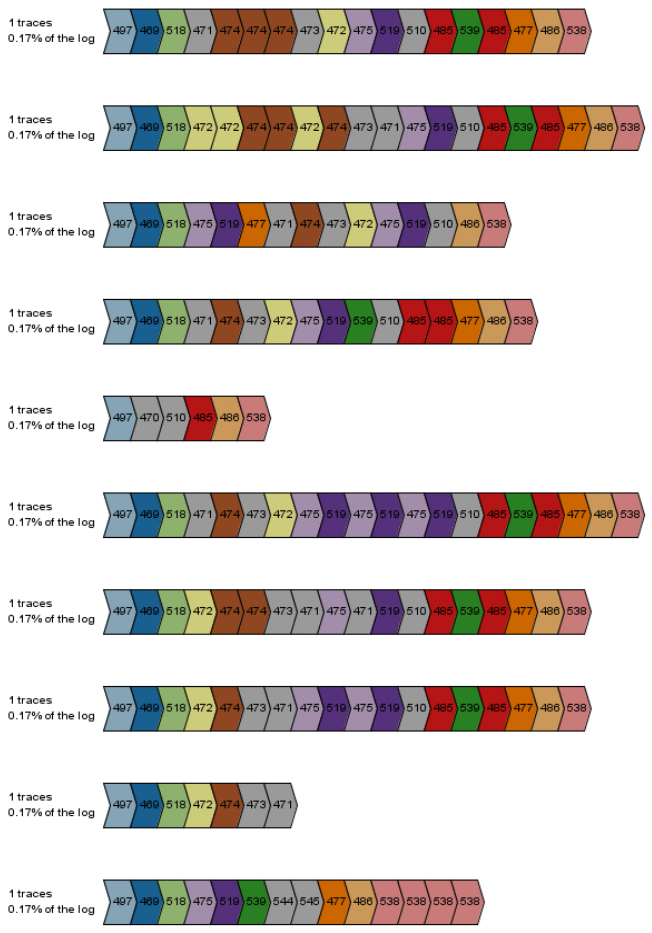

Fig. 4. 10 Arbitrary Unique Traces

In some cases even the pair itself can be interleaved with other events.

In conclusion, based on a qualitative analysis, it appears there is ample use of flexibility in the model, which cannot be explained away by parallelism or looping behaviours. One interesting point of variability is that since variations tend to fall into one round or the other, it seems likely that users have exploited the flexibility to change their practice between rounds.

VI. Flow-based Discovery Results

In order to answer our second research question: "Can automated approaches find a reasonable flow-based replacement for the declarative model?", we run a number of flow-based discovery algorithms on the logs and analyse the resulting models. We will show that contemporary miners cannot produce such a suitable alternative. However, from this we of course cannot conclude that no such model exists; we can merely see that no such model is easily procured using a miner.

But first: What should a model satisfy in order to "reasonably replace" the declarative model? We propose the following requirements, inspired by the standard quality criteria for process discovery [23]:

1) We know that all traces in the log are valid behaviour and stem from *a single* declarative model. Hence the model should have perfect *fitness*.
2) The original declarative model expresses business constraints. The mined model should respect these; that is, it must be *precise*. For instance, it should respect the three rules listed in Section III.
3) The flow-based model should be at least as *simple* as the declarative model.
4) The flow-based model should be sufficiently *general*, that is, it should allow further developments of end-users best practices.

DCR-based Transition System Miner. Since DCR graphs have transition system semantics, an obvious starting point is to simply extract the partial transition system actually explored by the logs. Note that this is different from one would obtain from the ProM transition system miner, since we are exploiting that we know the actual transition system of the model: choice points are guaranteed to be correct wrt. that model.

Figure 5 shows the resulting transition system, which is in essence a cross section of the full transition system underlying the DCR Graph with the traces in the log, containing only those states and transitions that have been visited in practice. States are annotated and shaded based on the number of times they are visited, transitions are annotated by the label of the activity being executed and the number of times that they occur. Square boxes represent accepting states according to the log. Because not all traces in the log have been completed, these are not necessarily accepting states in the original model, but since the other miners can only use the log in their construction of a model we feel that this choice of accepting states yields the fairest model for comparison.

By construction, the transition system has perfect fitness and precision. We feel that it offers a somewhat acceptable result in terms of simplicity as well: on the one hand it requires 246 states and 336 transitions, making it hard to get a general overview of all allowed behaviours, but on the other hand it is fairly straightforward to observe the details of particular paths through the process. The transition system lacks strongly in terms of generality however: with the exception that loops could be taken any number of times the modelled behaviour is exactly that what was encountered in the log. Considering the immense size of the full transition system of the DCR Graph and the fact that we already encountered significant changes in behaviour between the first and second round of applications, it is to be expected that the users will both evolve their best-practice and encounter new special cases in the future, which are supported by the DCR Graph model, but not by the transition system as they are not found in the current log. It is also clear that increased generality of the transition system model based on logs of behaviour past the first half year will incrementally lower its simplicity.

Inductive Miner. We mined the process log with the inductive miner [24] set to find a model representing all paths, thus yielding perfect fitness. The result is shown in Figure 6, while the model is certainly simple, the miner has modelled a large part of the process as a flower-model, trading precision for generality. In particular, business rules are not upheld, e.g., the model does not enforce the first two business rules identified in

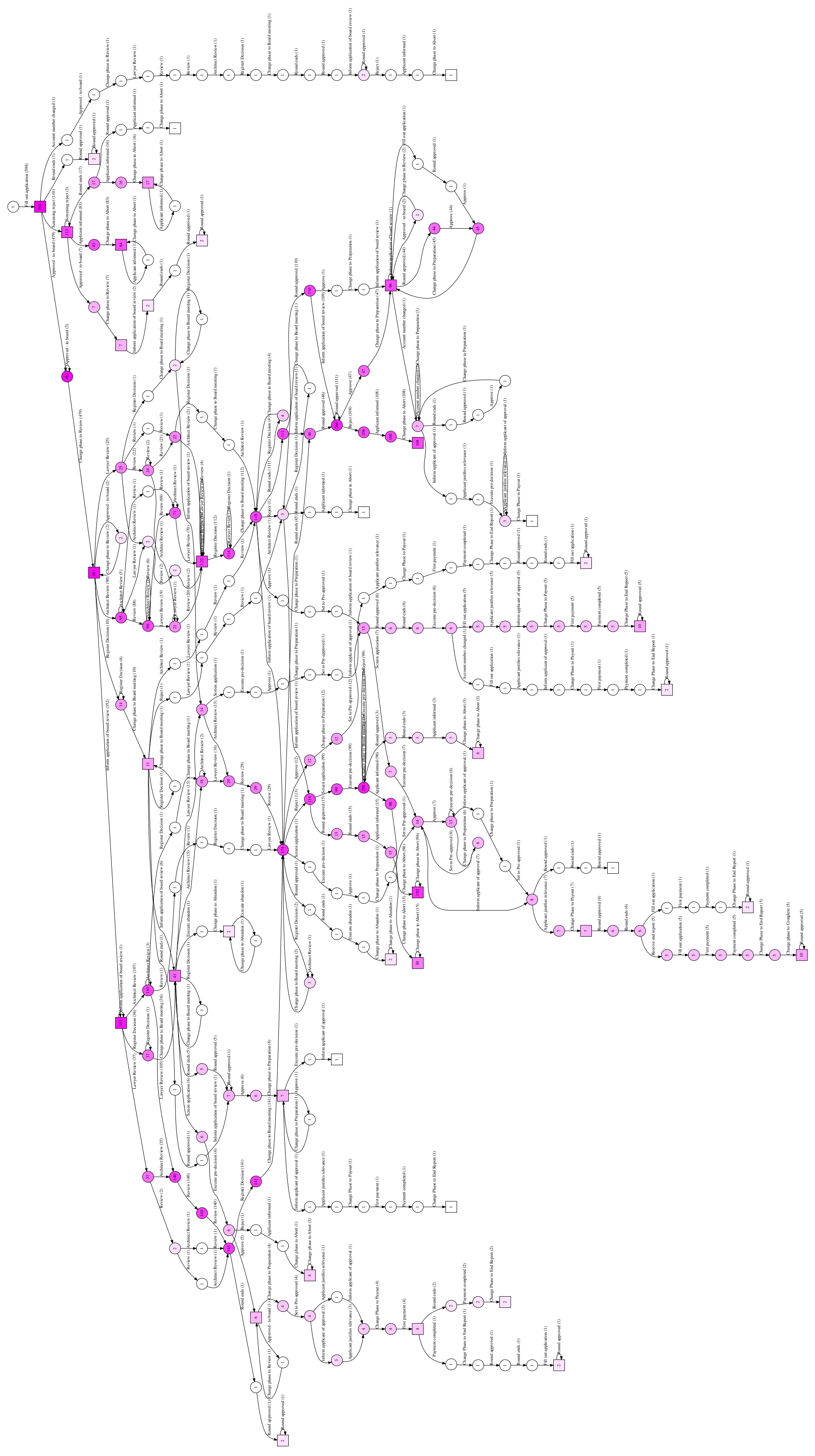

Fig. 5. The Transition System of the Log.

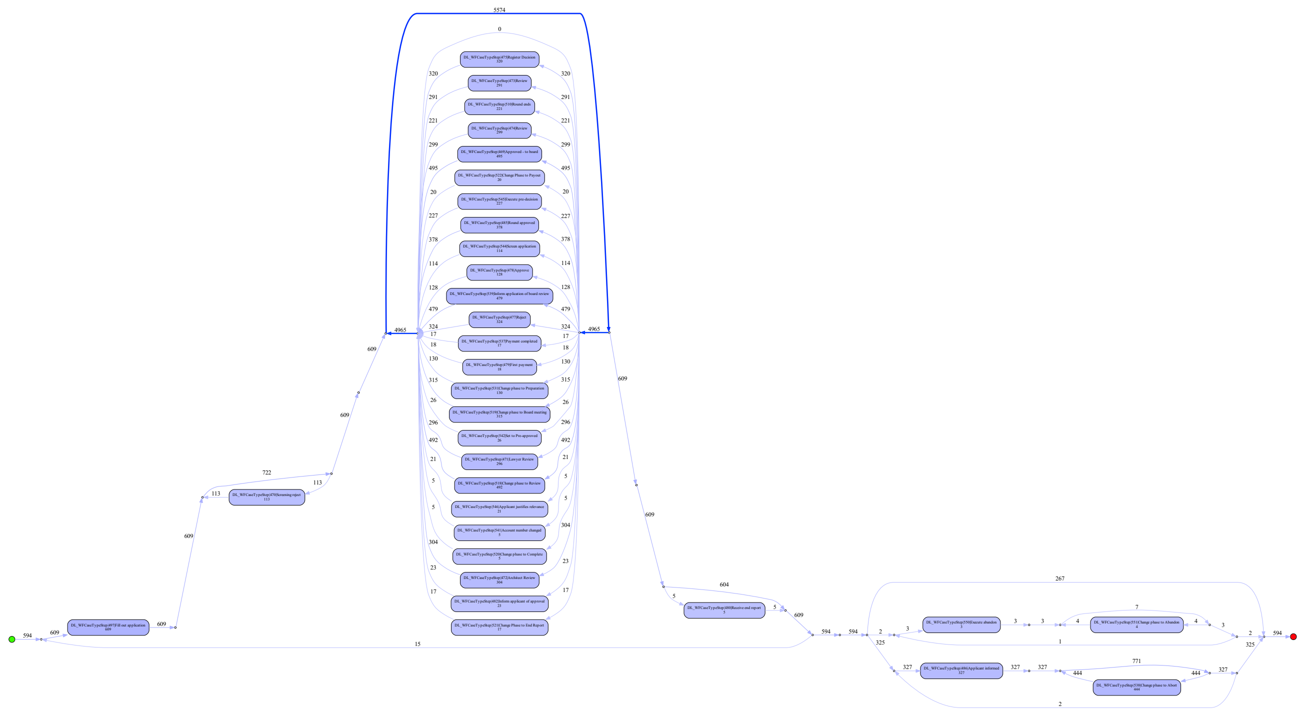
Fig. 6. Model Mined By The Inductive Miner

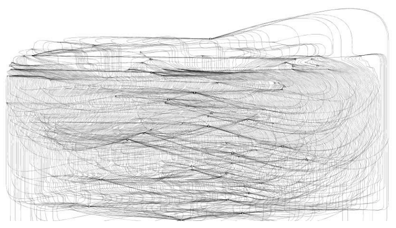
Fig. 7. Model Mined By The ILP Miner

Section III, allowing, e.g., “application” to be followed directly by payments, with no intervening review or approval.

ILP Miner. The ILP Miner [25] also finds a perfectly fitting model, however the resulting Petri net, shown in Figure 7, is complex to the point of uselessness. We have not attempted to rate the model on precision and generality.

Other Miners. We experimented with a number of other process discovery algorithms that did not yield perfect fitness: the alpha miner (0.64 fitness) [26], the heuristics miner (0.48 fitness) [27] and the Fuzzy Miner (0.88 fitness) [28].

In conclusion, we failed to find a mining technique that provided perfect fitness, reasonable simplicity, reasonable precision (as respect for business rules), and reasonable generality. Our attempts seem bracketed by the DCR transition system miner on the one hand, providing perfect precision but little generality, and the inductive miner on the other, providing generality but little precision.

Since we apparently cannot machine-generate a model, the question arises whether a hand-crafted model could strike a better balance. We leave this question as a challenge for the community.

VII. Conclusion

In this paper we investigated the logs generated by a real life, declaratively modelled process. We first undertook a qualitative investigation of the variability present in the log to determine to what extent the users have exploited the flexibility provided by the declarative model. The log exhibited a large amount of variations on a small set of base cases, demonstrated by a small number of highly populated trace classes and a large number of unique traces.

Secondly we experimented with process discovery algorithms to determine if they could generate a sufficiently concise and precise flow-based model based on the log. We did not find such a model.

Altogether, we were unsuccessful in disproving the hypothesis that the claimed advantages of declarative modelling are indeed helpful. Anecdotally and qualitatively, the present study seem to indicate that declarative modelling was indeed helpful.

Thus, we put forward a challenge to the community to find such a flow-based model, satisfying the quality requirements listed in Section VI: (1) perfect fitness in respect to the real-life behaviour exhibited in the Dreyers logs, (2) reasonable precision in terms of representing the main business rules, (3) a clear advantage in simplicity over the declarative model and (4) a stronger sense of generality than the transition system approach in Section VI.

VIII. Future Work

As the present study is qualitative in nature, the obvious next step is to conduct a companion quantitative study. A first step in this direction is to define precisely what it means for a log to exhibit "variability" and "flexibility" behaviour, based on parameters such as the number of trace classes compared to the size of the log and the string edit distance between the trace classes. Special care needs to be taken to distinguish between straightforward variability resulting from parallelism and looping constructs in traditional flow-based notations and more complex variability which can not be traced back to such flow-based constructs, but instead would require a declarative notation to be described concisely.

We have seen that most variability occurs in the middle of traces, whereas the start and end of the traces seems relatively uniform. Thus the question whether the Dreyers process is a good candidate for a Hybrid modelling approach [11], [29]–[31], where the stricter parts of the process are modelled using a flow-based notation and the more flexible parts are modelled using DCR Graphs. It would be interesting to see if a hybrid mining algorithm [32] would be up to the task of mining a fit and sufficiently precise model from our log, but this does not fall in the scope of the current paper as a hybrid model would still confirm the need for declarative notation.

References

[1] P. F. Drucker, *Management Challenges for the 21st Century*. HarperBusiness, 2001.

[2] T. H. Davenport, S. L. Jarvenpaa, and M. C. Beers, "Improving knowledge work processes," *Sloan management review*, 1996.

[3] M. Reichert and B. Weber, *Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies*. Berlin-Heidelberg: Springer, 2012.

[4] B. Weber, M. Reichert, and S. Rinderle-Ma, "Change patterns and change support features - enhancing flexibility in process-aware information systems," *Data and Knowledge Engineering*, vol. 66, no. 3, pp. 438–466, September 2008. [Online]. Available: http://dbis.eprints.uni-ulm.de/335/

[5] N. A. Mulyar, M. H. Schonenberg, Mans, and van der Aalst, "Towards a Taxonomy of Process Flexibility (Extended Version)." 2007.

[6] T. Hildebrandt and R. R. Mukkamala, "Declarative event-based workflow as distributed dynamic condition response graphs," in *Post-proceedings of PLACES 2010*, 2010. [Online]. Available: http://www.itu.dk/~rao/pubs_submitted/dcrsjournalversionfinal.pdf

[7] T. Slaats, "Flexible process notations for cross-organizational case management systems," Ph.D. dissertation, IT University of Copenhagen, January 2015.

[8] R. R. Mukkamala, "A formal model for declarative workflows - dynamic condition response graphs," Ph.D. dissertation, IT University of Copenhagen, March 2012.

[9] C. Haisjackl, S. Zugal, P. Soffer, I. Hadar, M. Reichert, J. Pinggera, and B. Weber, "Making Sense of Declarative Process Models: Common Strategies and Typical Pitfalls," in *BMMDS/EMMSAD*, ser. Lecture Notes in Business Information Processing, vol. 147. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 2–17.

[10] D. Fahland, D. Lbke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, and S. Zugal, "Declarative versus imperative process modeling languages: The issue of understandability," in *Enterprise, Business-Process and Information Systems Modeling*, ser. Lecture Notes in Business Information Processing. Springer, 2009, vol. 29, pp. 353–366.

[11] H. A. Reijers, T. Slaats, and C. Stahl, "Declarative modeling – An academic dream or the future for BPM?" in *BPM 2013*, 2013, pp. 307–322.

[12] S. Zugal, P. Soffer, C. Haisjackl, J. Pinggera, M. Reichert, and B. Weber, "Investigating expressiveness and understandability of hierarchy in declarative business process models," *Software & Systems Modeling*, June 2014. [Online]. Available: http://dbis.eprints.uni-ulm.de/942/

[13] M. Marquard, M. Shahzad, and T. Slaats, "Web-based modelling and collaborative simulation of declarative processes," in *Proceedings of 13th International Conference on Business Process Management (BPM 2015)*, 2015.

[14] T. Slaats, R. R. Mukkamala, T. Hildebrandt, and M. Marquard, "Exformatics declarative case management workflows as DCR graphs," in *BPM '13*, ser. LNCS. Springer, 2013, vol. 8094, pp. 339–354.

[15] S. Debois, T. Hildebrandt, M. Marquard, and T. a. Slaats, "A case for declarative process modelling: Agile development of a grant application system." 2014.

[16] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[17] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, and H. A. Reijers, "Imperative versus declarative process modeling languages: An empirical investigation," in *Business Process Management Workshops (1)*, 2011, pp. 383–394.

[18] N. C. Silva, C. A. L. de Oliveira, F. A. L. A. Albino, and R. M. F. Lima, "Declarative versus imperative business process languages - A controlled experiment," in *ICEIS '14*, 2014, pp. 394–401.

[19] "Full-size figures of the present paper." http://bit.ly/1KVtMem.

[20] T. T. Hildebrandt and R. R. Mukkamala, "Declarative event-based workflow as distributed dynamic condition response graphs," in *PLACES '10*, 2010, pp. 59–73.

[21] T. Hildebrandt, R. R. Mukkamala, and T. Slaats, "Nested dynamic condition response graphs," in *FSEN '11*, April 2011.

[22] S. Debois, T. Hildebrandt, and T. Slaats, "Hierarchical declarative modelling with refinement and sub-processes," in *Business Process Management*, ser. LNCS. Springer, 2014, vol. 8659, pp. 18–33.

[23] J. Buijs, B. van Dongen, and W. van der Aalst, "On the role of fitness, precision, generalization and simplicity in process discovery," in *On the Move to Meaningful Internet Systems: OTM 2012*, ser. LNCS. Springer Berlin Heidelberg, 2012, vol. 7565, pp. 305–322.

[24] S. Leemans, D. Fahland, and W. van der Aalst, "Discovering block-structured process models from event logs containing infrequent behaviour," in *Business Process Management Workshops*, ser. Lecture Notes in Business Inf. Proc. Springer, 2014, vol. 171, pp. 66–78.

[25] J. van der Werf, B. van Dongen, C. Hurkens, and A. Serebrenik, "Process discovery using integer linear programming," in *Applications and Theory of Petri Nets*, ser. LNCS, K. van Hee and R. Valk, Eds. Springer, 2008, vol. 5062, pp. 368–387.

[26] W. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: discovering process models from event logs," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 9, pp. 1128–1142, Sept 2004.

[27] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP*, vol. 166, pp. 1–34, 2006.

[28] C. W. Günther and W. M. van der Aalst, "Fuzzy mining—adaptive process simplification based on multi-perspective metrics," in *Business Process Management*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, vol. 4714, pp. 328–343.

[29] S. Sadiq, W. Sadiq, and M. Orlowska, "Pockets of flexibility in workflow specification," in *Conceptual Modeling – ER 2001*, ser. Lecture Notes in Computer Science, 2001, vol. 2224, pp. 513–526. [Online]. Available: http://dx.doi.org/10.1007/3-540-45581-7_38

[30] W. van der Aalst, M. Adams, A. ter Hofstede, M. Pesic, and H. Schonenberg, "Flexibility as a service," in *Database Systems for Advanced Applications*, ser. Lecture Notes in Computer Science, 2009, vol. 5667, pp. 319–333.

[31] M. Westergaard and T. Slaats, "Mixing paradigms for more comprehensible models," in *BPM 2013*, 2013, pp. 283–290.

[32] F. M. Maggi, T. Slaats, and H. A. Reijers, "The automated discovery of hybrid processes," in *Proceedings of 12th International Conference on Business Process Management (BPM 2014)*, 2014, pp. 392–399. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10172-9_27