Declarative Modelling and Safe Distribution of Healthcare Workflows

Thomas Hildebrandt[1], 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, 2100 Copenhagen, Denmark

Abstract. We present a formal technique for safe distribution of workflow processes described declaratively as nested Dynamic Condition Response (DCR) Graphs and apply the technique to a distributed healthcare workflow. Concretely, we provide a method to synthesize from a nested DCR Graph and any distribution of its atomic events a set of local process graphs communicating by shared events, such that the distributed execution of the local processes is equivalent to executing the original process. The technique extends our recent work on safe distribution of non-nested DCR Graphs applied to cross-organizational case management. The main contributions of the present paper is to adapt the technique to allow for nested processes and milestones and to apply it to a healthcare workflow identified in a previous field study at danish hospitals. We also provide a new formalization of the semantics of DCR Graphs which highlights its declarative nature.

1 Introduction

The overall goal of the interdisciplinary Trustworthy Pervasive Healthcare Services (TrustCare) project [16] is to develop a foundation for trustworthy it-supported healthcare workflows. Healthcare workflows involve coordination of a heterogeneous set of professionals, patients, organizations and sectors, and must be able to adapt to inevitable changes of treatment processes and organization of the work [23, 40]. This challenges traditional workflow management systems using an imperative process modeling language such as Business Process Model and Notation (BPMN) [35] in which the control flow is modeled explicitly. Typically a flow diagram will only cover the normal flow and a few possible exceptionally flows, leading to rigid and inflexible, over-specified workflows. Declarative process languages, allowing any flow that fulfill the specified constraints, have been suggested by a number of researchers as being more appropriate for representing workflow processes requiring a high degree of flexibility [8–10,33,43].

The TrustCare project combines research in pervasive user-interfaces [2, 3] with research in formal logic and domain specific process languages, taking as starting point [31]

* Authors listed alphabetically. This research is supported by the Danish Research Agency through the Trustworthy Pervasive Healthcare Services project (grant #2106-07-0019, www.trustcare.eu) and the Computer Supported Mobile Adaptive Business Processes project (grant #274-06-0415, www.cosmobiz.dk).

the declarative workflow process language developed and used by Resultmaker, the industrial partner of the project.

The present paper focus on and extends our work on formal process languages, in particular the development of a basic formal declarative workflow language called Dynamic Condition Response Graphs (DCR Graphs) introduced in [17] and extended to allow nested (i.e. hierarchical) process definitions in [19, 20] and a new milestone relation between activities. In [21] we have shown how, given a (non-nested) DCR Graph describing a global, collaborative process and any distribution of the activities, to derive a set of local DCR Graphs corresponding to the activity distribution and achieving the same global behavior by synchronously communicating the relevant events between the local processes. The main new contributions of the present paper is to adapt the distribution technique given in [21] to allow for nested processes and the new milestone relation between activities as introduced in [20] and demonstrate the use of the technique on an oncology healthcare workflow previously identified during a field study at danish hospitals [26]. The workflow was described loc. cit. using an early formalization of the Resultmaker workflow process language. Another contribution of the present paper is to formalize the workflow process as a nested DCR Graphs. Finally, we also provide a new presentation of the formal semantics of DCR Graphs that highlights the declarative nature. The present paper extends the results presented in the pre-proceedings of FHIES 2011 [22] which omitted the primitives of DCR Graphs for dynamically changing the set of included activities in the workflow.

The intention of the oncology healthcare workflow is to illustrate two important kinds of flexibility appearing in healthcare workflows: 1) The need for reconsidering a previous activity if its validity at a later stage is questioned by a co-worker and 2) The need for distribution of collaborative tasks and ability to tailor this distribution to local conditions (e.g. the size and organization of work within a hospital). These needs have also been identified during field studies of case management processes [19] and appears to be generally relevant for knowledge work processes and not only healthcare workflows.

The rest of the paper is structured as follows. In Sec. 1.1 ending the introduction we briefly discuss related work. We present the oncology workflow example in Sec. 2 as a nested Dynamic Condition Response (DCR) Graph, describing the semantics informally. In Sec. 3 we recall the formal definition of nested DCR Graphs and provide a new presentation of their semantics. We then in Sec. 4 formalize the distribution technique and exemplify it on the oncology workflow. Finally we conclude in Sec. 5.

1.1 Related Work

The problem of verifying the correctness of cross-organizational workflows described as variants of Petri nets has been studied in [1, 25, 27, 39, 41, 42, 46] and models of global behavior based on conversations among participating services have been studied in [4, 5, 14, 36, 48, 49]. A technique to partition a composite web service using program analysis was studied in [34] and using a similar approach, [24] explored decomposition of a business process modeled in BPEL, primarily focussing on P2P interactions . Using a formal approach based on I/O automata representing the services, the authors in [29] have studied the problem of synthesizing a decentralized choreography strategy, that

will have optimal overhead of service composition in terms of costs associated with each interaction.

The derivation of descriptions of local components from a global model has also been researched in the work on structured communication-centred programming for web services by Carbone, Honda and Yoshida [6]. The work formalizes the core of WS-CDL as the global process calculus and defines a formal theory of end-point projections projecting the global process calculus to abstract descriptions of the behavior of each of the local "end-points" given as pi-calculus processes typed with session types.

A methodology for deriving process descriptions from a business contract formalized in a formal contract language was studied in [28], while [38] proposes an approach to extract a distributed process model from collaborative business process. In [12, 13], the authors have proposed a technique for the flexible decentralization of a process specification with necessary synchronization between the processing entities using dependency tables. In [7,15,32] foundational work has been made on synthesizing distributed transition systems from global specification for the models of synchronous product and asynchronous automata [50].

The formalisms discussed above are all confined to imperative modeling languages such as Petri nets, workflow/open nets and automata based languages. To the best of our knowledge, there exists very few works on distributed cross-organizational workflows which consider declarative modeling languages and none where both the global and local processes are given declaratively using the same formalism. In [11], Fahland has studied synthesizing declarative workflows expressed in DecSerFlow [45] by translating to Petri nets. Only a predefined set of DecSerFlow constraints are used in the mapping to the Petri nets patterns, so this approach has a limitation with regards to the extensibility of the DecSerFlow language. On the other hand, in [30] Montali has studied the composition of ConDec [44] models with respect to conformance with a given choreography, based on the compatibility of the local ConDec models. But his study was limited to only composition of local models, whereas the problem of splitting a global model in local models has not been studied.

2 Distributed Declarative Healthcare Workflows by Example

In Fig. 1 below we show the graphical representation of the nested Dynamic Condition Response Graph formalizing a variant of the oncology workflow studied in [26]. In this section we informally describe the formalism and the distribution technique formalized in the rest of the paper using the example workflow. For details of the field study and the workflow we refer the reader to [26].

The boxes denote *activities* (also referred to as events in the following sections). Administer medicine is a *nested* activity having sub activities give medicine and trust. Give medicine is an *atomic* activity, i.e. it has no sub activities. Trust is again a nested activity having sub activities sign nurse 1 and sign nurse 2. Finaly, medicine preparation is a nested activity having seven sub activities dealing with the preparation of medicine. An activity may be either included or excluded, the latter are drawn as a dashed box as e.g. the edit and cancel activities.

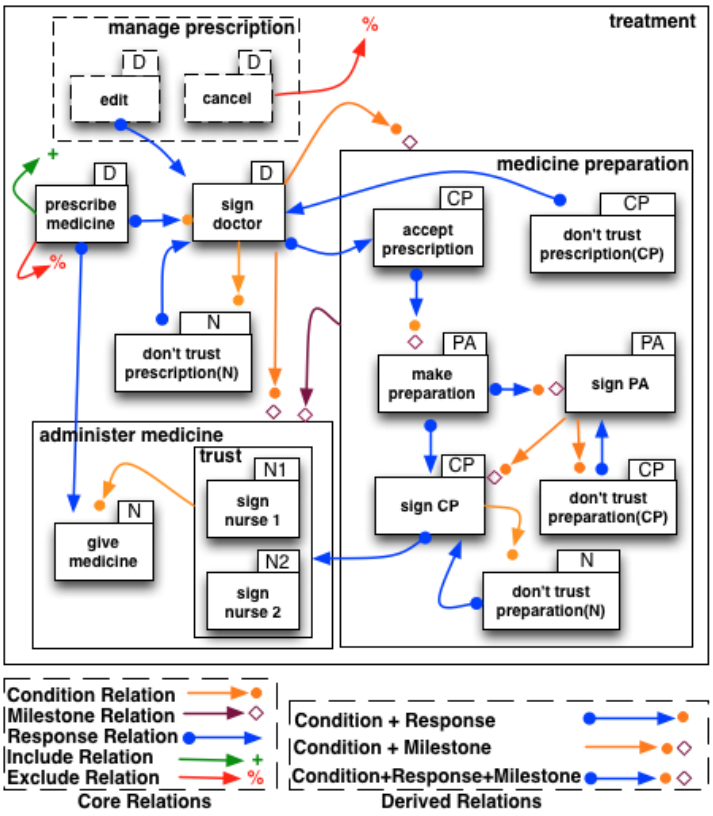


Fig. 1. Oncology Workflow as a nested DCR Graph

A *run* of the workflow consists of a (possibly infinite) sequence of executions of atomic activities. (A nested activity is considered executed when all its sub activities are executed). An activity can be executed any number of times during a run, as long as the activity is included and the constraints for executing it are satisfied, in which case we say the activity is *enabled*.

The constraints and dynamic exclusion and inclusion are expressed as five different core relations between activities represented as arrows in the figure above: The *condition relation*, the *response relation*, the *milestone relation*, the *include relation*, and the *exclude relation*.

The condition relation is represented by an orrange arrow with a bullet at the arrow head. E.g. the condition relation from the activity sign doctor to the activity don't trust prescription(N) means that sign doctor must have been executed at least once before the activity don't trust prescription(N) can be executed.

The response relation is represented by a blue arrow with a bullet at its source. E.g. the response relation from the activity prescribe medicine to the activity give medicine means that the latter must be executed (at some point) after (any execution

of) the activity prescribe medicine. We say that a workflow is in a *completed* state if all such response constraints have been fulfilled (or the required response activity is excluded). However, note that a workflow may be continued from a completed state and change to a non-completed state if an activity is executed that requires another response or includes an activity which has not been executed since it was last required as a response. Also note that the response constraint may cause some infinite runs to never pass through a complete state if the executed activities keep triggering new responses. In the following section we make precise when such infinite runs can be regarded as a complete execution.

The third core relation used in the example is the *milestone relation* represented as a dark red arrow with a diamond at the arrow head. The milestone relation was introduced in [20] jointly with the ability to nest activities. A relation to and/or from a nested activity simply unfolds to relations between all sub activities. A milestone relation from a nested activity to another activity then in particular means that the entire nested activity must be in a completed state before that activity can be executed. E.g. medicine preparation is a milestone for the activity administer medicine, which means that none of the sub activities of administer medicine can be carried out if any one of the sub activities of medicine preparation is included and has not been executed since it was required as a response.

Two activities can be related by any combination of these relations. In the graphical notation we have employed some shorthands, e.g. indicating the combination of a condition and a response relation by and arrow with a bullet in both ends.

Finally, DCR Graphs allow two relations for dynamic exclusion and dynamic inclusion of activities represented as a green arrow with a plus at the arrow head and a red arrow with a minus at the arrow head respectively. The exclusion relation is used in the example between the cancel activity and the treatment activity. Since all other activities in the workflow are sub activities of the treatment activity this means that all activities are excluded if the cancel activity is executed. The inclusion relation is used between the prescribe medicine activity and the manage prescription activity

The run-time state of a nested DCR Graph can be formally represented as a pair $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ of sets of atomic activities (referred to as the *marking* of the graph). The set Ex is the set of atomic activities that have been executed at least once during the run. The set Re is the set of atomic activities that, if included, are required to be executed at least one more time in the future as the result of a response constraint (i.e. they are pending responses). Finally, the set In denotes the currently included activities.

The set Ex thus may be regarded as a set of completed activities, the set Re as the set of activities on the to-do list and the set In as the activities that are currently relevant for the workflow.

Note that an activity may be completed once and still be on the to-do list, which simply means that it must be executed (completed) again. This makes it very simple to model the situation where an activity needs to be (re)considered as a response to the execution of an activity. In the oncology example this is e.g. the case for the response relation between the don't trust prescription(N) activity (representing that a nurse reports that he doesn't trust the prescription) and the sign doctor activity. The effect is that the doctor is asked to reconsider her signature on the prescription. In doing that

she may or may not decide to change the prescription, i.e. execute prescribe medicine again.

We indicate the marking graphically by adding a check mark to every atomic activity that has been executed (i.e. is included in the set Ex of the marking), an exclamation mark to every atomic activity which, if included, is required to be executed at least once more in the future (i.e. is included in the set Re), and making a box dashed if the activity is not included (i.e. is not included in the set In of the marking). In Fig. 1 we have shown an example marking where prescribe medicine has been executed. This has caused manage prescription and its sub activities edit and cancel to be included, and sign doctor and give medicine to be required as responses, i.e the two activities are included in the set Re of the marking (on the to-do list).

As described above, an activity can be executed if it is enabled. Sign doctor is enabled for execution in the example marking, since its only condition (prescribe medicine) has been executed and it has no milestones. Give medicine on the other hand is not enabled since it has the (nested) activity trust as condition, which means that all sub activities of trust (sign nurse 1 and sign nurse 2) must be executed before give medicine is enabled. Also, both give medicine and trust are sub activities of administer medicine which further has sign doctor as condition and milestone, and medicine preparation as milestone. The condition relation from sign doctor means that the prescription must be signed before the medicine can be administered. The milestone relations means that the medicine can not be given as long as sign doctor or any of the sub activities of medicine preparation is on the to-do list (i.e. in the set Re of pending responses).

Every activity should not be available to any user of the workflow system. For this reason the commercial implementation of the workflow management system provided by Resultmaker employs a role based access control, assigning to every atomic activity a finite set of roles and assigning to every role a set of access rights controlling if the activity is invisible or visible to users fulfilling the role. If an activity is visible it is specified wether the role are allowed to execute the activity or not. Users are either statically (e.g. by login) or dynamically assigned to roles (e.g. by email invitation).

In the formalization presented in this paper, the assigned roles are given as part of the name of the activity. In the graphical representation we have shown the roles within small ”ears” on the boxes. In the example workflow we have the following different roles: Doctor (D), Controlling Pharmacist (CP), Pharmacist Assistant (PA) and Nurse (N). Hereto comes roles N1 and N2 which must dynamically be assigned to two different authorized persons (nurses or doctors). This is at present the only way to implement the constraint stating that two different authorized persons must sign the product prepared by the pharmacists before the medicine is administered to the patient. Future work will address less ad hoc ways to handle these kind of constraints between activities referring to the identify of users.

The commercial implementation is based on a centralized workflow manager controlling the execution of the entire, global workflow. However, workflows often span different units or departments within the organization, e.g. the pharmacy and the patient areas, or even cross boundaries of different organizations (e.g. different hospitals). In some situations it may be very relevant to execute the local parts of the workflow on a

local (e.g. mobile) device without permanent access to a network, e.g. during preparation of the medicine in the pharmacy. Also, different organizations may want to keep control of their own parts of the workflow and not delegate the management to a central service. This motivates the ability to split the workflow in separate components, each only referring to the activities relevant for the local unit and being manageable independently of the other components.

The technique for distributing DCR Graphs introduced in [21] and extended in the present paper is a first step towards supporting this kind of splitting of workflow definitions. Given any division of activities on local units (assigning every activity to at least one unit) it describes how to derive a set of graphs, one for each unit, describing the local part of the workflow. Such a local process, referred to as a *projection* is again a DCR Graph. It includes the activities assigned to the unit but also the relevant *external* activities executed within other units for which an event must be send to the local process when they are executed. An example of a projection relative to the activities assigned the doctor role (D) is given in Fig. 2(a) in Sec. 4. The diagram shows that the projection also includes the two external activities (indicated as double line boxes) don't trust prescription (N) and don't trust prescription (CP). These two activities, representing respectively a nurse and a controlling pharmacist reporting that the prescription is not trusted, are the only external activities that may influence the workflow of the doctor by requiring sign doctor as a response. Similarly, Fig. 2(b),2(c), and 2(d) shows projections corresponding to the nurse, controlling pharmacist, and pharmacist assistant roles. However, if for instance the roles of the controlling pharmacist and the pharmacist assistant are always assigned to the same persons one may instead choose to keep all these activities together in a unit. This can be obtained by simply projecting on all activities assigned either the CP or the PA role.

3 Nested Dynamic Condition Response Graphs

Dynamic Condition Response Graphs (DCR Graphs) [17] is both a generalization of labelled event structures [47] and the Process Matrix workflow model developed by Resultmaker, our industrial partner in the TrustCare research project. The DCR Graphs were extended in [20] to Nested DCR Graphs, supporting sub graphs and a new milestone relation, motivated by a case study of cross-organizational case management [19]. Further in [21], we have considered safe distribution of DCR Graphs without milestone relation where as in [18], we have defined projection and distribution for a restricted nested model Condition Response Graphs, simplified by not allowing the dynamic inclusion and exclusion. In the present paper, we will consider full version of Nested DCR Graphs with both milestone and dynamic inclusion/exclusion relations and provide distribution of nested DCR Graphs. We employ the following notations in the paper.

Notation: For a set A we write $\mathcal{P}(A)$ for the power set of A. For a binary relation $\rightarrow \subseteq A \times A$ and a subset $\xi \subseteq A$ of A we write $\rightarrow \xi$ and $\xi \rightarrow$ for the set $\{a \in A \mid (\exists a' \in \xi \mid a \rightarrow a')\}$ and the set $\{a \in A \mid (\exists a' \in \xi \mid a' \rightarrow a)\}$ respectively. Also, we write $\rightarrow^{-1}$ for the inverse relation. Finally, for a natural number k we write $[k]$ for the set $\{1, 2, \ldots, k\}$.

We then formally define nested dynamic condition response graph as follows.

Definition 1. *A Dynamic Condition Response Graph (DCR Graph) G is a tuple* $(\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, where*

(i) E *is the set of* events *(or activities),*
(ii) $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \in \mathcal{M}(G)$ *is the* marking*, for* $\mathcal{M}(G) =_{def} \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E}) \times \mathcal{P}(\mathsf{E})$*,*
(iii) $\rightarrow\bullet \subseteq \mathsf{E} \times \mathsf{E}$ *is the* condition *relation,*
(iv) $\bullet\rightarrow \subseteq \mathsf{E} \times \mathsf{E}$ *is the* response *relation,*
(v) $\rightarrow\diamond \subseteq \mathsf{E} \times \mathsf{E}$ *is the* milestone *relation,*
(vi) $\rightarrow+, \rightarrow\% \subseteq \mathsf{E} \times \mathsf{E}$ *is the dynamic* include *relation and* exclude *relation, satisfying that* $\forall e \in \mathsf{E}. e \rightarrow+ \cap e \rightarrow\% = \emptyset$*,*
(vii) L *is the set of* labels*,*
(viii) $l : \mathsf{E} \rightarrow \mathcal{P}(\mathsf{L})$ *is a labeling function mapping events to sets of labels.*

A Nested Dynamic Condition Response Graph (Nested DCR Graph) G is a tuple $(\mathsf{E}, \triangleright, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, where*

(i) $(\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *is a DCR Graph,*
(ii) $\triangleright : \mathsf{E} \rightharpoonup \mathsf{E}$ *is a partial function mapping an event to its super-event (if defined),*
(iii) $\mathsf{M} \in \mathcal{P}(\mathsf{atoms}(\mathsf{E})) \times \mathcal{P}(\mathsf{atoms}(\mathsf{E})) \times \mathcal{P}(\mathsf{atoms}(\mathsf{E}))$*, where* $\mathsf{atoms}(\mathsf{E}) = \mathsf{E} \backslash \{e \in \mathsf{E} \mid \exists e' \in \mathsf{E}. \triangleright(e') = e\}$ *is the set of atomic events.*

We write $e \triangleright e'$ *if* $e' = \triangleright^k(e)$ *for* $0 < k$ *and write* $e \trianglerighteq e'$ *if* $e \triangleright e'$ *or* $e = e'$*, and* $e \trianglelefteq e'$ *if* $e' \triangleright e$ *or* $e = e'$*. We require that the resulting relation,*$\trianglerighteq \subset E \times E$*, referred to as the nesting relation, is a well founded partial order. We also require that the nesting relation is consistent with respect to dynamic inclusion/exclusion in the following sense: If* $e \triangleright e'$ *or* $e' \triangleright e$ *then* $e \rightarrow+ \cap e' \rightarrow\% = \emptyset$*.*

We already introduced the graphical notation for Nested DCR Graphs by example in the previous section. We will not write out the complete formal specification. The events are all boxes, e.g. $\mathsf{E} = \{\mathsf{treatment}, \mathsf{manage\ prescription}, \mathsf{prescribe\ medicine}, ...\}$, the nesting relation captures the inclusion of boxes, e.g. $\triangleright(e) = \mathsf{administer\ medicine}$, if $e \in \{\mathsf{give\ medicine}, \mathsf{trust}\}$ and $\triangleright(e) = \mathsf{trust}$, if $e \in \{\mathsf{sign\ nurse1}, \mathsf{sign\ nurse2}\}$ and so forth. The inital marking is the triple $M = (\emptyset, \emptyset, \mathsf{E} \backslash \{\mathsf{manage\ prescription}, \mathsf{edit}, \mathsf{cancel}\})$, meaning no events have been executed, no events are initially required as responses and all events except the events $\{\mathsf{manage\ prescription}, \mathsf{edit}, \mathsf{cancel}\}$ are included. The condition relation includes e.g. the pairs $\mathsf{sign\ doctor} \rightarrow\bullet\ \mathsf{don't\ trust\ prescription}(\mathsf{N})$ and $\mathsf{trust} \rightarrow\bullet\ \mathsf{give\ medicine}$, the response relation includes e.g. the pairs $\mathsf{prescribe\ medicine} \bullet\rightarrow\ \mathsf{sign\ doctor}$ and $\mathsf{edit} \bullet\rightarrow\ \mathsf{sign\ doctor}$, the milestone relation includes e.g. the pairs $\mathsf{sign\ doctor} \rightarrow\diamond\ \mathsf{administer\ medicine}$ and $\mathsf{sign\ PA} \rightarrow\diamond\ \mathsf{sign\ CP}$, the dynamic inclusion relation includes the single pair $\mathsf{prescribe\ medicine} \rightarrow+\ \mathsf{manage\ prescription}$ and the dynamic exclusion includes exactly the two pairs $\mathsf{prescribe\ medicine} \rightarrow\%\ \mathsf{prescribe\ medicine}$ and $\mathsf{cancel} \rightarrow\%\ \mathsf{treatment}$. We take as labels pairs of action names and roles, i.e. the set of labels L includes e.g. the pairs $(\mathsf{edit}, \mathsf{D})$, $(\mathsf{cancel}, \mathsf{D})$, $(\mathsf{give\ medicine}, \mathsf{N})$, and $(\mathsf{sign\ PA}, \mathsf{PA})$. Super events with no role assigned such as $\mathsf{manage\ prescription}$ are assigned the empty set of labels.

Note that the labels of events consist of the name of the event and a role which defines who can execute that event. In our implementation every event can be assigned

any number of roles and every user of the system can have multiple roles. A user can then execute an event if she has at least one role that is assigned to the event.

To define the execution semantics for Nested DCR Graphs we first define how to flatten a nested graph to the simpler DCR Graph. Essentially, all relations to and/or from nested events are extended to sub events, and then only the atomic events are preserved.

Definition 2. *For a Nested DCR Graph* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *define the underlying flat Dynamic Condition Response Graph as*

$$G^{\flat}_{l_f} = (\mathsf{atoms}(\mathsf{E}), \mathsf{M}, \rightarrow\bullet^{\flat}, \bullet\rightarrow^{\flat}, \rightarrow\diamond^{\flat}, \rightarrow+^{\flat}, \rightarrow\%^{\flat}, \mathsf{L}, l)$$

where $rel^{\flat} = \trianglerighteq rel \trianglelefteq$ *for some relation* $rel \in \{\rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%\}$.

It is easy to see from the definition that the underlying DCR Graph has at most as many events as the nested graph and that the size of the relations may increase by an order of n^2 where n is the number of atomic events.

Below we give a new presentation of the semantics of DCR Graphs stressing its declarative nature.

First we formalize in Def. 3 that an event e of a (flat) DCR Graph is enabled when it is included in current marking ($e \in \mathsf{In}$), all the included events that are conditions for it are in the set of executed events (i.e. $(\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}$) and none of the included events that are milestones for it are in the set of pending response events (i.e. $(\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$).

Definition 3. *For a Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, and* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *we define that an event* $e \in \mathsf{E}$ *is* enabled*, written* $G \vdash e$*, if* $e \in \mathsf{In}$*,* $(\mathsf{In} \cap \rightarrow\bullet e) \subseteq \mathsf{Ex}$ *and* $(\mathsf{In} \cap \rightarrow\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$*.*

Def. *4* below then defines the change of the marking when an enabled event is executed: First the event is added to the set of executed events and removed from the set of pending responses. Then all events that are a response to the event are added to the set of pending responses. Note that if an event is a response to itself, it will remain in the set of pending responses after execution. Similarly, the included events set will updated by adding all the events that are included by the event and by removing all the events that are excluded by the event.

Definition 4. *For a Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, where* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *and an enabled event* $G \vdash e$ *, the result of executing* e *is a Dynamic Condition Response Graph* $G = (\mathsf{E}, \mathsf{M}', \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, where* $\mathsf{M}' = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_G e =_{def} \big(\mathsf{Ex} \cup \{e\}, (\mathsf{Re} \backslash \{e\}) \cup e\bullet\rightarrow, (\mathsf{In} \cup e\rightarrow+) \setminus e\rightarrow\%\big)$.

We now define the semantics for Nested DCR Graph by using the corresponding flat graph.

Definition 5. *For a Nested Condition Response Graph* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$*, where* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *we define that* $e \in \mathsf{atoms}(\mathsf{E})$ *is enabled, written* $G \vdash e$*, if* $G^{\flat} \vdash e$*. Similarly, the result of executing* $G \vdash e$ *is defined as:* $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In}) \oplus_{G^{\flat}} e$*.*

As an example, in the intial marking $M = (\emptyset, \emptyset, \mathsf{E}\backslash\{\mathsf{manage\ prescription}, \mathsf{edit}, \mathsf{cancel}\})$ we have that $G \vdash$ prescribe medicine, i.e. the event prescribe medicine is enabled.

After executing prescribe medicine the new marking $M' = M \oplus_G$ prescribe medicine $= (\{\mathsf{prescribe\ medicine}\}, \{\mathsf{sign\ doctor}, \mathsf{give\ medicine}\}, \mathsf{E}\backslash\{\mathsf{prescribe\ medicine}\})$. That is, prescribe medicine is added to the set of executed events, and sign doctor and give medicine are added to the set of pending responses, because prescribe medicine $\bullet\!\!\rightarrow$ sign doctor and prescribe medicine $\bullet\!\!\rightarrow$ give medicine}. The event prescribe medicine is removed from the set of included events because prescribe medicine $\rightarrow\!\%$ prescribe medicine. The events {manage prescription, edit, cancel} are included since prescribe medicine $\rightarrow\!+$ manage prescription, and the inclusion relation is "flattened" to include also prescribe medicine $\rightarrow\!+$ edit and prescribe medicine $\rightarrow\!+$ cancel.

From the definition of enabling and execution above we can construct a labelled transition semantics for a nested DCR Graphs, with acceptance conditions for finite and infinite computations.

Definition 6. *For a nested DCR Graph* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\!\bullet, \bullet\!\!\rightarrow, \rightarrow\!\diamond, \rightarrow\!+, \rightarrow\!\%, \mathsf{L}, l)$ *we write* $G \xrightarrow{(e,a)} G'$ *if* $G' = (\mathsf{E}, \rhd, \mathsf{M}', \rightarrow\!\bullet, \bullet\!\!\rightarrow, \rightarrow\!\diamond, \rightarrow\!+, \rightarrow\!\%, \mathsf{L}, l)$*,* $G \vdash e$*,* $a \in l(e)$*, and* $\mathsf{M}' = \mathsf{M} \oplus_{G^\flat} e$*. We then define the corresponding labelled transition system* $TS(G)$ *to be the tuple* $(\mathcal{G}_{\mathsf{E}\times\mathsf{L}}, G, \mathsf{E} \times \mathsf{L}, \rightarrow \subseteq \mathcal{G}_{\mathsf{E}\times\mathsf{L}} \times \mathsf{E} \times \mathsf{L} \times \mathcal{G}_{\mathsf{E}\times\mathsf{L}})$ *where* $\mathcal{G}_{\mathsf{E}\times\mathsf{L}}$ *is the set of all nested DCR Graphs with events in* E *and labels in* L*.*

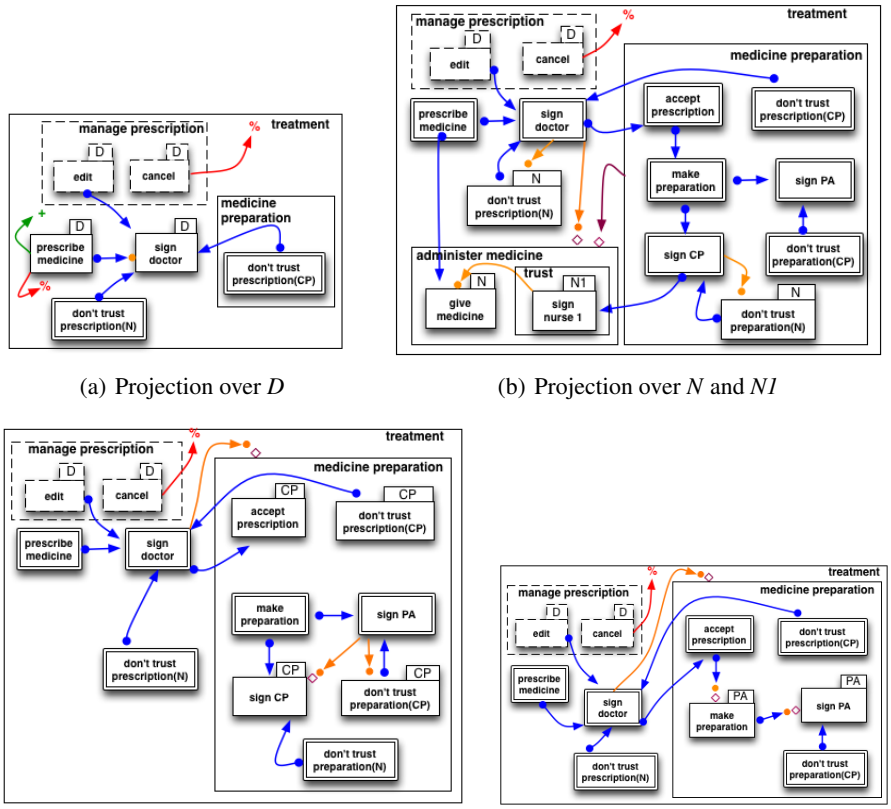
We define a run $a_0, a_1, \ldots$ *of a nested DCR Graph* G *to be a sequence of labels of a sequence of transitions* $G_i \xrightarrow{(e_i,a_i)} G_{i+1}$*. starting from* $G_0 = G$*. Assuming the marking of* G_i *is* $(\mathsf{Ex}_i, \mathsf{Re}_i, \mathsf{In}_i)$*, define a run to be accepting if for the underlying sequence of transitions it holds that* $\forall i \geq 0, e \in \mathsf{Re}_i . \exists j \geq i . (e = e_j \vee e \notin \mathsf{In}_j)$*. In words, a run is accepting if every required response event happens at some later stage or become excluded.*

4 Projections and Distributed Execution

In Sec. 4.1 below we define the notion of projection of a nested DCR Graphs, restricting the graph to a subset of the events, and in Sec. 4.2 we define the technique for distributing a nested DCR Graph as a set of local nested DCR Graphs obtained as projections and communicating by notifications of event executions.

4.1 Projections

A nested DCR Graph G is projected with respect to a *projection parameter* $\delta = (\delta_\mathsf{E}, \delta_\mathsf{L})$, where $\delta_\mathsf{E} \subseteq \mathsf{E}$ is a subset of the events of G satisfying that $\rhd(\delta_\mathsf{E}) \subseteq \delta_\mathsf{E}$, i.e. the subset is closed under the super event relation, and $\delta_\mathsf{L} \subseteq \mathsf{L}$ is a subset of the labels. The intuition is that the graph is restricted to only those events and relations that are relevant for the execution of events in δ_E and the labeling is restricted to the set δ_L. The technical difficulty is to infer the events and relations not in δ_E, referred to as *external events* below, that should be included in the projection because they influence the execution of the workflow restricted to the events in δ_E.



(a) Projection over *D*

(b) Projection over *N* and *N1*

(c) Projection over *CP*

(d) Projection over *PA*

Fig. 2. Oncology workflow projected with respect to different roles

Fig. 2 shows examples of projections of the oncology workflow with respect to different roles. For instance, Fig. 2(a) shows the projection with respect to the projection parameter (δ_E, δ_L) where δ_E={manage prescription, edit, cancel, prescribe medicine, sign doctor} and δ_L={(edit, D), (cancel, D), (prescribe medicine, D), (sign doctor, D)}. The two events don't trust prescription (N) and don't trust prescription (CP) shown with double line borders are external events included in the projected graph even though they don't appear in the projection parameter. It is interesting to note that the doctor only needs to be aware of these two activities carried out by other participants. In comparison, the projection over the roles for nurses (*N* and *N1*) contains all the events since they may influence (because of the milestone relations) the execution of the events with roles *N* and *N1*. In other words, the doctors can carry out workflows highly independent of the other activities while the nurses are dependent on any event carried out by the other roles.

The formal definition of projection for nested DCR Graphs is given in 7 below. It generalizes the definition of projection introduced in [21] for DCR Graphs to support nesting and milestones.

Definition 7. *If* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *then* $G_{|\delta} = (\mathsf{E}_{|\delta}, \rhd_{|\delta}, \mathsf{M}_{|\delta}, \rightarrow\bullet_{|\delta}, \bullet\rightarrow_{|\delta}, \rightarrow\diamond_{|\delta}, \rightarrow+_{|\delta}, \rightarrow\%_{|\delta}, \delta_{\mathsf{L}}, l_{|\delta})$ *is the projection of* G *with respect to* $\delta \subseteq \mathsf{E}$ *where:*

(i) $\mathsf{E}_{|\delta} = \rightarrow\delta_{\mathsf{E}}$, *for* $\rightarrow = \bigcup_{c \in C} c$, *and* $C = \{\mathsf{id}, \rightarrow\bullet^{\flat}, \bullet\rightarrow^{\flat}, \rightarrow\diamond^{\flat}, \rightarrow+^{\flat}, \rightarrow\%^{\flat}, \bullet\rightarrow^{\flat}\rightarrow\diamond^{\flat}, \rightarrow+^{\flat}\rightarrow\bullet^{\flat}, \rightarrow\%^{\flat}\rightarrow\bullet^{\flat}, \rightarrow+^{\flat}\rightarrow\diamond^{\flat}, \rightarrow\%^{\flat}\rightarrow\diamond^{\flat}\}$

(ii) $\rhd_{|\delta}(e) = \rhd(e)$, *if* $e \in \mathsf{E}_{|\delta}$

(iii) $l_{|\delta}(e) = \begin{cases} l(e) \cap \delta_{\mathsf{L}} & \text{if } e \in \delta_{\mathsf{E}} \\ \emptyset & \text{if } e \in \mathsf{E}_{|\delta} \backslash \delta_{\mathsf{E}} \end{cases}$

(iv) $\mathsf{M}_{|\delta} = (\mathsf{Ex}_{|\delta}, \mathsf{Re}_{|\delta}, \mathsf{In}_{|\delta})$ *where:*

(a) $\mathsf{Ex}_{|\delta} = \mathsf{Ex} \cap \mathsf{E}_{|\delta}$

(b) $\mathsf{Re}_{|\delta} = \mathsf{Re} \cap (\delta_{\mathsf{E}} \cup \rightarrow\diamond^{\flat}\delta_{\mathsf{E}})$

(c) $\mathsf{In}_{|\delta} = \mathsf{In} \cap \mathsf{E}_{|\delta}$

(v) $\rightarrow\bullet_{|\delta} = \rightarrow\bullet \cap ((\rightarrow\bullet^{\flat}\delta_{\mathsf{E}}) \times \delta_{\mathsf{E}})$

(vi) $\bullet\rightarrow_{|\delta} = \bullet\rightarrow \cap ((\bullet\rightarrow^{\flat}\rightarrow\diamond^{\flat}\delta_{\mathsf{E}}) \times (\rightarrow\diamond^{\flat}\delta_{\mathsf{E}})) \cup ((\bullet\rightarrow^{\flat}\delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}))$

(vii) $\rightarrow\diamond_{|\delta} = \rightarrow\diamond \cap ((\rightarrow\diamond^{\flat}\delta_{\mathsf{E}}) \times \delta_{\mathsf{E}})$

(viii) $\rightarrow+_{|\delta} = \rightarrow+ \cap \Big(\big((\rightarrow+^{\flat}\delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}\big) \cup \big((\rightarrow+^{\flat}\rightarrow\bullet^{\flat}\delta_{\mathsf{E}}) \times (\rightarrow\bullet^{\flat}\delta_{\mathsf{E}}\big) \cup \big((\rightarrow+^{\flat}\rightarrow\diamond^{\flat}\delta_{\mathsf{E}}) \times (\rightarrow\diamond^{\flat}\delta_{\mathsf{E}})\big) \Big)$

(ix) $\rightarrow\%_{|\delta} = \rightarrow\% \cap \Big(\big((\rightarrow\%^{\flat}\delta_{\mathsf{E}}) \times \delta_{\mathsf{E}}\big) \cup \big((\rightarrow\%^{\flat}\rightarrow\bullet^{\flat}\delta_{\mathsf{E}}) \times (\rightarrow\bullet^{\flat}\delta_{\mathsf{E}}\big) \cup \big((\rightarrow\%^{\flat}\rightarrow\diamond^{\flat}\delta_{\mathsf{E}}) \times (\rightarrow\diamond^{\flat}\delta_{\mathsf{E}})\big) \Big)$

(i) defines the set of events in the projection as all events that has a relation pointing to an event in the set δ_{E}, where the relation is either the identity relation (i.e. it is an event in δ_{E}), one of the core relations (flattened) or the relations such as $\bullet\rightarrow^{\flat}\rightarrow\diamond^{\flat}$ which includes all events that triggers as a response some event that is a milestone to an event in δ_{E} or the relations that include/exclude conditions and milestones to an event in the set δ_{E}.

Events in $\mathsf{E}_{|\delta} \backslash \delta_{\mathsf{E}}$ are referred to as external events and will be included in the projection without labels, as can be seen from the definition of the labeling function in *(iii)*. As we will formalize below, events without labels can not be executed by a user locally. However, when composed in a network containing other processes that can execute these events, their execution will be communicated to the process.

(iv) defines the projection of the marking: The executed and included event sets are simply restricted to the events in $\mathsf{E}_{|\delta}$. The responses are restricted to events in δ_{E} and events that have a milestone relation to an event in δ_{E} because these are the only responses that will affect the local execution of the projected graph. Note that these events will by definition be events in $\mathsf{E}_{|\delta}$ but may be external events.

Finally, *(v)* - *(ix)* state which relations should be included in the projection. For the events in δ_{E} all incoming relations should be included. Additionally response relations to events that are a milestone for an event in δ_{E} are included as well.

To define networks of communicating nested DCR Graphs and their semantics we use the following extension of a nested DCR Graph adding a new label to every event.

Definition 8. *For an DCR Graph* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *define* $G^{\varepsilon} = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L} \cup \{\varepsilon\}, l^{\varepsilon})$*, where* $l^{\varepsilon} = l(e) \cup \{\varepsilon\}$ *(assuming that* $\varepsilon \notin \mathsf{L}$*).*

We are now ready to state the key correspondence between global execution of events and the local execution of events in a projection.

Proposition 1. *Let* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *be a nested DCR Graph and* $G_{|\delta}$ *its projection with respect to a projection parameter* $\delta = (\delta_{\mathsf{E}}, \delta_{\mathsf{L}})$*. Then*

1. *for* $e \in \delta_{\mathsf{E}}$ *and* $a \in \delta_{\mathsf{L}}$ *it holds that* $G \xrightarrow{(e,a)} G'$ *if and only if* $G_{|\delta} \xrightarrow{(e,a)} G'_{|\delta}$*,*
2. *for* $e \notin \mathsf{E}_{|\delta}$ *it holds that* $G \xrightarrow{(e,a)} G'$ *implies* $G_{|\delta} = G'_{|\delta}$*,*
3. *for* $e \in \mathsf{E}_{|\delta}$ *it holds that* $G \xrightarrow{(e,a)} G'$ *implies* $(G_{|\delta})^{\varepsilon} \xrightarrow{(e,\varepsilon)} (G'_{|\delta})^{\varepsilon}$*,*

4.2 Distributed Execution

We are now ready to define networks of nested DCR Graphs and give the main technical theorem of the paper stating that a network of nested DCR Graphs obtained by projecting a nested DCR Graph G with respect to a covering vector of projection parameters has the same behavior as the original graph G.

Intuitively, a vector of projection parameters is covering if every event is included in at least one projection parameter and every label that is assigned to an event occurs at least once together with that event.

Definition 9. *We call a vector* $\Delta = (\delta_1, \ldots, \delta_k)$ *of projection parameters* covering *for some DCR Graph* $G = (\mathsf{E}, \rhd, \mathsf{M}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow\diamond, \rightarrow+, \rightarrow\%, \mathsf{L}, l)$ *if:*

1. $\bigcup_{i \in [k]} \delta_{\mathsf{E}i} = \mathsf{E}$ *and*
2. $(\forall a \in \mathsf{L}. \forall e \in \mathsf{E}. a \in l(e) \Rightarrow (\exists i \in [k]. e \in \delta_{\mathsf{E}i} \wedge a \in \delta_{\mathsf{L}i})$

Definition 10. *We define a* network of DCR Graphs N *by the grammar*

$$N := G \mid N \| N$$

and let $\mathcal{N}_{\mathsf{E} \times \mathsf{L}}$ *be the set of all networks with events in* E *and labels in* L*.*

We write $\Pi_{i \in [n]} G_i$ *for* $G_1 \| G_2 \| \ldots \| G_n$*. We define the set of events of a network of graphs inductively by* $\mathcal{E}(G) = \mathsf{E}$ *and* $\mathcal{E}(N_1 \| N_2) = \mathcal{E}(N_1) \cup \mathcal{E}(N_2)$*. Similarly, we define the set of labels of a network of graphs inductively by* $\mathcal{L}(G) = \mathsf{L}$ *and* $\mathcal{L}(N_1 \| N_2) = \mathcal{L}(N_1) \cup \mathcal{L}(N_2)$*.*

Definition 11. *The transition semantics of networks of DCR Graphs are given by the following inference rules:*

input $$\frac{G_1^{\varepsilon} \xrightarrow{(e,\varepsilon)} G_2^{\varepsilon}}{G_1 \overset{\triangleright e}{\rightsquigarrow} G_2}$$

sync input $$\frac{N_1 \overset{\triangleright e}{\rightsquigarrow} N_1' \quad N_2 \overset{\triangleright e}{\rightsquigarrow} N_2'}{N_1 \| N_2 \overset{\triangleright e}{\rightsquigarrow} N_1' \| N_2'}$$

local input $$\frac{N_i \overset{\triangleright e}{\rightsquigarrow} N_i' \quad e \notin \mathcal{E}(N_{1-i})}{N_0 \| N_1 \overset{\triangleright e}{\rightsquigarrow} N_0' \| N_1} \quad N_{1-i} = N_{1-i}', i \in \{0,1\}$$

sync step $$\frac{N_i \xrightarrow{(e,a)} N_i' \quad N_{1-i} \overset{\triangleright e}{\rightsquigarrow} N_{1-i}'}{N_0 \| N_1 \xrightarrow{(e,a)} N_0' \| N_1'} \; i \in \{0,1\}$$

local step $$\frac{N_i \xrightarrow{(e,a)} N_i' \quad e \notin \mathcal{E}(N_{i-1})}{N_0 \| N_1 \xrightarrow{(e,a)} N_0' \| N_1} \quad N_{1-i} = N_{1-i}', i \in \{0,1\}$$

For a network of nested DCR Graphs N we define the corresponding transition system $TS(N)$ by $(\mathcal{N}_{\mathcal{EL}(N)}, N, \mathcal{EL}(N), \rightarrow \subseteq \mathcal{N}_{\mathcal{EL}(N)} \times \mathcal{EL}(N) \times \mathcal{N}_{\mathcal{EL}(N)})$ where $\mathcal{EL}(N) = \mathcal{E}(N) \times \mathcal{L}(N)$ and the transition relation $\rightarrow \subseteq \mathcal{N}_{\mathcal{EL}(N)} \times \mathcal{EL}(N) \times \mathcal{N}_{\mathcal{EL}(N)}$ is defined by the inference rules above.

We define a run $a_0, a_1, \ldots$ of the transition system to be a sequence of labels of a sequence of transitions $N_i \xrightarrow{(e_i,a_i)} N_{i+1}$ starting from the initial network. We define a run for a network $N = \Pi_{i \in [n]} G_i$ to be accepting if for the underlying sequence of transitions it holds that $\forall j \in [n], \forall i \geq 0, e \in \mathsf{Re}_{j,i}. \exists k \geq i. (e = e_k \vee e \notin \mathsf{In}_k)$, where $\mathsf{Re}_{j,i}$ is the set of required responses in the jth nested DCR Graph in the network in the ith step of the run. In words, a run is accepting if every response event in a local nested DCR Graph in the network happens at some later state or become excluded.

Thm. 1 below now states the correspondence between a nested DCR Graph and the network of nested DCR Graphs obtained from a covering projection.

Theorem 1. *For a nested DCR Graph G and a covering vector of projection parameters $\Delta = (\delta_1, \ldots, \delta_n)$ it holds that $TS(G)$ is bisimilar to $TS(G_\Delta)$, where $G_\Delta = \Pi_{i \in [n]} G_{|\delta_i}$. Moreover, a run is accepting in $TS(G)$ if and only if the bisimilar run is accepting in $TS(G_\Delta)$.*

The generality of the distribution technique given above allows for fine tuned projections where we select only a few events for a specific role and actor, but in most cases the parameter is likely to be chosen so that the projected graph shows the full responsibilities of a specific role or actor. A set of nested DCR Graphs can be maintained and executed in a distributed fashion, meaning that there is a separate implementation for every graph and that the execution of shared events is communicated between them. Through the distributed execution of projected graphs, nested DCR Graphs can be used as a (declarative) choreography model to the line of work (on typed imperative process

models) in [6]: The original graph can be seen as the choreography, describing how the system as a whole should function, from which we project multiple end-points for individual roles or actors that can be implemented independently.

5 Conclusion and Future Work

We have presented a formal technique for safe distribution of collaborative, cross-organizational workflows modeled declaratively in the model of Nested Dynamic Condition Response (Nested DCR) Graphs [17, 20]. The key difference between the present work and the related work surveyed in Sec. 1.1 is that Nested DCR Graphs is a declarative model while most previous work has focussed on imperative models. We have argued for the use of a declarative approach for flexible workflows of knowledge workers and exemplified the techniques on a small workflow identified during a previous field study at danish hospitals [26]. The example is not supposed to demonstrate completeness of the technique but to capture two common examples of flexibility, namely the need to reconsider previous activities and the flexibility to distribute the execution of a workflow across different units within or across organizations.

The distribution technique presented is an extension of a method developed recently for flat DCR Graphs [21] to allow for nesting of events and the co-called milestone relations. This again allows us to apply the technique to the oncology workflow which we believe is an important new contribution in order to communicate the ideas better to people working within the healthcare domain.

A number of interesting questions are left for future work. We have implemented a prototype tool for design, simulation and verification (model checking via the SPIN and ZING model checkers) of DCR Graphs as reported on in [19]. These tools should be extended to nested graphs and the distribution technique should be implemented. This then leads to considering what can be achieved by performing verification of local components individually. We also aim to investigate how to support dynamic changes to local components, using the underlying idea of the distribution technique to determine what should be changed in other components when a local component is changed. Finally we are working on extending the model to allow for data and time to be represented and developing a prototype implementation integrated with the work on pervasive user interfaces carried out in the other track of the TrustCare project. This would allow us to carry out a larger demonstration project jointly with a hospital evaluating both the workflow modeling and the pervasive user interfaces. Along the same lines, it would be interesting to relate our work to the approach in the OpenKnowledge and Safe & Sound projects based on the Lightweight Coordination Calculus (LCC) [37].

Acknowledgements: We are grateful to the anonymous reviewers for valuable feedback.

References

1. Wil M. P. van der Aalst and Mathias Weske. The p2p approach to interorganizational workflows. In *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, CAiSE '01, pages 140–156, 2001.

2. Jakob E. Bardram, Jonathan Bunde-Pedersen, Afsaneh Doryab, and Steffen Sorensen. Clinical surfaces: Activity-based computing support for distributed multi-display environments inside hospitals. In *INTERACT 2009: 12th IFIP TC13 Conference in Human-Computer Interaction*, Uppsala, Sweden, 2009.
3. Jakob E. Bardram, Jonathan Bunde-Pedersen, and Mads Soegaard. Support for activity-based computing in a personal computing operating system. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 211–220, New York, NY, USA, 2006. ACM Press.
4. Mario Bravetti and Gianluigi Zavattaro. Contract based multi-party service composition. In *International Symposium on Fundamentals of Software Engineering (FSEN)*, volume 4767, pages 207–222. Springer, 2007.
5. Mario Bravetti and Gianluigi Zavattaro. A theory of contracts for strong service compliance. *Mathematical. Structures in Comp. Sci.*, 19:601–638, June 2009.
6. Marco Carbone, Kohei Honda, and Nobuko Yoshida. Structured Communication-Centred Programming for Web Services. In *16th European Symposium on Programming (ESOP'07)*, LNCS, pages 2–17. Springer, 2007.
7. Ilaria Castellani, Madhavan Mukund, and P. Thiagarajan. Synthesizing distributed transition systems from global specifications. In *Foundations of Software Technology and Theoretical Computer Science*, volume 1738, pages 219–231. Springer Berlin / Heidelberg, 1999.
8. Federico Chesani, Pietro De Matteis, Paola Mello, Marco Montali, and Sergio Storari. A framework for defining and verifying clinical guidelines: A case study on cancer screening. In Floriana Esposito, Zbigniew Ras, Donato Malerba, and Giovanni Semeraro, editors, *Foundations of Intelligent Systems*, volume 4203 of *Lecture Notes in Computer Science*, pages 338–343. Springer Berlin / Heidelberg, 2006.
9. Federico Chesani, E. Lamma, P. Mello, M. Montali, S. Storari, P. Baldazzi, and M. Manfredi. *Compliance Checking of Cancer-Screening Careflows: an Approach Based on Computational Logic*, chapter Computer- Based Medical Guidelines and Protocols: a Primer and Current Trends. IOS Press, 2008.
10. Federico Chesani, Paola Mello, Marco Montali, and Sergio Storari. Testing careflow process execution conformance by translating a graphical language to computational logic. In Riccardo Bellazzi, Ameen Abu-Hanna, and Jim Hunter, editors, *Artificial Intelligence in Medicine*, volume 4594 of *Lecture Notes in Computer Science*, pages 479–488. Springer Berlin / Heidelberg, 2007.
11. Dirk Fahland. Towards analyzing declarative workflows. In *Autonomous and Adaptive Web Services*, 2007.
12. Walid Fdhila and Claude Godart. Toward synchronization between decentralized orchestrations of composite web services. In *CollaborateCom'09*, pages 1–10, 2009.
13. Walid Fdhila, Ustun Yildiz, and Claude Godart. A flexible approach for automatic process decentralization using dependency tables. *International Conference on Web Services*, 2009.
14. Xiang Fu, Tevfik Bultan, and Jianwen Su. Realizability of conversation protocols with message contents. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '04, pages 96–, Washington, DC, USA, 2004. IEEE Computer Society.
15. Keijo Heljanko and Alin Stefanescu. Complexity results for checking distributed implementability. In *Proceedings of the Fifth International Conference on Application of Concurrency to System Design*, pages 78–87, 2005.
16. Thomas Hildebrandt. Trustworthy pervasive healthcare processes (TrustCare) research project. Webpage, 2008. `http://www.trustcare.dk/`.
17. Thomas Hildebrandt and Raghava Rao Mukkamala. Declarative event-based workflow as distributed dynamic condition response graphs. In *Post proceedings of International Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES 10)*, 2011.

18. Thomas Hildebrandt, Raghava Rao Mukkamala, and Tijs Slaats. Declarative modelling and safe distribution of healthcare workflows. In *International Symposium on Foundations of Health Information Engineering and Systems*, Johannesburg, South Africa, August 2011.
19. Thomas Hildebrandt, Raghava Rao Mukkamala, and Tijs Slaats. Designing a cross-organizational case management system using dynamic condition response graphs. In *Proceedings of IEEE International EDOC Conference*, 2011. to appear.
20. Thomas Hildebrandt, Raghava Rao Mukkamala, and Tijs Slaats. Nested dynamic condition response graphs. In *Proceedings of Fundamentals of Software Engineering (FSEN)*, April 2011. to appear.
21. Thomas Hildebrandt, Raghava Rao Mukkamala, and Tijs Slaats. Safe distribution of declarative processes. In *9th International Conference on Software Engineering and Formal Methods (SEFM) 2011*, 2011. to appear.
22. International symposium on foundations of health information engineering and systems, August 2011.
23. Ali Rahmanzadeh John Fox, Nicky Johns. Disseminating medical knowledge: the proforma approach. *Artificial intelligence in medicine*, 14:157–182, September 1998.
24. R. Khalaf and F. Leymann. Role-based decomposition of business processes using BPEL. In *Web Services, 2006. ICWS '06. International Conference on*, pages 770 –780, sept. 2006.
25. Ekkart Kindler, Axel Martens, and Wolfgang Reisig. Inter-operability of workflow applications: Local criteria for global soundness. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 235–253, London, UK, 2000. Springer-Verlag.
26. Karen Marie Lyng, Thomas Hildebrandt, and Raghava Rao Mukkamala. From paper based clinical practice guidelines to declarative workflow management. In *Proceedings of 2nd International Workshop on Process-oriented information systems in healthcare (ProHealth 08)*, pages 36–43, Milan, Italy, 2008. BPM 2008 Workshops.
27. Axel Martens. Analyzing web service based business processes. In *Fundamental Approaches to Software Engineering*. Springer Berlin / Heidelberg, 2005.
28. Zoran Milosevic, Shazia Sadiq, and Maria Orlowska. Towards a methodology for deriving contract-compliant business processes. In *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, pages 395–400. Springer Berlin / Heidelberg, 2006.
29. Saayan Mitra, Ratnesh Kumar, and Samik Basu. Optimum decentralized choreography for web services composition. In *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, 2008.
30. Marco Montali. *Specification and Verification of Declarative Open Interaction Models: A Logic-Based Approach*, volume 56 of *Lecture Notes in Business Information Processing*. Springer, 2010.
31. Raghava Rao Mukkamala, Thomas Hildebrandt, and Janus Boris Tøth. The resultmaker online consultant: From declarative workflow management in practice to LTL. In *Proceeding of DDBP*, 2008.
32. M. Mukund. From global specifications to distributed implementations. In *Synthesis and Control of Discrete Event Systems*. Springer, 2002.
33. Nataliya Mulyar, Maja Pesic, Wil M. P. Van Der Aalst, and Mor Peleg. Declarative and procedural approaches for modelling clinical guidelines: addressing flexibility issues. In *Proceedings of the 2007 international conference on Business process management*, BPM'07, pages 335–346, Berlin, Heidelberg, 2008. Springer-Verlag.
34. Mangala Gowri Nanda, Satish Chandra, and Vivek Sarkar. Decentralizing execution of composite web services. *SIGPLAN Not.*, 39:170–187, October 2004.
35. Object Management Group BPMN Technical Committee. Business Process Model and Notation, version 2.0. Webpage, january 2011. `http://www.omg.org/spec/BPMN/2.0/PDF`.

36. Stefanie Rinderle, Andreas Wombacher, and Manfred Reichert. Evolution of process choreographies in dychor. In *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE*, volume 4275 of *LNCS*, pages 273–290. Springer, 2006.
37. David Robertson. A lightweight coordination calculus for agent systems. In *In Declarative Agent Languages and Technologies*, pages 183–197, 2004.
38. W. Sadiq, S. Sadiq, and K. Schulz. Model driven distribution of collaborative business processes. In *Services Computing, 2006. SCC '06. IEEE International Conference on*, pages 281 –284, sept. 2006.
39. Arthur ter Hofstede, Rob van Glabbeek, and David Stork. Query nets: Interacting workflow modules that ensure global termination. In *Business Process Management*. Springer Berlin / Heidelberg, 2003.
40. P Terenziani, S Montani, A Bottrighi, M Torchio, G Molino, and G. Correndo. The glare approach to clinical guideline: Main features. *Symposium on Computerized Guidelines and Protocols*, 101:62–6, April 2004.
41. W. M. P. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.
42. Wil M. P. van der Aalst, Niels Lohmann, Peter Massuthe, Christian Stahl, and Karsten Wolf. Multiparty Contracts: Agreeing and Implementing Interorganizational Processes. *The Computer Journal*, 53(1):90–106, January 2010.
43. Wil M. P. van der Aalst, Maja Pesic, and Helen Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.
44. Wil M.P van der Aalst and Maja Pesic. A declarative approach for flexible business processes management. In *Proceedings DPM 2006*, LNCS. Springer Verlag, 2006.
45. Wil M.P van der Aalst and Maja Pesic. DecSerFlow: Towards a truly declarative service flow language. In M. Bravetti, M. Nunez, and Gianluigi Zavattaro, editors, *Proceedings of Web Services and Formal Methods (WS-FM 2006)*, volume 4184 of *LNCS*, pages 1–23. Springer Verlag, 2006.
46. W.M.P. van der Aalst. Inheritance of interorganizational workflows: How to agree to disagree without loosing control? *Information Technology and Management*, 4:345–389, 2003.
47. Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz (Eds.) Rozenberg, editors, *Advances in Petri Nets*, volume Vol. 255 of *Lecture Notes in Computer Science,*, pages 325–392. Springer, 1986.
48. Dirk Wodtke and Gerhard Weikum. A formal foundation for distributed workflow execution based on state charts. In *Proceedings of the 6th International Conference on Database Theory*, pages 230–246, London, UK, 1997. Springer-Verlag.
49. X. Yi and K.J Kochut. Process composition of web services with complex conversation protocols. In *Design, Analysis, and Simulation of Distributed Systems Symposium at Adavanced Simulation Technology*, 2004.
50. W. Zielonka. Notes on finite asynchronous automata. *Informatique Thorique et Applications*, 21(2):99–135, 1987.