# Safe Distribution of Declarative Processes

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 give a general technique for safe distribution of a declarative (global) process as a network of (local) synchronously communicating declarative processes. Both the global and local processes are given as Dynamic Condition Response (DCR) Graphs. DCR Graphs is a recently introduced declarative process model generalizing labelled prime event structures to a systems model able to finitely represent $\omega$-regular languages. An operational semantics given as a transition semantics between markings of the graph allows DCR Graphs to be conveniently used as both specification and execution model. The technique for distribution is based on a new general notion of projection of DCR Graphs relative to a subset of labels and events identifying the set of external events that must be communicated from the other processes in the network in order for the distribution to be safe. We prove that for any vector of projections that covers a DCR Graph that the network of synchronously communicating DCR Graphs given by the projections is bisimilar to the original global process graph. We exemplify the distribution technique on a process identified in a case study of an cross-organizational case management system carried out jointly with Exformatics A/S.

**Keywords:** formal specification, distributed synthesis, cross-organizational workflow, declarative processes, process composition

## 1 Introduction

A model-driven software engineering approach to distributed information systems typically include both *global* models describing the collective behavior of the system being developed and *local* models describing the behavior of the individual peers or components.

The global and local descriptions should be consistent. If the modeling languages have formal semantics and the local model language support composition of individual processes, the consistency can be formally established, which we will refer to as the *consistency problem*: Given a global model and a set of local models, is the behavior of

* Authors listed alphabetically. This research is supported by the Danish Research Agency through a Knowledge Voucher granted to Exformatics (grant #10-087067, www.exformatics.com), 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 composition of the local models consistent with the global model? In order to support "top-down" model-driven engineering starting from the global model, one should address the more challenging *distributed synthesis problem*: Given a global model and some formal description of how the model should be distributed, can we synthesize a set of local processes with respect to this distribution which are consistent to the the global model?

In past work, briefly surveyed below, the result of the distributed synthesis have been a network of local processes described in an imperative process model, e.g. as a network of typed pi-calculus processes or a product automaton. The global process description has either been given declaratively, e.g. in some temporal logic, or imperatively, e.g. as a choreography or more generally a transition system.

In the present paper we address the distributed synthesis problem in a setting where both the global and the local processes are described *declaratively* as Dynamic Condition Response Graphs (DCR Graphs). DCR Graphs is a declarative workflow model introduced previously in [14, 15] as a generalization of the classical event structure model [47] allowing finite specification of infinite or iterative behavior (by allowing events to be executed more than once and replacing the symmetric conflict relation by asymmetric exclusion and (re-)inclusion relations) and specification of progress conditions (by replacing the causal order relation of event structures with two relations, respectively defining the conditions for and required responses to the execution of an event).

The motivation for introducing the DCR Graph model is to give, as part of the Trustworthy Pervasive Healthcare Services [13] project, a declarative model that can be used both as specification language and execution language for flexible workflow and business process. Indeed, the DCR Graphs model is inspired by and formalizes the core primitives of the process model employed by the industrial partner (Resultmaker) in the TrustCare project and is now being implemented in the workflow engine developed at Exformatics. As identified in e.g. [6,43] declarative process languages make it easier to specify loosely constrained systems. Also, we believe the declarative approach is more promising when it comes to composition, and (dynamic) changes of processes which is one of the main objectives of the TrustCare project.

To safely distribute a DCR Graph we first define (Def. 3, Sec. 3.1) a new general notion of *projection* of DCR Graphs relative to a subset of labels and events. The key point is to identify the set of events that must be communicated from other processes in the network in order for the state of the local process to stay consistent with the global specification (Prop. 1, Sec. 3). To also enable the reverse operation, building global graphs from local graphs, we then define the composition of two DCR Graphs, essentially by gluing joint events. As a sanity check we prove (Prop. 2, Sec. 3.2) that if we have a collection of projections of a DCR Graph that cover the original graph (Def. 7, Sec. 3.2) then the composition yields back the same graph. We then finally proceed to the main technical result, defining networks of synchronously communicating DCR Graphs and stating (in Thm. 1, Sec. 3.3) the correspondence between a global process and a network of communicating DCR Graphs obtained from a covering projection (relying on Prop. 1). Throughout the paper we exemplify the distribution technique on a simple cross-organizational process identified within a case study carried out jointly

with Exformatics A/S using DCR Graphs for model-driven design and engineering of an inter-organizational case management system. We conclude in Sec. 4 and provide pointers to future work.

### 1.1 Related Work

There are many researchers [1, 20, 21, 40–42, 46] who have explicitly focussed on the problem of verifying the correctness of inter-organizational workflows in the domain of petri nets. In [41], message sequence charts are used to model the interaction between the participant workflows that are modeled using petri nets and the overall workflow is checked for consistency against an interaction structure specified in message sequence charts. In [20] Kindler et. al. followed a similar but more formal and concrete approach, where the interaction of different workflows is specified using a set of scenarios given as sequence diagrams and using criteria of local soundness and composition theorem, guaranteed the global soundness of an inter-organizational workflow. The authors in [40] proposed *Query Nets* based on predicate/transition petri nets to guarantee global termination, without the need for having the global specification. The work on workflow nets [1, 46] use a P2P (Public-To-Private) approach to partition a shared public view of an inter-organizational workflow over its participating entities and projection inheritance is used to generate a private view that is a subclass to the relevant public view, to guarantee the deadlock and livelock freedom. Further a more liberal and a weaker notion than projection inheritance, *accordance* has been used in [42] to guarantee the weak termination in the multiparty contracts based on open nets.

Modeling global behavior as a set of conversations among participating services has been studied by many researchers [2, 3, 11, 35, 48, 49] in the area business processes. An approach based on guarded automata studied in [11], for the realizability analysis of conversation protocols, whereas the authors in [49] used colored petri nets to capture the complex conversations. A framework for calculating and controlled propagation of changes to the process choreographies based on the modifications to partner's private processes has been studied in [35]. Similarly, but using process calculus to model service contracts, Bravetti-Zavattaro proposed conformance notion for service composition in [2] and further enhanced their correctness criteria in [3] by the notion of strong service compliance.

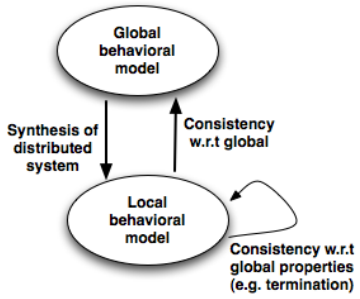



**Fig. 1.** Key problems studied in related work

Researchers [9,19,23,29] in the web services community have been working on web service composition and decentralized process execution using BPEL [30] and other re-

lated technologies to model the web services. A technique to partition a composite web service using program analysis was studied in [29] and on the similar approach, [19] 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 [23] 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 been researched for the imperative choreography language WS-CDL in the work on structured communication-centred programming for web services by Carbone, Honda and Yoshida [4]. To put it briefly, 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 [22], while [36] proposes an approach to extract a distributed process model from collaborative business process. In [9, 10], 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 [5, 12, 27] foundational work has been made on synthesizing distributed transition systems from global specification for the models of synchronous product and asynchronous automata [50]. In [27] Mukund categorized structural and behavioral characterizations of the synthesis problem for synchronous and loosely cooperating communication systems based on three different notions of equivalence: state space, language and bisimulation equivalence. Further Castellani et. al. [5] characterized when an an arbitrary transition system is isomorphic to its product transition systems with a specified distribution of actions and they have shown that for finite state specifications, a finite state distributed implementation can be synthesized. Complexity results for distributed synthesis problems for the three notions of equivalences were studied in [12].

Many commercial and research workflow management systems also support distributed workflow execution and some of them even support ad-hoc changes as well. ADEPT [34], Exotica [24], ORBWork [7], Rainman [31] and Newcastle-Nortel [39] are some of the distributed workflow management systems. A good overview and discussion about distributed workflow management systems can be found in [32, 33].

So far the formalisms discussed above are more or less 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 [8, 25] that have studied the synthesis problem in declarative modeling languages and none where both the global and local processes are given declaratively. In [8], 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 [25] 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, whereas the problem of synthesizing local models from a global model has not been studied.

## 2 Dynamic Condition Response Graphs

Dynamic Condition Response (DCR) Graphs has recently been introduced [15] as a declarative process model generalizing labelled event structures [47] to allow finite representations of infinite behavior (i.e. a systems model [37, 38]) and representation of progress properties.

A DCR Graph consists of a set of *labelled events*, a *marking* defining the *executed* events, *pending response* events and *included* events, and four binary relations between the events, defining the temporal constraints between events and dynamic inclusion and exclusion of events.

We employ the following notations in the rest of 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 a DCR Graph as follows.

**Definition 1.** *A Dynamic Condition Response Graph G is a tuple* $(\mathsf{E_G}, \mathsf{M_G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L_G}, l_\mathsf{G})$*, where*

*(i)* $\mathsf{E_G}$ *is the set of* events
*(ii)* $\mathsf{M_G} = (\mathsf{Ex_G}, \mathsf{Re_G}, \mathsf{In_G}) \in \mathcal{P}(\mathsf{E_G}) \times \mathcal{P}(\mathsf{E_G}) \times \mathcal{P}(\mathsf{E_G})$ *is the* marking,
*(iii)* $\rightarrow\bullet \subseteq \mathsf{E_G} \times \mathsf{E_G}$ *is the* condition *relation*
*(iv)* $\bullet\rightarrow \subseteq \mathsf{E_G} \times \mathsf{E_G}$ *is the* response *relation*
*(v)* $\pm : \mathsf{E_G} \times \mathsf{E_G} \rightharpoonup \{+, \%\}$ *is a partial function defining the* dynamic inclusion and exclusion *relations by* $e \rightarrow + \; e'$ *if* $\pm(e, e') = +$ *and* $e \rightarrow \% \; e'$ *if* $\pm(e, e') = \%$
*(vi)* $\mathsf{L_G}$ *is the set of* labels
*(vii)* $l_\mathsf{G} : \mathsf{E_G} \rightarrow \mathcal{P}(\mathsf{L_G})$ *is a labeling function mapping events to sets of labels.*

*We write* $\mathcal{M}(G)$ *for the set* $\mathcal{P}(\mathsf{E_G}) \times \mathcal{P}(\mathsf{E_G}) \times \mathcal{P}(\mathsf{E_G})$ *of markings of* $G$*.*

The marking $\mathsf{M_G} = (\mathsf{Ex_G}, \mathsf{Re_G}, \mathsf{In_G})$ is a tuple of three sets defining respectively the previously executed events ($\mathsf{Ex_G}$), the set of required responses ($\mathsf{Re_G}$), and the currently included events ($\mathsf{In_G}$). The set of required responses are the events that must eventually be executed (or excluded) in order to accept the execution, also referred to as the pending responses. The set of included events are the events that currently are relevant for conditions and may be executed (if their conditions are met). The condition relation $\rightarrow\bullet$ defines which (of the currently included) events must have been executed before an event can be executed. That is, for an event $e$ to be executed, it must be included, i.e. $e \in \mathsf{In_G}$ and the included conditions must be executed: $(\rightarrow\bullet \; e) \cap \mathsf{In_G} \subseteq \mathsf{Ex_G}$. The response relation $\bullet\rightarrow$ defines which responses are required after executing an event. That is, if the event $e$ is executed, the events $e \bullet\rightarrow$ are added to the set of required responses

in the marking. The dynamic inclusion and exclusion relations define how the set of included events changes by executing an event: If the event $e$ is executed, the events $e \rightarrow +$ are added to the set of included events in the marking and the events $e \rightarrow \%$ are removed. Finally, an event is labelled by zero or more labels. (This is slightly more general than previous work, where labels of events were sets of triples consisting of an action, a role and a principal.)

Fig. 2 below shows an example DCR Graph identified during the development by Exformatics of a cross-organizational case management system for the umbrella organization of unions in Denmark, named LO.

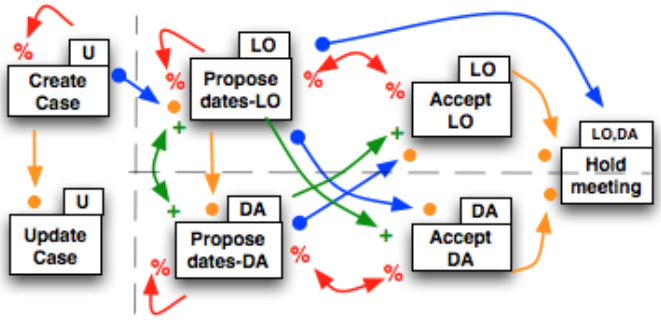



**Fig. 2.** Cross-organizational case management example

The graph has 7 events, drawn as boxes with "ears", and captures a process of creating a case, agreeing on meeting dates and holding meetings. The names of the events are written inside the box and the set of actions for each event, representing the roles that can execute the event, is written inside the "ear". That is, the event Create Case in the upper left has label U and represents the creation of a case by a case manager at a union (role U). The rightmost event, Hold meeting has two different labels, LO and DA, representing a meeting held by LO and DA (the umbrella organization of employers) respectively.

The semantics for DCR Graphs has been given in [14, 15] as a labelled transition system with acceptance condition for infinite computations. The set of accepted runs of DCR Graphs was characterized by a mapping to Büchi-automata in [26].

**Definition 2.** *For a DCR Graph* $G = (\mathsf{E_G}, \mathsf{M_G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L_G}, l_\mathsf{G})$*, we define the corresponding labelled transition system* $TS(G)$ *to be the tuple* $(\mathcal{M}(G), \mathsf{M_G}, \mathcal{EL}(G), \rightarrow)$ *where* $\mathcal{EL}(G) = \mathsf{E_G} \times \mathsf{L_G}$ *is the set of labels of the transition system,* $\mathsf{M_G} = (\mathsf{Ex_G}, \mathsf{In_G}, \mathsf{Re_G}) \in \mathcal{M}(G)$ *is the initial marking, and* $\rightarrow \subseteq \mathcal{M}(G) \times \mathcal{EL}(G) \times \mathcal{M}(G)$ *is the transition relation defined by* $\mathsf{M_G}' \xrightarrow{(e,a)} \mathsf{M_G}''$ *if*

*(i)* $\mathsf{M_G}' = (\mathsf{Ex_G}', \mathsf{In_G}', \mathsf{Re_G}')$ *is the marking before transition*
*(ii)* $\mathsf{M_G}'' = (\mathsf{Ex_G}'', \mathsf{In_G}'', \mathsf{Re_G}'')$ *is the marking after transition*

*(iii)* $e \in \mathsf{In_G}', a \in l_\mathsf{G}(e)$
*(iv)* $\rightarrow\bullet e \cap \mathsf{In_G}' \subseteq \mathsf{Ex_G}'$,
*(v)* $\mathsf{Ex_G}'' = \mathsf{Ex_G}' \cup \{e\}$
*(vi)* $\mathsf{In_G}'' = (\mathsf{In_G}' \cup e\rightarrow+) \setminus e\rightarrow\%$,
*(vii)* $\mathsf{Re_G}'' = (\mathsf{Re_G}' \setminus \{e\}) \cup e\bullet\rightarrow$,

*We define a run* $a_0, a_1, \ldots$ *of the transition system to be a sequence of labels of a sequence of transitions* $\mathsf{M_G}_i \xrightarrow{(e_i,a_i)} \mathsf{M_G}_{i+1}$ *starting from the initial marking. We define a run to be accepting if for the underlying sequence of transitions it holds that* $\forall i \geq 0, e \in \mathsf{Re_G}_i . \exists j \geq i . (e = e_j \vee e \notin \mathsf{In_G}_{j+1})$. *In words, a run is accepting if every response event either happen at some later state or become excluded.*

Condition *(iii)* in the above definition expresses that, only events that are currently included and mapped to the labels in $\mathsf{L_G}$ can be executed, Condition *(iv)* requires that all condition events to $e$ which are currently included should have been executed previously. Condition *(v)*, *(vi)* and *(vii)* are the updates to the sets of executed, included events and required responses respectively. Note that an event $e'$ can not be both included and excluded by the same event $e$, but an event may trigger itself as a response.

To ease keeping track of transition systems of different DCR Graphs we extend the transition system to transitions between graphs in the obvious way, writing $G \xrightarrow{(e,a)} G'$ if $G = (\mathsf{E_G}, \mathsf{M_G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L_G}, l_\mathsf{G}), \mathsf{M_G} \xrightarrow{(e,a)} \mathsf{M_G}'$ in $TS(G)$ and $G' = (\mathsf{E_G}, \mathsf{M_G}', \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L_G}, l_\mathsf{G})$.

## 3 Projection and Composition

In this section we define projections and compositions of dynamic condition response graphs.

### 3.1 Projection

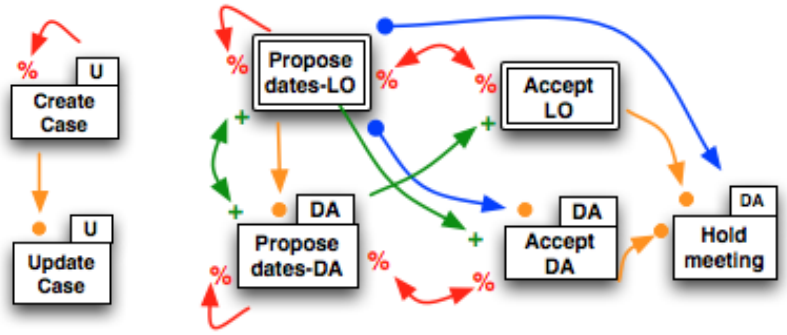
First we define how to project a DCR Graph $G$ with respect to a *projection parameter* $\delta = (\mathsf{E}_\delta, \mathsf{L}_\delta)$, where $\mathsf{E}_\delta \subseteq \mathsf{E_G}$ is a subset of the events of $G$ and $\mathsf{L}_\delta \subseteq \mathsf{L_G}$ is a subset of the labels.

Intuitivly, the projection $G_{|\delta}$ contains only those events and relations that are relevant for the execution of events in $\mathsf{E}_\delta$ and the labeling is restricted to the set $\mathsf{L}_\delta$. This includes both the events in $\mathsf{E}_\delta$ and any other event that can affect the marking, or ability to execute of an event in $\mathsf{E}_\delta$ through one or more relations.

**Definition 3.** *If* $G = (\mathsf{E_G}, \mathsf{M_G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L_G}, l)$ *then* $G_{|\delta} = (\mathsf{E_G}_{|\delta}, \mathsf{M_G}_{|\delta}, \rightarrow\bullet_{|\delta}, \bullet\rightarrow_{|\delta}, \pm_{|\delta}, \mathsf{L}_\delta, l_{|\delta})$ *is the projection of* $G$ *with respect to* $\delta \subseteq \mathsf{E_G}$ *where:*

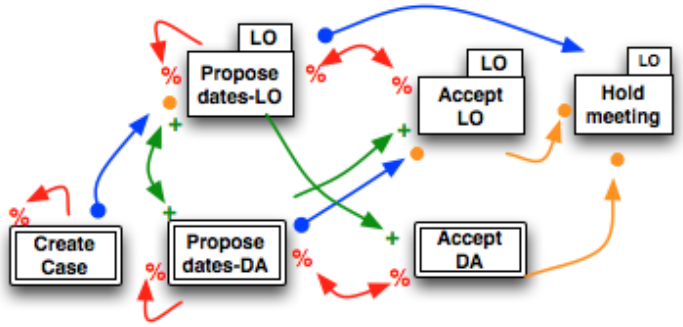
*(i)* $\mathsf{E_G}_{|\delta} = \rightarrow\mathsf{E}_\delta$, *for* $\rightarrow = \bigcup_{c \in C} c$, *and* $C = \{\mathsf{id}, \rightarrow\bullet, \bullet\rightarrow, \rightarrow+, \rightarrow\%, \rightarrow+\rightarrow\bullet, \rightarrow\%\rightarrow\bullet\}$

*(ii)* $l_{|\delta}(e) = \begin{cases} l_\mathsf{G}(e) \cap \mathsf{L}_\delta & \text{if } e \in \mathsf{E}_\delta \\ \emptyset & \text{if } e \in \mathsf{E_G}_{|\delta} \setminus \mathsf{E}_\delta \end{cases}$




(a) Projection Over Role U

(b) Projection Over Role DA




(c) Projection Over Role LO

**Fig. 3.** Projecting of Arrange Meeting Example Over Roles

*(iii)* $\mathsf{M}_{\mathsf{G}|_\delta} = (\mathsf{Ex}_{\mathsf{G}|_\delta}, \mathsf{Re}_{\mathsf{G}|_\delta}, \mathsf{In}_{\mathsf{G}|_\delta})$ *where:*
  *(a)* $\mathsf{Ex}_{\mathsf{G}|_\delta} = \mathsf{Ex}_{\mathsf{G}} \cap \mathsf{E}_{\mathsf{G}|_\delta}$
  *(b)* $\mathsf{Re}_{\mathsf{G}|_\delta} = \mathsf{Re}_{\mathsf{G}} \cap \mathsf{E}_\delta$
  *(c)* $\mathsf{In}_{\mathsf{G}|_\delta} = (\mathsf{In}_{\mathsf{G}} \cap ((\mathsf{id} \cup \rightarrow\bullet)\mathsf{E}_\delta)) \cup (\mathsf{E}_{\mathsf{G}|_\delta} \setminus ((\mathsf{id} \cup \rightarrow\bullet)\mathsf{E}_\delta))$

*(iv)* $\rightarrow\bullet_{|_\delta} = \rightarrow\bullet \cap ((\rightarrow\bullet\ \mathsf{E}_\delta) \times \mathsf{E}_\delta)$

*(v)* $\bullet\rightarrow_{|_\delta} = \bullet\rightarrow \cap ((\bullet\rightarrow\ \mathsf{E}_\delta) \times \mathsf{E}_\delta)$

*(vi)* $\rightarrow+_{|_\delta} = \rightarrow+ \cap (((\rightarrow+ \rightarrow\bullet\ \mathsf{E}_\delta) \times (\rightarrow\bullet\ \mathsf{E}_\delta)) \cup ((\rightarrow+\ \mathsf{E}_\delta) \times \mathsf{E}_\delta))$

*(vii)* $\rightarrow\%_{|_\delta} = \rightarrow\% \cap (((\rightarrow\% \rightarrow\bullet\ \mathsf{E}_\delta) \times (\rightarrow\bullet\ \mathsf{E}_\delta)) \cup ((\rightarrow\%\ \mathsf{E}_\delta) \times \mathsf{E}_\delta))$

*(i)* defines the set of events as the union of the set $\mathsf{E}_\delta$ of events that we project over, any event that has a direct relation towards an event in $\mathsf{E}_\delta$ and events that exclude or include an event which is a condition for an event in $\mathsf{E}_\delta$. The additional events will be included in the projection without labels, as can be seen from the definition of the labeling function in *(ii)*. This means that the events can not be executed locally. However, when composed in a network containing other processes that can execute these events, their execution will be communicated to the process. For this reason we refer to these

events as the (additional) external events of the projection. As proven in Prop. 1 the communication of the execution of this set of external events in addition to the local events shared by others ensure that the local state of the projection stay consistent with the global state. *(iii)* defines the projection of the marking: The executed events remain the same, but are limited to the events in $\mathsf{E}_{\mathsf{G}|\delta}$. The responses are limited to events in $\mathsf{E}_\delta$ because these are the only responses that will affect the local execution of the projected graph. The set of included events remains the same for events in $\mathsf{E}_\delta$ or $\mathsf{E}_\delta{}^{\rightarrow\bullet}$, because these can affect which events are enabled in the projected graph. All other external events of the projected graph are included regardless of their state in the marking of the global graph. This is because in the local process is only notified of the execution of these events, not their in- or exclusion. Finally, *(iv)*, *(v)*, *(vi)* and *(vii)* state which relations should be included in the projection. For the events in $\mathsf{E}_\delta$ all incoming relations should be included. Additionally inclusion and exclusion relations to events that are a condition for an event in $\mathsf{E}_\delta$ are included as well.

To define networks of communicating DCR Graphs and their semantics we use the following extension of a DCR Graph allowing any event to be executed with a special input label. These transitions will only be used for the communication in a network and thus not be visible as user events.

**Definition 4.** *For a DCR Graph* $G = (\mathsf{E}_\mathsf{G}, \mathsf{M}_\mathsf{G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L}_\mathsf{G}, l)$ *define* $G^\flat = (\mathsf{E}_\mathsf{G}, \mathsf{M}_\mathsf{G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L}_\mathsf{G} \cup \{\flat\}, l^\flat)$*, where* $l^\flat = l(e) \cup \{\flat\}$ *(assuming that* $\flat \notin \mathsf{L}_\mathsf{G}$*).*

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* $G = (\mathsf{E}_\mathsf{G}, \mathsf{M}_\mathsf{G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L}_\mathsf{G}, l)$ *be a DCR Graph and* $G_{|\delta}$ *its projection with respect to a projection parameter* $\delta = (\mathsf{E}_\delta, \mathsf{L}_\delta)$*. Then*

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

### 3.2 Composition

Now we define the binary composition of two DCR Graphs. Intuitively, the *composition* of $G_1$ and $G_2$ glues together the events that are both in $G_1$ and $G_2$.

**Definition 5.** *Formally, the composite* $G_1 \oplus G_2 = (\mathsf{E}_\mathsf{G}, \mathsf{M}_\mathsf{G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L}_\mathsf{G}, l)$*, where* $G_i = (\mathsf{E}_{\mathsf{G}i}, \mathsf{M}_{\mathsf{G}i}, \rightarrow\bullet_i, \bullet\rightarrow_i, \pm_i, \mathsf{L}_{\mathsf{G}i}, l_i)$*,* $\mathsf{M}_{\mathsf{G}i} = (\mathsf{Ex}_{\mathsf{G}i}, \mathsf{Re}_{\mathsf{G}i}, \mathsf{In}_{\mathsf{G}i})$ *and:*

*(i)* $\mathsf{E}_\mathsf{G} = (\mathsf{E}_{\mathsf{G}1} \cup \mathsf{E}_{\mathsf{G}2})$
*(ii)* $\mathsf{M}_\mathsf{G} = (\mathsf{Ex}_\mathsf{G}, \mathsf{Re}_\mathsf{G}, \mathsf{In}_\mathsf{G})$*, where:*
  *(a)* $\mathsf{Ex}_\mathsf{G} = \mathsf{Ex}_{\mathsf{G}1} \cup \mathsf{Ex}_{\mathsf{G}2}$
  *(b)* $\mathsf{In}_\mathsf{G} = (\mathsf{In}_{\mathsf{G}1} \cup \mathsf{In}_{\mathsf{G}2}) \setminus (((\mathsf{E}_{\mathsf{G}1}{}^i_1 \cup \rightarrow\bullet\mathsf{E}_{\mathsf{G}1}{}^i_1) \setminus \mathsf{In}_{\mathsf{G}1}) \cup ((\mathsf{E}_{\mathsf{G}2}{}^i_2 \cup \rightarrow\bullet\mathsf{E}_{\mathsf{G}2}{}^i_2) \setminus \mathsf{In}_{\mathsf{G}2}))$
  *(c)* $\mathsf{Re}_\mathsf{G} = \mathsf{Re}_{\mathsf{G}1} \cup \mathsf{Re}_{\mathsf{G}2}$
  *for* $\mathsf{E}_{\mathsf{G}}{}^i_j = \{e \in \mathsf{E}_{\mathsf{G}j} \mid l_j(e) \neq \emptyset\}$

*(iii)* $\rightarrow = \rightarrow_1 \cup \rightarrow_2$ *for each* $\rightarrow \in \{\rightarrow\bullet, \bullet\rightarrow, \rightarrow+, \rightarrow\%\}$
*(iv)* $l(e) = l_1(e) \cup l_2(e)$
*(v)* $\mathsf{L_G} = \mathsf{L_{G_1}} \cup \mathsf{L_{G_2}}$

*(iib)* states that events are included, if they're either included in $G_1$ or $G_2$, unless they are events that are either internal or have a condition towards an internal event and are excluded in $G_1$ or $G_2$. The intuition here is that if an event is internal or has a condition towards an internal event, then it affects the enabled events of the graph, so it's inclusion status should be the same in the composed graph. The inclusion/exclusion status of other external events however may simply not have been updated because the graph is not aware of all relations towards these events. This is not unsafe because the inclusion of these events does not affect the execution of the graph. Therefor the definition states that if an event is internal or has a condition towards an internal event in $G_1$ or $G_2$, then it's inclusion status should be the same in the composed graph, and in any other case the event is included if it was included in $G_1$ or $G_2$. *(iic)* states that the events with pending responses are those events that have a pending response in $G_1$ or $G_2$.

**Definition 6.** *The composition* $G_1 \oplus G_2$ *is* well-defined *when:*

*(i)* $\forall(e \in \mathsf{E_{G1}} \cap \mathsf{E_{G2}} \mid (e \in \mathsf{Ex_{G_1}} \Leftrightarrow e \in \mathsf{Ex_{G_2}})$
*(ii)* $\forall(e \in (\mathsf{E}^i_{\mathsf{G}_1} \cup \rightarrow\bullet\mathsf{E}^i_{\mathsf{G}_1}) \cap (\mathsf{E}^i_{\mathsf{G}_2} \cup \rightarrow\bullet\mathsf{E}^i_{\mathsf{G}_2}) \mid (e \in \mathsf{In_{G1}} \Leftrightarrow e \in \mathsf{In_{G_2}})$
*(iii)* $\forall(e \in \mathsf{E}^i_{\mathsf{G}_1} \cap \mathsf{E}^i_{\mathsf{G}_2} \mid (e \in \mathsf{Re_{G1}} \Leftrightarrow e \in \mathsf{Re_{G_2}})$
*(iv)* $\forall(e, e' \in \mathsf{E_{G1}} \cap \mathsf{E_{G2}} \mid \neg((e \rightarrow_1 e' \wedge e \rightarrow\%_2 e') \vee (e \rightarrow\%_1 e' \wedge e \rightarrow+_2 e')))$

*(i)* ensures that those events that will be glued together have the same execution marking. *(ii)* ensures that events that will be glued together and in both DCR Graphs belong to either the set of internal events or the set of events that have a conditional relation towards an internal event, have the same inclusion marking. *(iii)* ensures that events that will be glued together and in both DCR Graphs belong to the set of internal events have the same pending response marking. *(iv)* ensures that by composing the two DCR Graphs no event both includes and excludes the same event. If $G_1 \oplus G_2$ is well-defined, then we also say that $G_1$ and $G_2$ are *composable* with respect to eachother.

**Lemma 1.** *The composition operator* $\oplus$ *is commutative and associative.*

**Definition 7.** *We call a vector* $\Delta = \delta_1 \ldots \delta_k$ *of projection parameters* covering *for some DCR Graph* $G = (\mathsf{E_G}, \mathsf{M_G}, \rightarrow\bullet, \bullet\rightarrow, \pm, \mathsf{L_G}, l_\mathsf{G})$ *if:*

1. $\bigcup_{i \in [k]} \mathsf{E}_{\delta_i} = \mathsf{E_G}$ *and*
2. $(\forall a \in \mathsf{L_G}. \forall e \in \mathsf{E_G}. a \in l_\mathsf{G}(e) \Rightarrow (\exists i \in [k]. e \in \mathsf{E}_{\delta_i} \wedge a \in \mathsf{L}_{\delta_i})$

**Proposition 2.** *If some vector* $\Delta = \delta_1 \ldots \delta_k$ *of projection parameters is covering for some DCR Graph* $G$ *then:* $\bigoplus_{i \in [k]} G_{|\delta_i} = G$

### 3.3 Safe Distributed Synchronous Execution

In this section we define networks of synchronously communicating DCR Graphs and prove the main technical theorem of the paper stating that a network of synchronously communicating DCR Graphs obtained by projecting a DCR Graph G with respect to a covering set of projection parameters has the same behavior as the original graph G.

**Definition 8.** *We define a* network of synchronously communicating 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 $\mathsf{E}$ and labels in $\mathsf{L}$.*

*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 $\mathcal{E}(G) = \mathsf{E}_\mathsf{G}$ 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 $\mathcal{L}(G) = \mathsf{L}_\mathsf{G}$ and $\mathcal{L}(N_1\|N_2) = \mathcal{L}(N_1) \cup \mathcal{L}(N_2)$.*

**Definition 9.** *The semantics of networks of buffered DCR Graphs are given by the following inference rules:*

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

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

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

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

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

*For a network of synchronously communicating 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 $\forall j \in [n], \forall i \geq 0, e \in \mathsf{Re}_{\mathsf{G}_{j,i}}.\exists k \geq i.(e = e_k \vee e \notin \mathsf{In}_{\mathsf{G}_{j,k+1}})$,*

*where* $\mathsf{Re}_{\mathsf{G}_{j,i}}$ *is the set of required responses in the* $j$*th DCR Graph in the network in the* $i$*th step of the run. In words, a run is accepting if every response event in a local DCR Graph in the network either happen at some later state or become excluded.*

We are now ready to give the main theorem of the paper, stating the correspondence between a global DCR Graph and the network of synchronously communicating DCR Graph obtained from a covering projection.

**Theorem 1.** *For a Dynamic Condition Response Graph G and a covering vector of projection parameters* $\Delta = \delta_1 \dots \delta_n$ *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* $TS(G_\Delta)$*.*

### 3.4 Example

In this section, we will use the arrange meeting example from Sec. 1 and show how events are executed in distributed setting. We assume the arrange meeting example is projected to a network $G^1_u \mid\mid G^1_{da} \mid\mid G^1_{lo}$ of three DCR Graphs as shown in the Fig. 3 and described in Sec. 3 and abbreviate the names for the events.

1. Using *sync step*, *local input*, and *input* we get the transition $G^1_u \mid\mid G^1_{da} \mid\mid G^1_{lo} \xrightarrow{(\texttt{Cc},\texttt{U})} G^2_u \mid\mid G^1_{da} \mid\mid G^2_{lo}$ capturing the local execution of the event `Cc` labelled with `U` in $G^1_u$ which is communicated synchronously to $G^1_{lo}$. This updates the markings by adding the event `Cc` to the set of executed events in both $G^1_u$ and $G^1_{lo}$. But since `Cc` has an exclude relation to itself in both $G^1_u$ and $G^1_{lo}$ (see Fig. 3(a) and 3(c)), the event is also excluded from the set of included events in both markings. Finally, because of the response relation to the event `PdLO` in $G^1_{lo}$ (see Fig. 3(c)), the event `PdLO` is added to the set of required responses in the resulting marking $G^2_{lo}$.
2. We can now execute the event `PdLO` in the DCR graph $G^2_{lo}$ concurrently with the event `Uc` in DCR graph $G^2_u$.
   As the event `Uc` is only local to $G^2_u$ we get by using *local step* the transition $G^2_u \mid\mid G^1_{da} \mid\mid G^2_{lo} \xrightarrow{(\texttt{Uc},\texttt{U})} G^3_u \mid\mid G^1_{da} \mid\mid G^2_{lo}$ that only updates the marking of $G^2_u$.
   In addition to being local to $G^2_{lo}$, the event `PdLO` is also external event in graph $G^1_{da}$, so as in the first step by using *sync step local input*, and *input* we get the transition $G^3_u \mid\mid G^1_{da} \mid\mid G^2_{lo} \xrightarrow{(\texttt{PdLO},\texttt{LO})} G^3_u \mid\mid G^2_{da} \mid\mid G^3_{lo}$, where the event `PdLO` has been added to the executed event set of both the marking of $G^1_{da}$ and $G^2_{lo}$. Again, because of the self-exclusion relations, the event `PdLO` is also excluded from the sets of included events in the two markings, and because of the response relations, the events `ADA` and `Hm` are added to the set of pending responses in $G^1_{da}$ and the event `Hm` is added to the set of pending responses in $G^2_{lo}$.
3. In response to the dates proposed by `LO`, the `DA` may choose to propose new dates by executing the event `PdDA` in the graph graph $G^2_{da}$.
   $G^3_u \mid\mid G^2_{da} \mid\mid G^3_{lo} \xrightarrow{(\texttt{PdDA},\texttt{DA})} G^3_u \mid\mid G^3_{da} \mid\mid G^4_{lo}$ This triggers the exclusion of the events `PdDA` and `ADA` and the inclusion of the events `PdLO` and `ALO` in the markings of both $G^2_{da}$ and $G^3_{lo}$. It will also include the event `ALO` in the required response set in the resulting marking $G^4_{lo}$.

4. Now LO may choose to accept the new dates proposed by DA by executing the event ALO in the graph graph $G^4_{lo}$, giving the transition
   $G^3_u \mid\mid G^3_{da} \mid\mid G^4_{lo} \xrightarrow{(\texttt{ALO},\texttt{LO}))} G^3_u \mid\mid G^4_{da} \mid\mid G^5_{lo}$. This records the event ALO as executed in markings of both $G^4_{da}$ and $G^5_{lo}$ and excludes PdLO in both markings (i.e. it is not possible to propose new dates after acceptance).
5. Since the event ALO is recorded as executed in markings of both $G^4_{da}$ and $G^5_{lo}$ and the event ADA is excluded, the hold meeting event Hm will be enabled in both graphs $G^5_{lo}$ and $G^4_{da}$. The LO may choose to hold the meeting, giving the transition
   $G^3_u \mid \rhd G^4_{da} \mid\mid G^5_{lo} \xrightarrow{(\texttt{Hm},\texttt{LO})} G^3_u \mid\mid G^5_{da} \mid\mid G^6_{lo}$
   Note that this event is also communicated to DA, added to the set of executed events and removed from the set of pending responses. Since there are no pending responses in any of the local graphs the finite run is in an accepting state.

## 4 Conclusion

We have given a general technique for distributing a declarative (global) process as a network of synchronously communicating (local) declarative processes and proven the global and distributed execution to be equivalent.

The global and local processes are given as Dynamic Condition Response (DCR) Graphs, a recently introduced declarative process model generalizing labelled prime event structures to a systems model able to finitely represent $\omega$-regular languages. The DCR Graph model has the advantage that it is on the one hand declarative and compositional, and on the other hand it has a simple and intuitive operational semantics given as a transition semantics between markings of the graph. This allows the model to be used both as specification and execution model.

As briefly surveyed in Sec. 1.1 there have been a lot of related work on synthesis of distributed systems and proving consistency with respect to a global model or property. We believe this is the first treatment where both the local and global models are given declaratively in the same model. This maintains the flexibility of a declarative model for the local processes, and allows local processes to be further distributed if necessary.

We exemplified the safe distribution technique on a process identified in a case study of an inter-organizational case management system carried out jointly with Exformatics A/S.

We leave for future work to study the harder problem of asynchronously communicating distributed processes. This may benefit from researching the true concurrency semantics inherent in the model and extend the transition semantics to include concurrency, e.g. like in [18, 28]. We also plan to study behavioral types describing the interfaces between communicating DCR Graphs, extending the work on session types in [4] to a declarative setting. Moreover, we intend to address extension of the DCR Graph model with time, data and dynamic instantiation of sub processes (also referred to multiple instances) to be able to model more realistic workflow processes. A first step is taken in [17] extending DCR Graphs to allow nested sub graphs. This extension introduced an additional relation between events, the milestone relation, making it possible to express the acceptance of a sub graph succinctly. We believe the results

in the present paper can be extended to nested DCR Graphs and the milestone relation, although it will complicate the definition of projections.

Finally, we plan to continue the ongoing implementation of tools for DCR Graphs, and in particular to implement the safe distribution technique in the current prototype design and simulation tools briefly described in [16].

## 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. 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.
3. Mario Bravetti and Gianluigi Zavattaro. A theory of contracts for strong service compliance. *Mathematical. Structures in Comp. Sci.*, 19:601–638, June 2009.
4. 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.
5. 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.
6. David Cohn and Richard Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Eng. Bull.*, 32(3):3–9, 2009.
7. S. Das, K. Kochut, J. Miller, A. Sheth, and D. Worah. Orbwork: A reliable distributed corba-based workflow enactment system for meteor2. Technical report, The University of Georgia, 1996.
8. Dirk Fahland. Towards analyzing declarative workflows. In *Autonomous and Adaptive Web Services*, 2007.
9. Walid Fdhila and Claude Godart. Toward synchronization between decentralized orchestrations of composite web services. In *CollaborateCom'09*, pages 1–10, 2009.
10. Walid Fdhila, Ustun Yildiz, and Claude Godart. A flexible approach for automatic process decentralization using dependency tables. *International Conference on Web Services*, 2009.
11. 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.
12. 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.
13. Thomas Hildebrandt. Trustworthy pervasive healthcare processes (TrustCare) research project. Webpage, 2008. `http://www.trustcare.dk/`.
14. Thomas Hildebrandt and Raghava Rao Mukkamala. Declarative event-based workflow as distributed dynamic condition response graphs. In *Post-proceedings of PLACES 2010*, 2010.
15. Thomas Hildebrandt and Raghava Rao Mukkamala. Distributed dynamic condition response structures. In *Pre-proceedings of International Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software (PLACES 10)*, March 2010.
16. 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) `http://www.itu.dk/people/rao/pubs_accepted/dcrscasestudy-edoc11.pdf`.

17. 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.
18. Thomas Hildebrandt and Vladimiro Sassone. Comparing transition systems with independence and asynchronous transition systems. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 84–97. Springer Berlin / Heidelberg, 1996.
19. 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.
20. 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.
21. Axel Martens. Analyzing web service based business processes. In *Fundamental Approaches to Software Engineering*. Springer Berlin / Heidelberg, 2005.
22. 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.
23. 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.
24. C. Mohan, D. Agrawal, G. Alonso, A. El Abbadi, R. Guenthoer, and M. Kamath. Exotica: a project on advanced transaction management and workflow systems. *SIGOIS Bull.*, 16:45–50, August 1995.
25. 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.
26. Raghava Rao Mukkamala and Thomas Hildebrandt. From dynamic condition response structures to büchi automata. In *Proceedings of 4th IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE 2010)*, August 2010.
27. M. Mukund. From global specifications to distributed implementations. In *Synthesis and Control of Discrete Event Systems*. Springer, 2002.
28. Madhavan Mukund and Mogens Nielsen. Ccs, locations and asynchronous transition systems. In Rudrapatna Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 652 of *Lecture Notes in Computer Science*, pages 328–341. Springer Berlin / Heidelberg, 1992.
29. Mangala Gowri Nanda, Satish Chandra, and Vivek Sarkar. Decentralizing execution of composite web services. *SIGPLAN Not.*, 39:170–187, October 2004.
30. OASIS WSBPEL Technical Committee. Web Services Business Process Execution Language, version 2.0, 2007. `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf`.
31. Santanu Paul, Edwin Park, and Jarir Chaar. Rainman: a workflow system for the internet. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, 1997.
32. F. Ranno and S. K. Shrivastava. A review of distributed workflow management systems. In *Proceedings of the international joint conference on Work activities coordination and collaboration*, 1999.
33. M. U. Reichert, T. Bauer, and P. Dadam. Flexibility for distributed workflows. In *Handbook of Research on Complex Dynamic Process Management: Techniques for Adaptability in Turbulent Environments*, pages 137–171. IGI Global, Hershey, PA, 2009.

34. Manfred Reichert and Thomas Bauer. Supporting ad-hoc changes in distributed workflow management systems. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, pages 150–168. Springer Berlin / Heidelberg, 2007.
35. 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.
36. 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.
37. V. Sassone, M. Nielsen, and G. Winskel. A classification of models for concurrency. In *4th International Conference on Concurrency Theory, CONCUR '93.*, volume Lectur of *LNCS*, pages 82–96. Springer, 1993.
38. Vladimiro Sassone, Mogens Nielsen, and Glynn Winskel. Models for concurrency: Towards a classification. *Theoretical Computer Science*, 170:297–348, 1996.
39. Wheater Shrivastava, S. M. Wheater, S. K. Shrivastava, and F. Ranno. A corba compliant transactional workflow system for internet applications. In *Proc. Of IFIP Intl. Conference on Distributed Systems Platforms and Open Distributed Processing, Middleware 98*, pages 1–85233. Springer-Verlag, 1998.
40. 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.
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 Rozenberg, editors, *Advances in Petri Nets*, volume 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.