Dynamic Condition Response Graphs for Trustworthy Adaptive Case Management

Thomas Hildebrandt[1], Morten Marquard[2],
Raghava Rao Mukkamala[1], and Tijs Slaats[1,2,⋆]

[1] IT University of Copenhagen, Rued Langgaardsvej 7, 2300 Copenhagen, Denmark
{hilde,rao,tslaats}@itu.dk
http://www.itu.dk
[2] Exformatics A/S, Lautrupsgade 13, 2100 Copenhagen, Denmark
{mmq,ts}@exformatics.com
http://www.exformatics.com

Abstract. By trustworthy adaptive case management we mean that it should be possible to adapt processes and goals at runtime while guaranteeing that no deadlocks and livelocks are introduced. We propose to support this by applying a formal declarative process model, DCR Graphs, and exemplify its operational semantics that supports both run time changes and formal verification. We show how these techniques are being implemented in industry as a component of the Exformatics case management tools. Finally we discuss the planned future work, which will aim to allow changes to be tested for conformance wrt policies specified either as linear time logic (LTL) or DCR Graphs, extend the language with time and data and offer extended support for cross-organizational case management systems.

Introduction

Adaptive case management (ACM) processes are characterized as *unpredictable*, *emergent* and *individual* in nature and typically being carried out by knowledge workers [8,11]. At the same time, many case management processes (e.g. in healthcare or the financial sector) are of a critical nature and subject to regulations, demands for visibility and efficiency.

Consequently, *trustworthy* adaptive case management should allow the case workers to iteratively adapt/model the process during its execution, possibly by combining and adapting process fragments [9] obtained from repositories, and verify if the current process description can indeed fulfill the (currently identified) goals, without violating any of the (current) regulations or rules.

In the present paper we show how Dynamic Condition Response Graphs (DCR Graphs) [1–3] can be used for the specification and trustworthy, adaptive execution of case management processes. DCR Graphs is a formal, *declarative* process modeling notation introduced in the Trustworthy Pervasive Healthcare Services (www.TrustCare.dk) research project as described in the third author's PhD thesis [6], and currently being

⋆ Authors listed alphabetically. This research is supported by the Danish Research Agency through an industrial PhD Grant.

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 166–171, 2013.

embedded in the Exformatics case management tools [10] as part of the industrial PhD project carried out by the last author.

Declarative process notations, such as Declare [12] (based on templates formalized as LTL formulae) and the *Guard-Stage-Milestone* model [5] (based on ECA-like rules) have been put forward as offering flexibility in execution and modelling. By declarative it is usually meant that the notation is aimed at describing *what* is to be achieved, i.e. goals and regulations. This is opposed to *imperative* process notations aiming at describing *how* the goal is to be achieved.

Similar to other declarative models any DCR Graph can be mapped to a (Büchi) automaton describing the possible executions and which executions fulfill the goal of the process. This mapping makes it possible to formally verify the process using standard model-checking tools such as the SPIN model checker.

However, if the process is to be adapted/changed *during* its execution, the compilation of a declarative description into a lower level, imperative description (*before* execution) constitutes a problem. In case the change is applied to the original declarative description, it also affects the process as it was before its execution. On the other hand, if the change is applied to the imperative description of the process being executed, the change must be expressed in terms of the operational model, which requires a non-trivial translation of the meaning of the change between the declarative and the operational model.

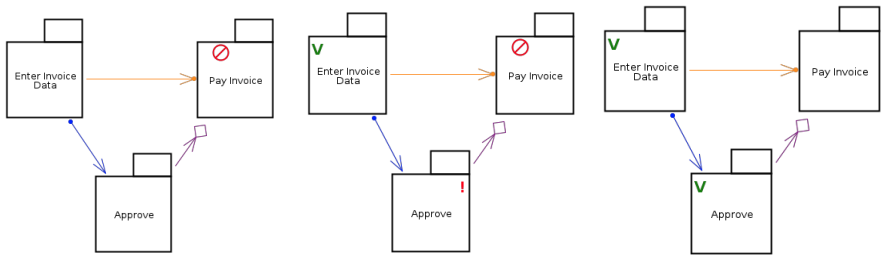
A key feature of the DCR Graph notation is that it allows a formal operational semantics in which the intermediate states of running processes are described by a simple marking of the graph, that can also be understood by the case worker. This allows for changes to the declarative description to be applied and take effect in intermediate states of the execution as we will exemplify below.

State of the Art

DCR Graphs by Example. Fig. 1(a) shows an example of a DCR Graph model designed in the DCR Graph editor developed at Exformatics. It represents an invoice workflow with just three events: Enter Invoice Data, Approve and Pay Invoice. The events are represented as boxes with their assigned roles given in small ears at the top. The example process declares two roles: Administration (Adm), representing the administration office of a company and Responsible (Res) the person responsible for the invoice. The administration office has access to the tasks Enter Invoice Data and Pay Invoice and the responsible has access to the task Approve.

Unconstrained events in a DCR Graph can be executed any number of times and in any order. In order to eliminate undesirable behavior in the process, a DCR Graph contains a set of constraints defined using five different kinds of relations between the events, named the condition ($\rightarrow\bullet$), response ($\bullet\rightarrow$), milestone ($\rightarrow\diamond$), inclusion ($\rightarrow+$) and exclusion ($\rightarrow\%$) respectively. The use of the condition, response and milestone relations is illustrated i Fig. 1(a), where we have declared three constraints on the workflow. The first constraint is that we can not pay an invoice before data has been entered. This is declared by the condition relation (an arrow with a bullet at its tip) from Enter Invoice Data to Pay Invoice. But note that we can enter data twice before paying the invoice, and that Pay Invoice is not required to happen when it gets enabled.

The second constraint is that an approval must eventually follow when invoice data has been entered, which is declared by the response relation (an arrow with a bullet at its start) from Enter invoice Data to Approve. Note that this constraint is also fulfilled if several Enter Invoice Data events are followed by a single Approve event. Finally, the third constraint is that if an approval is pending, i.e. required but not yet done, then it is not allowed to pay the invoice. This constraint is declared by the milestone relation (an arrow with a diamond at its tip) from Approve to Pay Invoice. The *stop sign* at the Pay Invoice event is not part of the graph, but indicates that the event is not enabled in the current marking (where no events have been executed and thus the condition constraint is not satisfied).



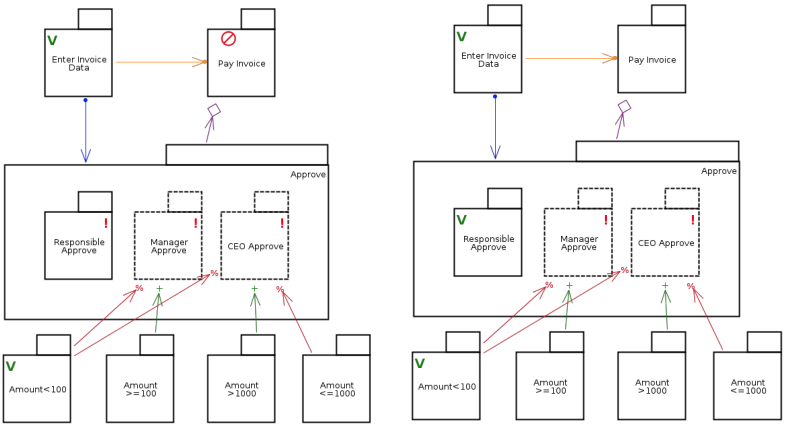
(a) DCR Graph for invoice workflow (b) Marking after data (c) Marking after approve

Fig. 1. Examples of DCR Graphs with markings

If Enter Invoice Data is executed, the marking of the graph changes to the one in Fig. 1(b). The checkmark inside Enter Invoice Data shows that it has been executed, the exclamation mark inside Approve shows that it is required to be executed before the workflow has reached its goal. Now, Enter Invoice Data can be executed again without changing the marking, since it does not record how many times an event has been executed. Pay Invoice is still not enabled, even though the condition constraint that Enter Invoice Data should be executed first is fulfilled, the milestone relation disables Pay Invoice since Approve is required to happen. Now, if Approve is executed, the marking changes to the one in Fig 1(c) where all events are enabled, but no events are required.

The final two constraints are the dynamic exclude ($\rightarrow\!\%$) and its dual the dynamic include ($\rightarrow\!+$). The dynamic exclude relation is used to remove an event dynamically from the workflow when another event happens, and the include relation to add an event (back) to the workflow. Excluded events can never be executed, but neither are they considered when evaluating enabledness or when evaluating whether the workflow is completed.

Fig. 2(a) shows an adaptation of the invoice process that makes use of the dynamic inclusion and exclusion relations. The Approve event is adapted to contain three *sub events*: Responsible Approve, Manager Approve and CEO Approve. The meaning of the relations to and from Approve is simply that they relate to all sub events of Approve. The sub events also inherited the response marking and are thus all required.



(a) Adapted process with excluded events (b) After Responsible Approval

Fig. 2. Example of nested events and dynamically included and excluded events

However, three other events have also been added to represent the decision logic behind the detailed approval process: Amount $<$ 100, Amount $>=$ 100 and Amount $>=$ 1000. These events are to be executed by the system when the amount (entered as part of the Enter Invoice Data event) changes and the constraint is satisfied. For the sake of the example, we assume the amount entered was less than 100, so the Amount $<$ 100 event has been executed. This resulted in Manager Approve and CEO Approve to be excluded, as represented by the dashed boxes. Fig. 2(b) shows the resulting marking if Responsible Approve is executed. Pay Invoice is enabled since the required responses on the excluded events Manager Approve and CEO Approve are not considered when the milestone relation is evaluated.

Adaptive DCR Graphs. As shown in the previous examples, when using DCR Graphs we maintain information on the original events and relations of the model at runtime. Executing a DCR Graph simply results in a new DCR Graph with a (optionally) changed marking. This makes it easy to adapt the model at runtime: one can add or remove relations and events and continue executing afterwards. In recent work [7], we investigated this in more detail and formalized a number of operations for adding/removing events and relations to DCR Graphs and composing process fragments. In the same paper we presented techniques for checking DCR Graphs for dead- and live-lock, and showed how these techniques can be used for safely adapting DCR Graphs at runtime.

DCR Graphs at Exformatics. DCR Graphs have been adopted by our industrial partner Exformatics A/S as the underlying formal model for workflows in their state-of-the-art Electronic Case Management (ECM) system, which is used by various Danish and international clients, both in the private and public sector, for case- and project-management.

Exformatics has developed a number of tools to support the modeling and execution of DCR Graphs. These tools are stand-alone and can be used without requiring installation of their Electronic Case Management product. The tools include several webservices for execution, verification, storage and visualization of DCR Graphs based workflows and a graphical editor for modelling, simulation and visualization of DCR Graphs. All diagrams in this paper have been produced by the editor, which also supports adaptation during a simulation. However, assignment of data values as part of events and the automatic execution of events based on changes in data values is not supported yet, i.e. it must so far be manually simulated by the designer.

Future Work

Currently we are extending the verification techniques for DCR Graphs to allow for checking any arbitrary property, defined as either a LTL formula or a DCR Graph. This technique will allow us to verify that run-time adaptations of a process model continue to adhere to a set of given policies. We are also exploring different ways of handling data within DCR Graphs, inspired by different application domains that have different requirements to the processes being modelled. The first proposal [6, 10] has been described in the examples. The second proposal is aimed at cross-organizational workflows where there is no central location of the data that all parties have access too. In this approach events are parameterized by variables, for example an employee ID, and occurrences of the same event with different parameters are considered distinct. Relations in this approach are relative to the parameters on the events, ie if Enter Invoice Data[employeeId] requires Approve[employeeId] as a response, then an occurrence of Enter Invoice Data for a specific employee requires approval by that same employee. The third approach is based on a hybrid model that combines DCR Graphs with Coloured Petri nets. [13] The motivation for the hybrid model was an attempt to combine declarative and imperative workflow languages, but it also allows for data constraints to be added to DCR Graphs by using a language that is already popular as a formal foundation for workflows. In future work we will select the approach that best fits to adaptive processes and extend the adaptation operators and verification techniques to also cover this data variant. In addition we will extend the aforementioned techniques to work on timed DCR Graphs [4]. Finally an algorithm for safely distributing the execution of a process has been developed [3]. This can be used to provide a global description of a collaborative process, which can then be distributed on different participants. In future work we plan to extend the theory of distributed DCR Graphs with a behavioral type system, which will allow us to more efficiently verify cross-organizational adaptive processes.

Conclusion

We have outlined by an example the declarative primitives and operational semantics for DCR Graphs and how the model could be used as the foundation for trustworthy adaptive case management. The declarative nature of the model provides flexibility in execution, i.e. it avoids the usual overspecification of flow graphs, as well as flexibility with respect to adaptations: Process fragments can be composed and events and

constraints can be added and removed through three basic adaptation operations, without breaking the syntactic validity of the graph, and correspond to the identification of new activities and rules in an emergent process. The representation of the state of a process in terms of a marking that can be understood by the case worker means that adaptations can be made to partially executed processes. The formal semantics and mapping to Büchi-automata (also of intermediate process states), makes it possible to formally verify if it is still possible to achieve the goal of the process after adaptation. We also showed how the work is being implemented by our industrial partner in their ECM system. Finally we discussed planned future work that will make DCR Graphs even more suitable for supporting trustworthy adaptive case management processes.

References

1. Hildebrandt, T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: PLACES. EPTCS, vol. 69, pp. 59–73 (2011)
2. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Nested dynamic condition response graphs. In: Arbab, F., Sirjani, M. (eds.) FSEN 2011. LNCS, vol. 7141, pp. 343–350. Springer, Heidelberg (2012)
3. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Safe distribution of declarative processes. In: Barthe, G., Pardo, A., Schneider, G. (eds.) SEFM 2011. LNCS, vol. 7041, pp. 237–252. Springer, Heidelberg (2011)
4. Hildebrandt, T., Mukkamala, R.R., Slaats, T., Zanitti, F.: Contracts for cross-organizational workflows as timed dynamic condition response graphs. Journal of Logic and Algebraic Programming (JLAP) (May 2013), http://dx.doi.org/10.1016/j.jlap.2013.05.005
5. Hull, R.: Formal study of business entities with lifecycles: Use cases, abstract models, and results. In: Bravetti, T., Bultan, M. (eds.) 7th International Workshop on Web Services and Formal Methods. LNCS, vol. 6551 (2010)
6. Mukkamala, R.R.: A Formal Model For Declarative Workflows: Dynamic Condition Response Graphs. PhD thesis, IT University of Copenhagen (June 2012), http://www.itu.dk/people/rao/phd-thesis/DCRGraphs-rao-PhD-thesis.pdf
7. Mukkamala, R.R., Hildebrandt, T., Slaats, T.: Towards trustworthy adaptive case management with dynamic condition response graphs. In: Proceedings of the 17th IEEE International EDOC Conference, EDOC (2013)
8. Mundbrod, N., Kolb, J., Reichert, M.: Towards a system support of collaborative knowledge work. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 31–42. Springer, Heidelberg (2013)
9. Sirbu, A., Marconi, A., Pistore, M., Eberle, H., Leymann, F., Unger, T.: Dynamic composition of pervasive process fragments. In: Proceedings of the 2011 IEEE International Conference on Web Services, ICWS 2011, pp. 73–80. IEEE Computer Society, Washington, DC (2011)
10. Slaats, T., Mukkamala, R.R., Hildebrandt, T., Marquard, M.: Exformatics declarative case management workflows as dcr graphs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 339–354. Springer, Heidelberg (2013)
11. Swenson, K.D.: Mastering the Unpredictable: How Adaptive Case Management Will Revolutionize the Way That Knowledge Workers Get Things Done. Meghan-Kiffer Press (2010)
12. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - R&D 23(2), 99–113 (2009)
13. Westergaard, M., Slaats, T.: Mixing paradigms for more comprehensible models. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 283–290. Springer, Heidelberg (2013)