Exformatics Declarative Case Management Workflows as DCR Graphs

Tijs Slaats[1,2], Raghava Rao Mukkamala[1], Thomas Hildebrandt[1], and Morten Marquard[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. Declarative workflow languages have been a growing research subject over the past ten years, but applications of the declarative approach in industry are still uncommon. Over the past two years Exformatics A/S, a Danish provider of Electronic Case Management systems, has been cooperating with researchers at IT University of Copenhagen (ITU) to create tools for the declarative workflow language Dynamic Condition Response Graphs (DCR Graphs) and incorporate them into their products and in teaching at ITU. In this paper we give a status report over the work. We start with an informal introduction to DCR Graphs. We then show how DCR Graphs are being used by Exformatics to model workflows through a case study of an invoice workflow. Finally we give an overview of the tools that have been developed by Exformatics to support working with DCR Graphs and evaluate their use in capturing requirements of workflows and in a bachelor level course at ITU.

Keywords: workflows, declarative specifications, tools, teaching, case study.

1 Introduction

Declarative workflow modelling [8, 9, 16] is an emerging field in both academia and industry which offers a new paradigm that supports flexibility and adaptability in business processes. Traditional imperative workflow languages describe *how* a process is carried out as a procedure with explicit control flow. This often leads to rigid and overspecified process descriptions, that fails to capture *why* the activities must be done in the given order. Declarative workflow languages on the other hand specify processes by the constraints describing *why* activities can or must be executed in a particular order, and not how the the process is to be executed, i.e. activities can be executed in any order and any number of times, as long as not prohibited by a constraint [15, 19]. This may lead to under specified process descriptions and make it difficult to perceive the path from

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

F. Daniel, J. Wang, and B. Weber (Eds.): BPM 2013, LNCS 8094, pp. 339–354, 2013.

start to end, but captures the reason for the ordering of activities and leaves flexibility in execution.

An example a constraint between activities is the response constraint [4, 16] (e.g. $A \bullet\!\!\rightarrow B$), which requires that an execution of one task (A) is eventually followed by an execution of another task (B), but it does not put any further limits on the number of times and order in which the tasks are executed. For example, it would be perfectly valid if the second task occurs first, as long as it also occurs after the first task. In other words, $B, AB, BAB, AAB, ...$ are all valid runs, where as $A, BA, BBA, ...$ are not valid runs, as they fail to satisfy the constraint by having an occurence of A that is not followed by an occurence of B.

Examples of processes that require more flexibility are commonly found in the healthcare [5] and case management [1] domains. In those processes, the work is being carried out by knowledge workers who typically have the experience and expertise needed to deal with the complexity of a process whose requirements may vary from case to case. For this reason, knowledge-intensive processes require flexible workflow systems that support the users in their work (instead of dictating them what to do) and allow them to make their own choices as long as they do not break those constraints that do need to be strictly followed in all cases (e.g. laws or organizational policies).

Over the last decade, several declarative languages for business processes have been proposed in academic literature. The first of these languages is Declare [15, 19] which gave a number of common workflow constraints formalized in Linear-time Temporal Logic (LTL). More recently, DCR Graphs [4] have been developed as a generalization of event structures [21], where processes are described as a graph of events related by only 4 basic constraints. A simple operational semantics based on markings of the process graph makes it possible to clearly visualize the runtime state. Furthermore, the Guard-Stage-Milestone [10] has been developed, which is a *data-centric* workflow model with declarative elements for modeling life cycles of business artifacts.

Even though by now these techniques have become well known in academia, their application in the industry is relatively uncommon. Over the last two years Exformatics A/S, a Danish provider of Electronic Case Management(ECM) systems, has been collaborating with researchers of IT University of Copenhagen (ITU), to develop tools for the declarative workflow language DCR Graphs with the aim to apply and evaluate the use of DCR Graphs on real world scenarios in the case management domain and in teaching at ITU.

The goal of the present paper is to give a status report, presenting and evaluating the tools developed so far. As the first step, the core DCR Graphs model were used by Exformatics A/S in a case study to capture some of the requirements in the design phase of a cross-organizational case management system [1]. The case study led to the further development of the DCR Graphs model by adding support for *hierarchical* modelling using nested events and a (*milestone*) constraint [6], making it possible to concisly specify that some event(s) must not be pending in order for some event to happen. It also encouraged developing a graphical design, simulation and verification tool [18] which is being used successfully in further case studies with industry and in teaching at ITU.

In the remainder of this paper we will first introduce DCR Graphs informally in Sec. 2, in Sec. 3 we will explain how they are used as the underlying formalism for workflows within the Exformatics ECM system and in Sec. 4 we will give an overview of the tools for managing DCR Graphs that have been developed by Exformatics. We evaluate and describe related work in Sec. 5 and conclusions and future work in Sec. 7.

2 DCR Graphs by Example

This section describes DCR Graphs informally by giving an overview of the declarative nature of the language and its graphical modeling notation. (All figures shown are produced in the developed graphical editor and simulation tool [18]). The formal semantics of DCR Graphs are given in [4, 6, 12].

A DCR Graph specifies a process as a set of *events*, typically representing the (possibly repeated) execution of activities in the workflow process, changes to a dataset or timer events. The events are represented graphically as rectangular boxes with zero or more *roles* in a small box on top of the event as depicted in Fig. 1, showing an excerpt of an invoice workflow with three events: Recieve Invoice, Enter Invoice Data and Responsible Approval and two roles: Administration (Adm), representing the administration office of a company and Responsible (Res), the person responsible for the invoice. The administation office has access to the tasks Recieve Invoice and Enter Invoice Data and the responsible has access to the task Responsible Approval.

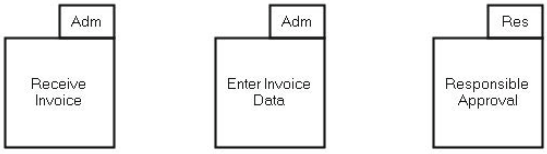


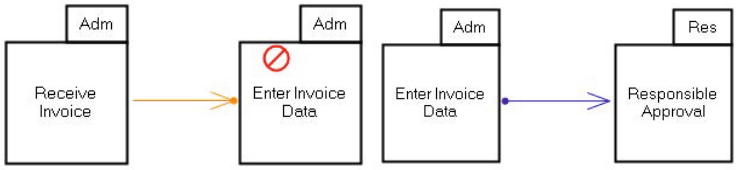
Fig. 1. DCR Graphs: Tasks and Roles

The concrete principals/actors (either human or automated) are typically not shown in the graphical notation, but will at runtime be assigned one or more of the roles and can then execute any of the events that are assigned to one of these roles.

The events in a DCR Graph can happen any number of times and in any order, unless prevented by a constraint relation. The graph in Fig. 1 has no constraints, so it would be valid to e.g. just receive an invoice and do nothing else, or to receive an invoice and then approve the invoice twice. Constraints are defined using five different kinds of relations between the events, named the *condition*, *response*, *milestone*, *inclusion* and *exclusion* relation respectively.

Fig. 2(a) gives an example of a condition relation (depicted graphically as $\rightarrow\bullet$) between Recieve Invoice and Enter Invoice Data, which states that before Enter Invoice Data can happen, the event Recieve Invoice must first have happened. In other words, we have to receive an invoice before we can enter the details of the invoice into

the system. The DCR Graph shown in Fig. 2(a) allows possible runs such as Recieve Invoice.Enter Invoice Data or Recieve Invoice.Enter Invoice Data.Recieve Invoice or Recieve Invoice.Recieve Invoice.Enter Invoice Data, but it does not allow e.g. Enter Invoice Data. Recieve Invoice as it invalidates the condition constraint. As a help for the user, the graphical editor shows a "no entry" sign at the event Enter Invoice Data to indicate that it is not enabled.



(a) The Condition Relation (b) The Response Relation

Fig. 2. The Condition and Response relations

In Fig. 2(b) is given an example of the response relation (depicted graphically as $\bullet\!\!\rightarrow$), which states that if Enter Invoice Data happens, Responsible Approval eventually has to happen in order for the workflow to be completed. Note that this relation is not counting, i.e., it is not required to execute Responsible Approval once for each execution of Enter Invoice Data. In other words, the response relation offers the flexibility of approving one to many invoices just by executing Responsible Approval once. Examples of completed runs in the process represented by the graph in Fig. 2(b) are: Enter Invoice Data.Responsible Approval, and Enter Invoice Data.Enter Invoice Data.Responsible Approval. An example of a run which is possible, but not completed is Enter Invoice Data.Responsible Approval.Enter Invoice Data as the last Enter Invoice Data is not (yet) followed by Responsible Approval.

In [6] we extended DCR Graphs to allow *nested* events as shown in Fig. 3. Nesting both acts as a logical grouping and as a shorthand notation for having the same relation between many events. For instance, the response relation from Enter Invoice Data in Fig. 3 represents a response relation from Enter Invoice Data to all three sub events of the super event Approval.

Adding nesting to the model, made it apparant, that it is useful to be able of express, that an event can not happen when a nested subgraph is not in an accepting state. We call this relation the milestone relation (depicted graphically as $\rightarrow\!\!\diamond$), and is exemplified shown in Fig. 3 from the Approval super event to Pay Invoice. The meaning is, that after doing Enter Invoice Data, we will have a pending response on each approval task and therefore we can't execute Pay Invoice until each of these tasks has been done. Note that in contrast to the condition relation, by using a combination of the response and milestone relations we can require approval again after it was already given.

Finally, the exclude relation (depicted graphically as $\rightarrow\!\!\%$) and its dual the include relation (depicted graphically as $\rightarrow\!\!+$) allows for dynamically respectively exclude and include events from the workflow. Fig. 4(a) shows a typical example of the use of the dynamic include and exclude relations to model *exclusive choice* between events: The

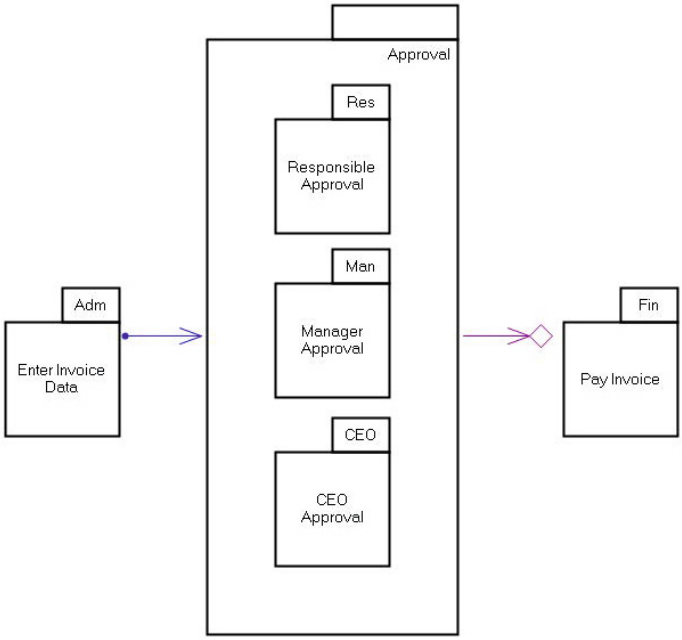
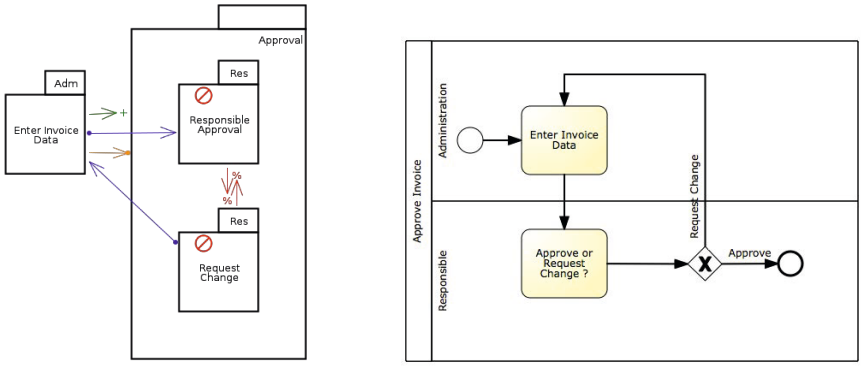


Fig. 3. Example of Nesting and the Milestone Relation

responsible may choose between approving or request a change to the invoice. The choice is modelled by letting the two events mutually exclude each other. If a change is requested, the administration is required to enter data again (because of the response relation from Request Change to Enter Invoice Data), and when data is entered again, the two events nested under the *Approval* super event is included again because of the include relation from Enter Invoice Data to Approval. This example illustrates the flexible nature of DCR Graphs in process modeling, as compared to the typical BPMN procedural model in Fig. 4(b). In the DCR Graph, invoice data can be entered any number of times before approval, and changes can also be requested any number of times before data is entered again, while the BPMN process only allows every task to be executed once for each cycle in the loop. It is of course possible to model the more flexible execution in BPMN, but not in a natural way.

2.1 Execution Semantics

The runtime state of a DCR Graph is defined by a *marking* of the graph, formally given by 3 finite sets of events representing respectivly which events are *executed* (at least once), *pending responses* and *included*. By keeping track of which events have been executed at least once in the *executed* set, we can determine which conditions have been satisfied. The *pending responses* set keeps track of which events need to be executed before the workflow is in a completed state. Finally, the *included* set keeps track of the currently included events. An event is enabled for execution if it is currently included



(a) Modeling choice with include and exclude (b) Imperative BPMN model

Fig. 4. Declarative DCR Graph and imperative BPMN model of invoice approval

(i.e. part of the included set in the current marking) and all of its conditions are either executed or excluded (i.e all condition events that are currently included should be part of the executed events set) and no event related to it by the milestone relation is included and a pending response. A (finite or infinite) execution is defined to be accepting, when no event from some point stays included and as a pending response forever without eventually being executed.

The excluded events are graphically depicted by a dashed border, the executed events by a green checkmark at the event, and pending response events by a red exclamation mark. This is shown in Fig. 5, where Enter Invoice Data and Request Change are executed, and thereby Responsible Approval is a pending response, but it is also excluded and Enter Invoice Data is a pending response too.

A DCR Graph contains an initial marking defined as part of the graph. For example, a graph may have a number of initial pending responses (representing tasks that are required to be executed mandatorily for the workflow to be considered finished), or initially excluded events.

2.2 DCR Graphs with Global Data

In one of the more recent extensions to DCR Graphs [12], we have introduced the concept of global data. In DCR Graphs, data is modelled as a global store that contains a number of named variables. The variables are mapped to events so that we can specify which events can read/write to specific variables. Furthermore, *guards* are defined as boolean expressions over the values of variables. Guards can be placed on both events and relations. If a guard is assigned to an event, then as long as the guard does not evaluate to true, the event is blocked from execution. On the other hand, having a guard on a relation means that the relation is only evaluated when the guard evaluates to true, in other words the condition constraint only needs to hold and an event is only recorded as a response while the guard holds.

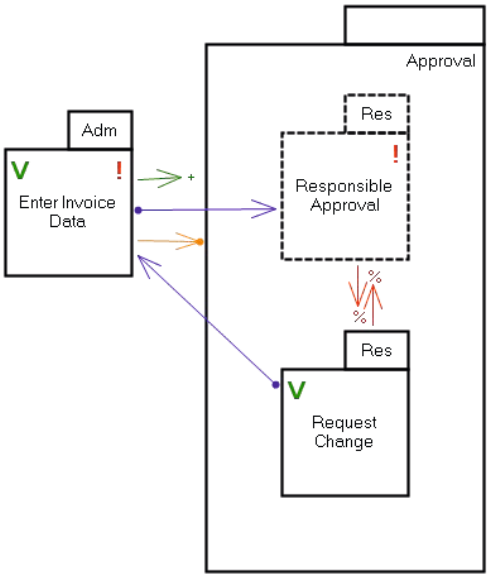


Fig. 5. Example marking after executing Enter Invoice Data followed by Request Change

For example, the response between Enter Invoice Data and Manager Approval in Fig 6 is only recorded when the amount of the invoice is equal or larger than 1000 euro, if the amount is lower than 1000 euros, executing Enter Invoice Data will not make Manager Approval a pending response.

3 Exformatics Workflows as DCR Graphs

Before the introduction of (Nested) DCR Graphs, the Exformatics workflow model consisted of tasks grouped under phases. There was always one active phase, which could be changed manually by the user, tasks belonging to that phase were then enabled. When introducing DCR Graphs we chose to map tasks to events and to maintain the phase model, mapping it to a single-level nesting structure. We removed the practice that tasks were enabled when their phase was active and allowed the active phase to be changed automatically through the execution of certain tasks. In the new model, the active phase no longer controls the workflow but instead just gives a general indication of the state that the case is in. We introduced all five relations of DCR Graphs as ways of constraining the flow of tasks. One distinction from the traditional DCR Graph approach is that tasks in the Exformatics system are normally only done once. As a result, when a task is executed, it is not shown in the list of tasks that need to be done anymore. However, unless it is exlicitely excluded through the exclude relation it remains possible to open the task again manually and do it again, so the execution semantics remains faithful to the DCR Graphs semantics.

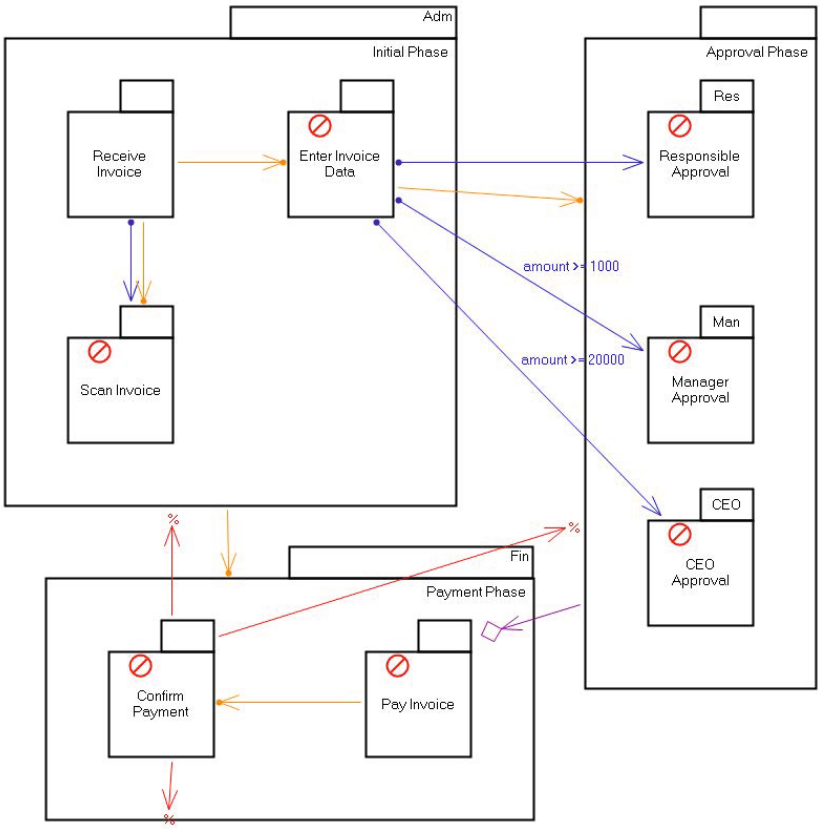


Fig. 6. Exformatics Invoice Workflow as a DCR Graph

Fig. 6 shows a workflow that is being used internally by Exformatics and has been modelled using DCR Graphs. It describes how to handle the process of receiving invoices.

The workflow contains five roles: 1) the administration department (Adm), which is responsible for receiving the invoice, scanning it and creating an invoice case . 2) The invoice responsible (Res), which is responsible for the invoice, usually because they are the person that bought the items that the invoice concerns, they are expected to check and approve the invoice. 3) The manager of the responsible (Man), whose approval may be needed in certain circumstances. 4) The CEO (CEO) who may also need to give approval in certain exceptional cases. And finally 5) the finance department (Fin), which takes care of paying the invoice and confirming that payment has succeeded. The tasks are divided into three phases, the Initial Phase which contains the tasks of the administration department, the Approval Phase which consists of the approval tasks and the Payment Phase which contains the tasks that handle the payment of the invoice.

The process starts when an invoice is received by the administration department, because Exformatics wants to keep all their documents in an electronic format it is required (through the response relation from Receive Invoice to Scan Invoice) that the invoice is scanned. The administration department is also required to decide if the invoice should be entered into the system (sometimes fake or wrong invoices are received which can be easily filtered out at first sight, for example because they are addressed to a non-existent employee). If they decide that the invoice appears legit then they enter all relevant data into the system, in particular the amount the invoice is for, which is used by the workflow system to determine whose approval is needed for the invoice. The responsible for the invoice should always approve the invoice (modelled by an unguarded response relation), if the amount of the invoice is higher then 1000 euros, approval from the responsible's manager is required as well (modelled by a response relation with the guard amount $\geq$ 1000). In special cases where the amount is higher then 20000 euros, approval from the CEO of the company is required as well.

It is possible that data is entered again, for example because a mistake was made by the administration department, or because a correction on the invoice was received, in this case new approvals will be required. When all necessary approvals have been received the invoice can be paid, this is modelled through the milestone relation from the Approval Phase to the task Pay Invoice, which means that Pay Invoice can not be done while there are pending responses in the Approval Phase. Once payment is confirmed, the invoice case should be closed, modelled through an exclusion relation from Confirm Payment to all three phases. There are five conditions in the workflow: first of all, Receive Invoice is required before the administration department can execute Enter Invoice Data or Scan Invoice. Enter Invoice Data is required before any approval can be given and all of the tasks in the Initial Phase should be done before any of the tasks in the Payment Phase can be done. Finaly, we have to pay the invoice before we can confirm payment.

4 Tool Support

Several tools have been developed at Exformatics to design and execute DCR Graphs internally or externally when presenting DCR Graphs at seminars or when interacting with customers. First of all, to facilitate the exchange of process descriptions between the tools developed by Exformatics and the tools being developed at IT University of Copenhagen, we defined a common XML format, which we will show in the first subsection. Secondly we developed a set of webservices that provide functionality for the execution, verification, storage and visualization of DCR Graphs, we named this set of services the *Process Engine*. Finally, as already mentioned above, we developed a standalone graphical editor to support the visual modelling and simulation of DCR Graphs, called the *DCR Graphs Editor*, which has also been used for teaching at a bachelor level course on Business Processes and IT at the IT University of Copenhagen.

Fig. 7 gives an overview of these tools and how they interact with eachother and the Exformatics ECM. The Process Engine is central to our tools and is used by the ECM to execute, verify and visualize workflows. The DCR Graphs Editor allows for execution of single steps by itself, but also uses the Process Engine for verification

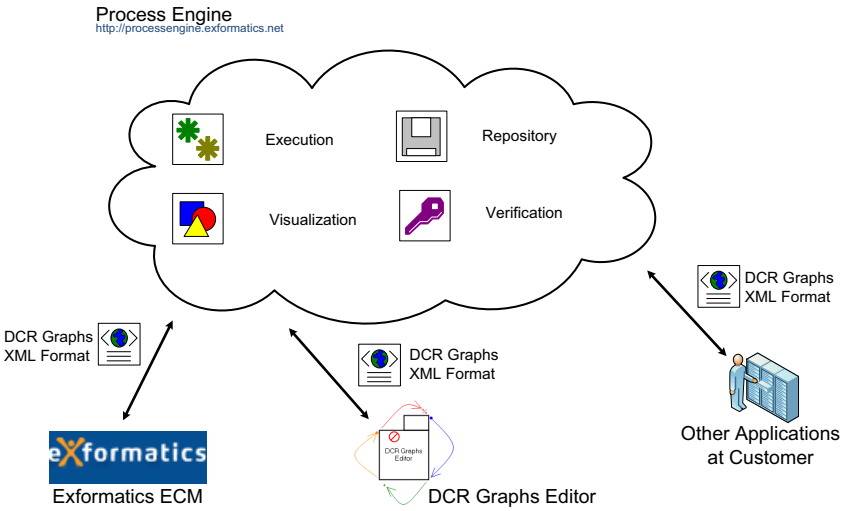


Fig. 7. Overview of the Exformatics DCR Graphs Tools

of DCR Graphs. Finally the purpose of the Process Engine is to be easily plugged in to other case management solutions as well, so that we may provide only workflow functionalities such as execution, verification, visualization and storage to customers without them being required to adopt the full Exformatics ECM package.

4.1 DCR Graphs XML Format

In listing 1 we give an example of the XML format for describing DCR Graphs.

The xml file consists of two main parts: the specification of the DCR Graph and the runtime state of the DCR Graph. The specification is split up into a section decribing resources and section describing constraints. The resource section contains subsections for events (possibly nested), labels, a mapping from labels to events, variables, expressions and variable acccess rights. The constraint section contains five subsections for the DCR Graph relations. The runtime section contains a subsection for the marking, containing the set of executed events, pending responses and included events, and a subsection for the state of the globalstore, which contains the values assigned to the variables in the current state.

Listing 1. Overview of DCR Graph XML Format

```
<?xml version = "1.0" encoding = "utf 8"?>
<dcrgraph>
    <specification>
        <resources>
            <events>
                <event id="Initial Phase">
                    <event id="Enter Invoice Data"/>
                    ...
                </event>
```

```
            ...
        </events>
        <labels>
            <label id="CEO Approval"/>
            ...
        </labels>
        <labelMappings>
            <labelMapping eventId="CEO Approval" labelId="CEO Approval"/>
            ...
        </labelMappings>
        <variables>
            <variable id="amount" value="0"/>
        </variables>
        <expressions>
            <expression id="gte1000" value="amount >= 1000"/>
            ...
        </expressions>
        <variableAccesses>
            <readAccesses>
                <readAccess eventId="Enter Invoice Data" variableId="amount"/>
                ...
            </readAccesses>
            <writeAccesses>
                <writeAccess eventId="Enter Invoice Data" variableId="amount"/
                    >
            </writeAccesses>
        </variableAccesses>
    </resources>
    <constraints>
        <conditions>
            <condition sourceId="Receive Invoice" targetId="Scan Invoice"/>
            ...
        </conditions>
        <responses>
            <response sourceId="Enter Invoice Data" targetId="Manager Approval
                " expressionId="gte1000"/>
            ...
        </responses>
        <excludes>
            <exclude sourceId="Confirm Payment" targetId="Approval Phase"/>
            ...
        </excludes>
        <includes/>
        <milestones>
            <milestone sourceId="Approval Phase" targetId="Pay Invoice"/>
        </milestones>
    </constraints>
</specification>
<runtime>
    <marking>
        <executed/>
        <included>
            <event id="Approval Phase"/>
            ...
        </included>
        <pendingResponses/>
    </marking>
    <globalStore>
        <variable id="amount" value="0"/>
    </globalStore>
</runtime>
</dcrgraph>
```

Next to the standard elements described above, it is possible to insert *custom* elements at all nodes of the XML tree. This allows one to add additional data for specific tools that is not required for the formal definition of a DCR Graph. Examples of these

are the roles (they are not a part of the formal model as they are not necesairily interesting for applications in other domains than BPM) and the location of events when drawn in the visual editor as shown in listing 2.

Listing 2. Example of how custom data can be insterted into the XML format

```
<?xml version="1.0" encoding="utf 8"?>
<event id="CEO Approval">
    <custom>
        <visualization>
            <location xLoc="449" yLoc="123"/>
        </visualization>
            <roles>
                <role>CEO</role>
        </roles>
    </custom>
</event>
```

4.2 Process Engine

Currently the Process Engine consists of three main webservices: the first for execution, the second for storage of DCR Graphs and the third for visualization of DCR Graphs. The execution service contains methods for executing and verifying DCR Graphs. The execution methods support the global data model, verification consists of checking for deadlock and livelock, but only for standard DCR Graphs without data. In the future we plan to extend the verification aspect and move it to its own service. The repository service for storage of DCR Graphs is currently very limited and mainly a proof of concept, it is planned to extend this in the future so it can be used to support sharing of workflows between cooperating organizations. The visualization service can be used to automatically layout and draw DCR Graphs, currently limited to the basic model without guards on data. All of these services are used by the Exformatics ECM for modelling and executing workflows.

4.3 DCR Graphs Editor

The DCR Graphs Editor is a graphical editor for modelling and simulating DCR Graphs. There are two main screens in the tool: in the Process Model screen one can design DCR Graphs by drawing events, changing the name, label and initial marking, adding roles and adding relations between events. In he Process Simulation Screen one can simulate DCR Graphs by clicking on the events that one wants to execute, the tool will give feedback on the current trace of executed events, which events can be executed and if the DCR Graph is in an accepting state. The tool can also interact with the verification methods of the Process Engine to check DCR Graphs for deadlock and livelock. It currently supports nested DCR Graphs including the milestone relation and work is underway to also add support for the global data model. All the images of DCR Graphs in this paper come directly from the editor.

5 Related Work

As mentioned in the introduction Declare [15, 19] was the first serious attempt at creating a declarative notation for describing business processes. Tool support for Declare

consists of a design tool, a server and corresponding user client for executing Declare processes. The designer is similar to the DCR Graphs Editor, allowing modellers to draw and verify Declare models (including a notion of data) by using a graphical user interface. The server is similar to the execution webservices contained in the Process Engine, allowing execution of Declare models by client programs. Finally the user client is somewhat comparable to the simulation part of the DCR Graphs Editor, although it offers more features to support the user in the execution of the process. These tools have been in development since the inception of the Declare language and therefor have seen a fair amount of iterations and reached a high level of maturity. The DCR Graphs tools on the other hand can be seen as being an advanced prototype version (with the most mature parts, such as the execution engine, currently being brought into production), where new features are still frequently being added. Both Declare and DCR Graphs are being included as extensions to the newest version of CPN Tools [20], for Declare it is the intention that this will become the main vehicle for further developments on the language and that no further features will be added to the previously mentioned tools. Declare also offers extensive support for analysis of Declare logs through ProM and support for process mining through the Declare Miner [11]. At the moment nothing comparable exists for DCR Graphs, however there is an interest in investigating process mining on running instances of DCR Graphs, particularly in the context of adaptive processes, with the goal of identifying common adaptation patterns. DCR Graphs also offer extended tool support for verification, allowing users to specify properties to be verified as a DCR Graph and then verifying processes modelled as DCR Graphs against these properties [13]. These tools are being developed at the IT University of Copenhagen and are therefor not described in detail this paper, however since these tools use the common XML format described in sec. 4.1, the Exformatics tools can easily interact with them.

The business artifacts [14] model developed by IBM Research combines both data and process aspects in a holistic manner. An artifact type contains both an information model (data for business objects) and a lifecycle model, which describes the possible ways a business entity might progress through and responds to events and external activities. A declarative approach using *Guard-Stage-Milestone* (GSM model) [9] based on ECA(Event Condition Action)-like rules for specification of life cycles on business artifacts has been developed in the recent years. Compared to DCR Graphs, the GSM-model has a richer support for data, but also a more complex semantics that does not capture acceptance criteria for infinite executions.

6 Evaluation

This work provides an initial report on tools being developed at Exformatics A/S examplified by a use-case being used internally within the company itself. As such no concrete quantitative evaluation of the usefulness and commercial viabilty of the tools exists yet. However, DCR Graphs as a modelling paradigm and the Exformatics tools themselves have already seen both commercial and academic use. As a modelling paradigm, DCR Graphs were applied in a commercial project involving Exformatics and *Landsorganisationen i Danmark (LO)*, the umbrella organisation for Danish

unions. During this project DCR Graphs were used to model the IT system that Exformatics developed for LO [1], but the lack of tool support for design and simulation limited its use. In [5] we showed how DCR Graphs can be used to model a distributed healthcare process encountered in a Danish hospital. DCR Graphs and the tools are currently employed in a project jointly with a danish research foundation for modelling the case management process for handling funding applications from submission to decission. All of these cases have been demonstrated for industry at seminars with positive feedback resulting in several requests for follow up meetings. Finally, Exformatics has recently started a commercial project for the Danish Cancer Society, including the development of an invoice approval solution based on the example used in this paper and using the Process Engine for execution of the workflows in the solution.

In the recent paper [17] we give the first empirical evaluation on what practitioners think of declarative modelling based on a study performed at a Dutch provider of ECM software. During the study some of those participating were presented Declare, while others were presented DCR Graphs. While the overall results of the study point in the direction of a hybrid model combining the imperative and declarative paradigms, it was also clear that the declarative paradigm by itself was percieved as useful for the right application domains.

In the Spring 2012 and 2013, the DCR Graphs model has been introduced in a bachelor course in IT and Business Process Modelling at the IT University of Copenhagen [2]. Each year, the course was followed by about 40 students, and the DCR Graph model was introduced for capturing process requirements, along with BPMN 2.0 for modelling processes imperatively. The students worked in groups, modelling their own processes identified in a field study performed in a previous course. They first modelled the process in BPMN and subsequently were asked to model the requirements in DCR Graphs and compare the models. They all experienced that the initial BPMN was good at describing a procedure of *how* to carry out the process. However, when turning to the DCR Graph model, they also realized that in most cases their BPMN model only described a fairly rigid, happy path through the process. In most cases it took the group two iterations to change their mindset to model requirements instead of the procedure. This may however be influenced by the fact, that they did no longer have access to the company in which they had performed the field study. Only in 2013, the DCR Graphs editor was available, and we experienced that it made it much easier for the students to learn the notation and semantics, and to appreciate its use for modelling process requirements. However, it was also clear that it still could be difficult for some of the students to visualize the possible paths of the process specified as DCR Graphs.

7 Conclusion

In this paper, we have given an informal introduction to DCR Graphs and briefly described current tool support, and how DCR Graphs and the tools are being used by Exformatics and in teaching at ITU university to model workflows.

Even though the uses in practice and teaching so far is limited, it has been very encouraging. At presentations for industry the models have generally been appreciated and easily understood. At the course the students were able to apply DCR Graphs to

model processes obtained from their own field studies in a previous course. They reported back that using the simulation facility in the tool was a great help to understand both the constraints of their own process and DCR Graphs as a model language.

As part of the future work, we plan to further develop the tools, making them more easily accessible and user-friendly to process modelers, based on the usability studies and feedback from students and clients of Exformatics. Furthermore, we also intend to upgrade the tools to support some of the latest extensions on DCR Graphs such as time [7], a distributed data model and more advanced verification techniques. Similarly, we are also working on extending the theory of DCR Graphs to provide a behavioral type system for cross-organizational workflows as initiated in [3]. In the future we also want to research the challenge of developing business processes for knowledge-intensive and adaptive case management processes as initiated in [13], which require more focus on evolutionary process data and adaptability of the process during execution.

References

1. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Designing a cross-organizational case management system using dynamic condition response graphs. In: 2011 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC), October 2-September 2, pp. 161–170 (2011)
2. Hildebrandt, T.: It and business process modelling course. IT University of Copenhagen (2013), https://blog.itu.dk/BIMF-F2013/
3. Hildebrandt, T., Carbone, M., Slaats, T.: Rsvp: Live sessions with responses. In: Proceedings of BEAT 2013, 1st International Workshop on Behavioural Types (2013)
4. Hildebrandt, T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: Post-Proceedings of PLACES 2010 (2010)
5. Hildebrandt, T., Mukkamala, R.R., Slaats, T.: Declarative modelling and safe distribution of healthcare workflows. In: International Symposium on Foundations of Health Information Engineering and Systems, Johannesburg, South Africa (August 2011)
6. 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)
7. 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
8. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: Honda, K., Mycroft, A. (eds.) PLACES. EPTCS, vol. 69, pp. 59–73 (2010)
9. 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, Springer, Heidelberg (2001)
10. Hull, R., Damaggio, E., Fournier, F., Gupta, M., Heath III, F.T., Hobson, S., Linehan, M., Maradugu, S., Nigam, A., Sukaviriya, P., Vaculin, R.: Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In: Proc. of WS-FM 2010, pp. 1–24. Springer, Heidelberg (2011)

11. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-Guided Discovery of Declarative Process Models. In: 2011 IEEE Symposium on Computational Intelligence and Data Mining, IEEE (2011)
12. Mukkamala, R.R.: A Formal Model For Declarative Workflows - Dynamic Condition Response Graphs. PhD thesis, IT University of Copenhagen (March 2012) (forthcomming)
13. 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 (2013)
14. Nigam, A., Caswell, N.S.: Business artifacts: An approach to operational specification. IBM Syst. J. 42, 428–445 (2003)
15. Pesic, M., Schonenberg, M.H., Sidorova, N., Van Der Aalst, W.M.P.: Constraint-based workflow models: change made easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
16. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
17. Reijers, H.A., Slaats, T., Stahl, C.: Declarative Modeling — An Academic Dream or the Future for BPM? In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 307–322. Springer, Heidelberg (2013)
18. Slaats, T.: Dcr graphs wiki. IT University of Copenhagen (2013), http://www.itu.dk/research/models/wiki/index.php/DCR_Graphs_Editor
19. 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)
20. 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)
21. Winskel, G.: Events in Computation. PhD thesis, Edinburgh University (1980)