Deriving Consistent GSM Schemas from DCR Graphs

Rik Eshuis[1], Søren Debois[2,3](✉), Tijs Slaats[2,4], and Thomas Hildebrandt[2]

[1] School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
h.eshuis@tue.nl

[2] IT University of Copenhagen, Copenhagen, Denmark
{debois,tslaats,hilde}@itu.dk

[3] Exformatics A/S, Copenhagen, Denmark
debois@exformatics.com

[4] Department of Computer Science, University of Copenhagen, Copenhagen, Denmark

Abstract. Case Management (CM) is a BPM technology for supporting flexible services orchestration. CM approaches like CMMN, an OMG standard, and GSM, one of CMMN's core influences, use Event-Condition-Action rules, which can be inconsistent due to cyclic interdependencies between the rules; repairing such an inconsistent case management schema is difficult. To avoid the problem of inconsistencies altogether, we provide a technique for automatically deriving consistent GSM case management schemas from higher-level business policies defined as DCR graphs, an alternative CM approach. Concretely, we define a behaviour-preserving mapping that (1) removes the burden from the modeller of GSM schemas to prove consistency and define the ordering of rules, (2) provides high-level patterns for modelling GSM schemas, and (3) gives a way to define a notion of progress (liveness) and acceptance for GSM instances. The mapping is illustrated by a running example of a mortgage loan application; and a prototype implementation available at http://dcr.itu.dk/icsoc16.

1 Introduction

The standard notations for business process modelling and service orchestration such as BPMN [4] and BPEL [3] are tailored to highly stable and repeatable business processes, for which it makes sense to pre-specify an explicit control flow in terms of classical imperative primitives for sequencing, looping and branching. However, recently Case Management has emerged as a response to the need for more flexibility when supporting knowledge workflows [23]. A knowledge workflow is characterised by having a more unpredictable sequencing of tasks and service executions, e.g. depending on the concrete case at hand and its context, individual tasks or service executions may need to be skipped or repeated, thereby deviating from the "happy path".

The OMG recently defined the standard Case Management Model and Notation (CMMN) [2], which takes a declarative approach and places a stronger focus

Q.Z. Sheng et al. (Eds.): ICSOC 2016, LNCS 9936, pp. 467–482, 2016.
DOI: 10.1007/978-3-319-46295-0_29

on describing the rules instead of the flow of a process. While CMMN is still only in its early stages and has no formal semantics yet, it has been highly influenced by the Guard Stage Milestone (GSM) model [16], which has both a formal semantics [5,10] and is supported by an open source implementation [14]. Activities, e.g. human tasks or service calls, in GSM are tied to *stages*. Conditions ("guards") referring to events and data, govern which stages are open, hence which activities/services are available, and which goals ("milestones") have been met. The constraint language used to specify guards in GSM is a variant on Event-Condition-Action (ECA) rules.

A particular challenge in the creation of GSM and CMMN Schemas is that the rules are *cascading*, i.e., the firing of one rule may require the firing of another in the same reaction step. To avoid race conditions and infinite cascading, rules are required to be *ordered* and *consistent*. The right ordering and consistency of rules can however be difficult in practice to obtain for the modeller: It is dangerously easy to accidentally specify two distinct rules where each trigger the condition of the other. Recent research has explored both syntactic [5,10] and semantic [13] approaches to *discovering* inconsistent rules. However, the question remains how modellers can repair discovered inconsistencies and how consistent GSM schemas can be modelled from high-level requirements and business policies.

In the present paper we address and answer this latter question by providing a mapping from Dynamic Condition Response (DCR) graphs [15] to GSM Schemas and prove that the resulting schema is always consistent. The DCR graph model is a formal, declarative case management notation which is implemented in and supported by an industrial modelling and simulation tool [12,17] and a commercial case management software product [6,8,21]. Compared to GSM and CMMN, the DCR graph model is centred around the high-level notion of *events*. An event can e.g. be the start or end of an activity/service invocation. Five basic relations between events allow to constraint their temporal ordering and define obligations before the process can end, e.g. that one event is a condition for another event to happen, or that an event is required as a response following another event.

The main technical contribution of the present paper is to provide a formal, semantics-preserving translation from DCR to GSM. This translation is notable for the following reasons.

1. It proves that *any* DCR graph has a semantically equivalent GSM schema (Theorem 1, Corollary 1).
2. It demonstrates how the notion of *acceptance* of DCR graphs can be recovered in GSM schemas *even though GSM does not have a corresponding primitive notion* (Subsect. 4.4).
3. It provides a method for deriving *consistent* GSM schemas from a given set of high-level rules, via formalisation of these rules in a DCR model (Lemma 1).
4. It relates the DCR notation via GSM to the emerging OMG CMMN standard [2,16].

In particular (3) shows that the present work embodies a potential solution to the thorny issue of consistency of GSM schemas: Using DCR rule-patterns as a high-level rule-specification mechanism, we can, using the translation, *generate automatically* a consistent GSM schema guaranteed to have the same semantics (1).

Related Work. Both GSM and DCR models are so-called "declarative notations", an approach introduced to the BPM community by the DECLARE [1] notation. Compared to DECLARE, GSM is strongly data-centric, whereas DCR combines a marking-based operational semantics with a smaller set of declarative constraints, yet yielding a higher degree of formal expressiveness.

Tentative steps to relating GSM and DCR were taken already at the inception of DCR graphs [18], but no full mapping has been developed prior to the present work. DCR models were formally proven to express exactly the union of regular and ω-regular languages in [7], via an encoding to Büchi automata. Otherwise, we are unaware of work relating DCR to other languages via formal translations.

GSM models were formalised as Data-Centric Dynamic Systems (DCDS) in [22], for the purposes of supporting automatic verification. The present work takes the inverse approach: We use DCR Graphs as a declarative policy language from which we can automatically derive consistent GSM models. Closer to the present work, [11] proposes a semi-automated approach to synthesise GSM models from UML activity diagrams, which specify the flow of multiple stateful objects between activities. However, whereas the UML language emphasises describing the life-cycles of objects in an imperative way, DCR emphasises describing in a declarative way compliance rules and policies that activity execution should adhere to. Finally, [19] provides a translation from Petri nets to GSM models to enable the use of process mining algorithms (which output Petri nets) for generating GSM models, while [20] maps GSM models to a public/subscribe abstraction. These papers do not address repair of inconsistencies.

Overview. In Sect. 2 resp. 3, we recall the formal definitions of DCR graphs and GSM schemas; we intertwine the formal definitions with a running example of a mortgage application process from [8]. In Sect. 4, we give the formal translation from DCR to GSM and prove it is semantics preserving. In Sect. 5, we conclude. Due to space limitations, proofs have been relegated to the technical report [9]. A prototype implementation of the translation is available in the DCR Workbench, available at http://dcr.itu.dk/icsoc16.

2 DCR Graphs

In this Section, we recall the theory of DCR graphs. We exemplify DCR graphs by giving a model of a mortgage application process based a real-world process [8]. The purpose of the process is to arrive at a point where a loan application can be assessed. This requires in turn:

1. Collecting appropriate documentation,
2. collecting a budget from the applicant, and
3. appraising the property.

The main actor in the process is the *caseworker*, who collects the documents, may perform a statistical appraisal of the property, and finally assesses the application. The budget needs to be submitted by the *customer*, then screened by an *intern*. The case worker only proceeds to assess the application once the intern has screened the budget.

If a statistical appraisal is unavailable or undesirable, a *mobile consultant* may instead perform an on-site appraisal, however, only one type of appraisal is required. In particular, if neighbourhood of the property will is marked as *irregular* by the IT system, an on-site appraisal is required. An on-site appraisal requires access to the property and therefore an appointment with the owner.

We shall shortly formalise this process as a DCR graph. First, let us define exactly what is such a graph:

Definition 1 (DCR Graph [15]). *A* DCR graph *is a tuple* $(\mathsf{E}, \mathsf{R}, \mathsf{M})$ *where*

– E *is a finite set of (labelled)* events*, the nodes of the graph.*
– R *is the edges of the graph. Edges are partitioned into five kinds, named and drawn as follows: The* conditions $(\rightarrow\bullet)$*,* responses $(\bullet\rightarrow)$*,* milestones $(\rightarrow\diamond)$*,* inclusions $(\rightarrow+)$*, and* exclusions $(\rightarrow\%)$*.*
– M *is the* marking *of the graph. This is a triple* $(\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$ *of sets of events, respectively the previously executed* (Ex)*, the currently pending* (Re)*, and the currently included* (In) *events.*

When G is a DCR graph, we write, e.g., $\mathsf{E}(G)$ *for the set of events of G, as well as, e.g.,* $\mathsf{Ex}(G)$ *for the executed events in the marking of G.*

The mortgage application process is formalised as a DCR graph in Fig. 1. The events of the process are nodes in the graph. Each event (node) has certain attributes, indicated graphically on the node: Assess loan application and Budget screening approve are initially pending, shown by the blue exclamation mark—they are the initial goals of the process. The label of the event provides both the role and name of the event. In the present paper all events have unique labels, so we will identify the event and its label.

Constraints between activities are represented as edges in the graph. DCR graphs have five kinds of constraints; we shall see all five in this example. First, *conditions*. To be screened, the budget must first have been submitted. This is require by the condition relation $(\rightarrow\bullet)$ between Submit budget and Budget screening approve.

Second, *milestones*. As long as the budget is awaiting screening, the application cannot be assessed. The milestone relation $(\rightarrow\diamond)$ from Budget screening approve to Assess loan application ensures this.

Third, *response*. The customer may have an error in his submitted budget and submit a new one. Even if Assess loan application has already been executed, the blue *response* $(\bullet\rightarrow)$ from Submit budget makes it a requirement to repeat that assessment. We call such a "required activity" *pending*.

Unlike the condition relation, an event constrained by a milestone can become blocked again, for example in our case, if a new budget is submitted then a new

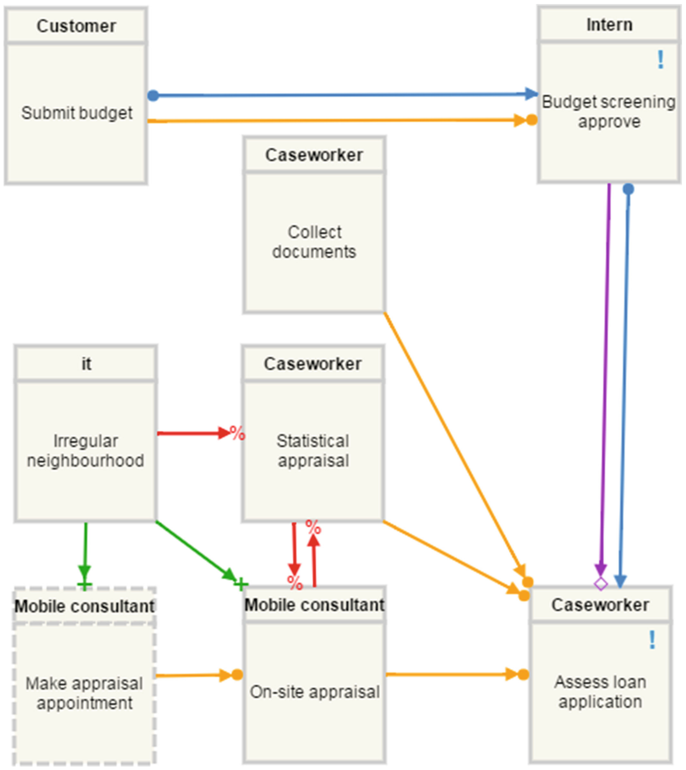

Fig. 1. Declarative DCR model of a mortgage application process (Color figure online)

screening is required, denoted by the response relation (•→) from Submit budget to Budget screening approve. If this occurs, the activity Budget screening approve becomes pending and the activity Assess loan application becomes blocked again. A new screening approval of the budget also requires a new assessment, denoted by the response relation between Budget screening approve and Assess loan application.

There are three more conditions for Assess loan application: Collect documents, Statistical appraisal and On-site appraisal. However, the two kinds of appraisals should be mutually exclusive. Hence, the third and fourth relations, *inclusions and exclusion*. The red *exclusions* (→%) between Statistical appraisal and On-site appraisal mean that when either activity is performed, the other is removed from the process.

Exclusions are dynamic and can be reverted: When the it system registers that the property is in an irregular neighbourhood Statistical appraisal is excluded, On-site appraisal is *included* by the green arrow (→+) (in case a statistical appraisal was already performed and removed it), as is Make appraisal appointment. Make appraisal appointment is a condition for On-site appraisal, but is initially excluded (denoted by the dashed border) and therefore does not block doing an on-site appraisal, only after the IT system marks the neighbourhood as irregular does it become included and a requirement for the on-site appraisal.

We proceed to give the operational semantics of DCR graphs; to answer the question: "What does it mean to *run* a DCR graph"? First, the notion of an event being *enabled*, ready to execute.

Notation. For a binary relation $\rightarrow \subseteq X \times Y$ and set Z, we write "$\rightarrow Z$" for the set $\{x \in X \mid \exists z \in Z.\, x \rightarrow z\}$, and similarly for "$X \rightarrow$". For singletons we usually omit the curly braces, writing $\rightarrow e$ rather than $\rightarrow \{e\}$.

Definition 2 (Enabled Events). *Let* $G = (\mathsf{E}, \mathsf{R}, \mathsf{M})$ *be a DCR graph, with marking* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$. *We say that an event* $e \in \mathsf{E}$ *is* enabled *and write* $e \in \mathsf{enabled}(G)$ *iff (a)* $e \in \mathsf{In}$, *(b)* $\mathsf{In} \cap (\rightarrow\!\bullet e) \subseteq \mathsf{Ex}$, *and (c)* $\mathsf{In} \cap (\rightarrow\!\diamond e) \subseteq \mathsf{E} \backslash \mathsf{Re}$.

That is, enabled events (a) are included, (b) their included conditions are already executed, and (c) have no included pending milestones. Note that enabledness can be determined by considering the marking of the event itself and its immediate conditions and milestones.

The enabled events for the DCR Graph in Fig. 1 are: Submit budget, Collect documents, Statistical appraisal, On-site appraisal and Irregular neighbourhood.

Definition 3 (Execution). *Let* $G = (\mathsf{E}, \mathsf{R}, \mathsf{M})$ *be a DCR graph with marking* $\mathsf{M} = (\mathsf{Ex}, \mathsf{Re}, \mathsf{In})$. *Suppose* $e \in \mathsf{enabled}(G)$. *We may* execute e *obtaining the* resulting *DCR graph* $(\mathsf{E}, \mathsf{R}, \mathsf{M}')$ *with* $\mathsf{M}' = (\mathsf{Ex}', \mathsf{Re}', \mathsf{In}')$ *defined as follows.*

1. $\mathsf{Ex}' = \mathsf{Ex} \cup e$
2. $\mathsf{Re}' = (\mathsf{Re} \backslash e) \cup (e \bullet\!\!\rightarrow)$
3. $\mathsf{In}' = (\mathsf{In} \backslash (e \rightarrow\!\%)) \cup (e \rightarrow\!+)$

That is, to execute an event e one must: (1) add e to the set Ex of executed events; (2) update the currently required responses Re by first removing e, then adding any responses required by e; and (3) update the currently included events by first removing all those excluded by e, then adding all those included by e.

Definition 4 (Transitions, Runs, Traces). *Let G be a DCR graph. If* $e \in \mathsf{enabled}(G)$ *and executing e in G yields H, we say that G has* transition on e to *H and write* $G \longrightarrow_e H$. *A* run *of G is a (finite or infinite) sequence of DCR graphs G_i and events e_i such that* $G = G_0 \longrightarrow_{e_0} G_1 \longrightarrow_{e_1} \ldots$. *A* trace *of G is a sequence of labels of events e_i associated with a run of G. We write* $\mathsf{runs}(G)$ *and* $\mathsf{traces}(G)$ *for the set of runs and traces of G, respectively*

Not every run or trace represents an acceptable execution of the graph: We need also that every response requested is eventually fulfilled or excluded.

Definition 5 (Acceptance). *A run* $G_0 \longrightarrow_{e_0} G_1 \longrightarrow_{e_1} \ldots$ *is* accepting *iff for all n with* $e \in \mathsf{In}(G_n) \cap \mathsf{Re}(G_n)$ *there exists $m \geq n$ s.t. either $e_m = e$, or* $e \notin \mathsf{In}(G_m)$. *A* trace is accepting *iff it has an underlying run which is.*

Acceptance tells us which workflows a DCR graph accepts, its *language*.

Definition 6 (Language). *The* language *of a DCR graph G is the set of its accepting traces. We write* $\mathsf{lang}(G)$ *for the language of G.*

We exemplify the operational semantics of DCR graphs with a run of the model.

1. After executing the event Irregular neighbourhood in Fig. 1 this event is marked as executed, the event Make appraisal appointment becomes included and the event Statistical appraisal becomes excluded. Afterwards the event Make appraisal appointment will be enabled, but the events Statistical appraisal and On-site appraisal will no longer be enabled; the former because it is no longer included and the latter because a condition that was previously excluded is now included.
2. Executing the event Make appraisal appointment will mark it as executed, therefore satisfying the condition to On-site appraisal and making this event enabled again. Executing On-site appraisal will satisfy its condition to Assess loan application, but this event will not become included as there is still an unsatisfied included condition and blocking milestone.
3. Executing Collect documents will satisfy the remaining included condition (note that the condition from Statistical appraisal is no longer relevant as it was excluded).
4. Executing Submit budget will satisfy its condition to (and enable) Budget screening approve, it will also make this event a pending response (but since it already was a pending response this has no noticeable effect).
5. Executing Budget screening approve will remove it from the set of pending responses and thereby satisfy the milestone relation to Assess loan application, enabling it.
6. Finally, executing Assess loan application will remove the pending response on this event, meaning that there are no pending responses left and making the graph accepting. Note that it is not required to end the process at this point, it would for example be possible to execute Submit budget again, once more requiring Budget screening approve and thereafter Assess loan application.

3 GSM Schemas

In this Section, we recall the formal syntax and semantics of GSM [10]. Since we focus on GSM features that are similar to DCR graphs, we omit data attributes, hierarchy, and consider only external events that signify stage completions.

3.1 Syntax

A GSM schema defines the life cycles of artefacts. To simplify the presentation, we focus on the life cycle of a single artefact here.

Definition 7. *A* GSM schema *is a tuple $\Gamma = (Ev, Stg, Mst, R)$, where*

- *Ev is a set of events that can occur; each event is a stage completion event that refers to a stage in Stg or an internal change event;*

– *Stg is a set of stages;*
– *Mst is a set of milestones;*
– *R is a set of rules, defined in Definition 8;*

GSM schemas are governed by rules, also known as sentries. Rules can refer to events denoting that a stage or a milestone has changed value. If a is a stage or a milestone, then $+a$ (resp. $-a$) denotes that a becomes true (resp. false). Such changes are generated by the system in performing B-steps, whereas stage completion events are generated by the environment.

Definition 8. *A rule $r \in R$ has the form* **on** e **if** $cond$ **then** $\odot\, a$*, where* $\odot \in \{+, -\}$*. The* **on** *part is optional. The event e is either a stage completion event or a change event $+a$ or $-a$ for some milestone or stage a. The condition cond is a boolean constraint that only refers to milestones and stages. We call e the* trigger *of φ and cond the* condition *of φ. The* **then** *part signifies the change: $+a$ means that a becomes true, while $-a$ means that a becomes false. For a rule $r =$* **on** *e* **if** *cond* **then** *$\odot\, a$, we let $trigger(r) = e$, $condition(r) = cond$, and $action(r) = \odot a$.*

A stage or milestone x is referenced *by a rule r if x occurs in the* **on** *or* **if** *parts. A stage or milestone x is* triggered *by a rule r if x occurs in the* **then** *part. Note that triggering may be negative, i.e., action $a = -x$.*

To illustrate these definitions, consider the partial GSM schema in Fig. 2 and the rules in Table 1. The rules define for each stage and each milestone a when the stage/milestone becomes true $+a$ (odd numbered rules) and false $-a$ (even numbered rules). Note the inter-dependencies between the rules. For instance, if rule R5 fires, it enables rule R3.
The GSM fragment expresses that:

– Submit budget is opened as soon as possible, but only once.
– If Submit budget completes, then the milestone Budget submitted is achieved, stage Submit budget is closed, and the stage Budget screening approve is opened.
– If Budget screening approved completes, then the milestone Budget screening approve is achieved, and the stage Budget screening approve is closed.

Note that this example GSM fragment is *more* restrictive than the DCR graph model, where both Submit budget and Budget screening approve can be executed repeatedly. A better GSM fragment can be derived from the DCR graph of the previous Section through the translation presented in the next.

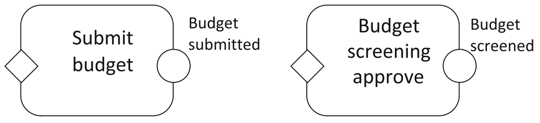

Fig. 2. Part of GSM schema of mortgage application process

Table 1. Rules for GSM model in Fig. 2 (*not* equivalent to Fig. 1)

R1	**if**	true	**then**	+Submit budget
R2	**on**	+Budget submitted	**then**	−Submit budget
R3	**if**	Budget submitted	**then**	+Budget screening approve
R4	**if**	Budget screened	**then**	−Budget screening approve
R5	**on**	C:Submit budget	**then**	+Budget submitted
R6	**on**	+Submit budget	**then**	−Budget submitted
R7	**on**	C:Budget screening approve	**then**	+Budget screened
R8	**on**	+Budget screening approve	**then**	−Budget screened

3.2 Semantics

A snapshot Σ of a GSM schema is a tuple (S, M), where $S \subseteq Stg$ and $M \subseteq Mst$. A stage completion event e is applicable to Σ if e refers to a stage $s \in S$.

The semantics of a GSM schema is event-based: If a stage completion event occurs in a snapshot, the system takes a B-step in response, in which a set of "relevant rules" is fired.

Definition 9. (Relevant Rules). *Let e be a stage completion event. A rule in R is relevant for e if it may be fired in the subsequent B-step. The set of relevant rules R_e is defined inductively as follows*

- *each sentry of the form* **on** *e* **if** *condition is in R_e;*
- *if a sentry in R_e triggers a stage or milestone a, then each sentry that references a is in R_e.*

Each relevant rule that is fired results in a change of the stage or milestone that is triggered by the rule. To ensure that the rules have maximal effect, the rules are evaluated in a pre-specified order. A rule r_1 is evaluated before r_2, written $r_1 \prec r_2$ if r_1 triggers a stage or milestone that is referenced in r_2. For instance, if $r_1 =$ **on** e **then** $+m_1$ and $r_2 =$ **on** f **if** m_1 **then** $+S_2$, then $r_1 \prec r_2$ because of m_1 triggered by r_1 and referenced by r_2.

There are two healthiness constraints on GSM schema [10]. First, the induced $\prec$ ordering must be acyclic. This ensures that when an event e is processed, the set R_e of relevant rules can be evaluated and fired one by one according to the order specified by $\prec$, until eventually a snapshot is reached in which no rule is relevant and can be fired.

Second, the set of R_e of relevant rules should not specify contradictory effects, i.e. there are no two rules in R_e such that one rule has $+a$ and the other rule $-a$ as effect. If the set of rules is consistent, then each stage and milestone will change at most once during a response to an event (toggle-once property [5]). Toggle-once can also been ensured by defining a more liberal constraint that is, however, more intricate [10] and not needed for the purpose of this paper.

Definition 10. ***(Consistency).*** *A set R_e of relevant rules is* consistent *if the induced $\prec$ ordering is acyclic and the rules do not specify contradictory effects. A GSM schema Γ has consistent rules if for each event e its set of relevant rules R_e is consistent.*

Rules in B-steps are evaluated relative to snapshots Σ and a set I of input events. We write $(\Sigma, I) \models$ **on** e **if** *cond* **then** $\odot\, a$, if $e \in I$ and $\Sigma \models$ *cond.*

Let $(\Sigma, I) \models r$, where $r =$ **on** e **if** *cond* **then** $\odot\, a$. The effect of rule r on Σ, denoted $apply(\Sigma, r)$, is $\Sigma' = (S', M')$ where:

- if $a \in S$ and $\odot = +$ then $S' = S \cup \{a\}$ and $M' = M$;
- if $a \in S$ and $\odot = -$ then $S' = S \backslash \{a\}$ and $M' = M$;
- if $a \in M$ and $\odot = +$ then $S' = S$ and $M' = M \cup \{a\}$;
- if $a \in M$ and $\odot = -$ then $S' = S$ and $M' = M \backslash \{a\}$.

We next define B-steps.

Definition 11 (B-step, Run [10]). *Let $\Gamma = (Ev, Stg, Mst, S, R)$ be a GSM schema, let Σ, Σ' snapshots of Γ, and e a stage completion event that is applicable to Σ, and let R_e be the set of relevant rules. The tuple (Σ, e, Σ') is a* B-step *of Γ if there is a sequence $\Sigma_0 = \Sigma, \Sigma_1, \Sigma_2, \ldots, \Sigma_n = \Sigma'$ of snapshots of Γ, such that $e \in Stg_{\Sigma}$ and each Σ_i is the result of applying a rule $r \in R_e$ to Σ_{i-1}, so $\Sigma_i = apply(\Sigma_{i-1}, r)$, for each $i \in [1..n]$ and the ordering of the applied rules is compatible with $\prec$.*

A GSM run *is a sequence $\Sigma_0, \Sigma_1, .., \Sigma_n$ of snapshots interleaved with a sequence of events $e_0, e_1, .., e_n$ such that for each pair Σ_i, Σ_{i+1} of snapshots, where $i \geq 0$, $(\Sigma_i, e_i, \Sigma_{i+1})$.*

4 Translating DCR to GSM

We define a translation from DCR graphs to GSM schemas. First, we define how GSM stages and milestones are derived from a DCR graph. Next, we define how GSM rules are derived from a DCR graph.

4.1 Defining GSM Stages and Milestones

We first need to interpret the DCR concept of an event in terms of GSM concepts. The obvious interpretation is to see a DCR event as being similar to a GSM event. However, each GSM event is either a stage completion event or an internal change event. Since a DCR event relates to an activity in DCR graphs, we interpret a DCR event as a GSM stage completion event. The corresponding stage is inferred from the label of the DCR event. For instance, the DCR event Submit budget in Fig. 1 translates into a GSM completion event C:Submit budget, which signals that stage Submit budget has completed.

So each DCR event e maps into a GSM stage s_e with stage completion event C:s_e. We shall arrange our encoding such that an *enabled* DCR event e has its corresponding stage s_e *active*. Execution of the DCR event will correspond to completion of the stage. We shall see below how a stage s_e is opened precisely when e becomes enabled.

Next, we define for a DCR event e, which corresponds to a GSM stage s_e, three different milestones. These milestones are needed to capture the different possible states of that event, according to DCR graphs.

1. An "executed" milestone m_e^{exec} is achieved when the stage s_e has completed for the first time. It stays true even after reopening of that stage.
2. A "response" milestone m_e^{res} is false when the stage s_e must be opened in the future. When the stage is subsequently opened, it becomes true, i.e., the response has been given.
3. An "inclusion" milestone m_e^{inc}, which is achieved when the corresponding stage is relevant. If the stage is not relevant, its execution is out of scope. Inclusion milestones represent varying scopes of the process, depending on the specific execution.

The use of these milestone is somewhat unconventional, compared to traditional GSM schemas. They are necessary to have the GSM schema reflect DCR notions of dynamic change of rules.

4.2 Defining GSM Rules

We first analyse the rules needed for opening and closing stages. A stage s_e should be open iff the underlying DCR event e is enabled. We derive the following three conditions, based on Definition 2.

First, an enabled DCR event e must be included. In the GSM translation, the milestone m_e^{inc} should be achieved. E.g., in Fig. 1 stage On-site appraisal can only open if milestone $\mathsf{On\text{-}site\ appraisal}^{\mathsf{inc}}$ is achieved.

Second, for a DCR event e to become enabled, each predecessor event f of e that is a condition for e must have occurred, but only if f is included. In the translated GSM schemas, this means that stage s_e is only opened if for each DCR constraint $f \rightarrow\!\bullet\ e$, if the milestone m_f^{inc} has been achieved, then milestone m_f^{exec} has been achieved too. E.g., in Fig. 1 event Make appraisal appointment is a condition for On-site appraisal. Hence stage On-site appraisal can only open if milestone $\mathsf{Make\ appraisal\ appointment}^{\mathsf{inc}}$ is not achieved (corresponding to event Make appraisal appointment not being included) or if milestone $\mathsf{Make\ appraisal\ appointment}^{\mathsf{inc}}$ is achieved and $\mathsf{Make\ appraisal\ appointment}^{\mathsf{exec}}$ is achieved too. Translating the DCR graph in Fig. 1 leads to stage On-site appraisal with guard:

$$\begin{aligned}&\mathsf{On\text{-}site\ appraisal}^{\mathsf{inc}}\\ &\quad\wedge(\mathsf{Make\ appraisal\ appointment}^{\mathsf{inc}} \Rightarrow \mathsf{Make\ appraisal\ appointment}^{\mathsf{exec}}).\end{aligned}$$

Third, for a DCR event e to become enabled, for each predecessor event f of e that is a DCR milestone for e, if f is included, then f must not be pending, i.e., f does not have to be executed in the future. In the translated GSM schemas,

Table 2. Definition of rule set R

Rule ID	Generated by	Rule definition
R1	$\forall e, f.\ e \rightarrow+ f$	**on** e **then** $+m_f^{\mathsf{inc}}$
R2	$\forall e, f.\ e \rightarrow\% f \wedge \neg e \rightarrow+ f$	**on** e **then** $-m_f^{\mathsf{inc}}$
R3	$\forall e.\ \neg\ (e \bullet\!\!\rightarrow e)$	**on** e **then** $+m_e^{\mathsf{res}}$
R4	$\forall e, f.\ e \bullet\!\!\rightarrow f \wedge e \neq f$	**on** e **then** $-m_f^{\mathsf{res}}$
R5	$\forall e$	**on** e **then** $+m_e^{\mathsf{exec}}$
R6	$\forall e$	**if** $\mathsf{enabled}(e)$ **then** $+s_e$
R7	$\forall e$	**if** $\neg\mathsf{enabled}(e)$ **then** $-s_e$

this means that stage s_e is only opened if for each DCR constraint $f \rightarrow\!\diamond\, e$, if the GSM milestone m_f^{inc} has been achieved, then GSM milestone m_f^{res} has been achieved too (recall that m_f^{res} signifies that stage s_f does not have to executed in the future). For instance, in Fig. 1 event Budget screening approve is a DCR milestone for Assess loan application. Therefore, stage Assess loan application can only open if either GSM milestone Budget screening approve$^{\mathsf{inc}}$ is not achieved or if GSM milestones Budget screening approve$^{\mathsf{inc}}$ and Budget screening approve$^{\mathsf{res}}$ are both achieved.

Altogether, these three conditions dictate, for a given DCR event e, when the corresponding stage s_e should be open. We capture these three conditions for a DCR event e in the predicate $\mathsf{enabled}(e)$, which states under what condition stage s_e opens:

$$\mathsf{enabled}(e) = m_e^{\mathsf{inc}} \wedge \bigwedge_{f \rightarrow\!\bullet e} (m_f^{\mathsf{inc}} \Rightarrow m_f^{\mathsf{exec}}) \wedge \bigwedge_{f \rightarrow\!\diamond e} (m_f^{\mathsf{inc}} \Rightarrow m_f^{\mathsf{res}})$$

If the predicate $\mathsf{enabled}(e)$ becomes true, stage s_e opens. If the predicate $\mathsf{enabled}(e)$ is no longer true, stage s_e needs to close.

Table 2 summarises the rules discussed so far. Rules R1-R5 define when the milestones m^{inc}, m^{res} and m^{exec} are achieved and invalidated, respectively. Rules R6 and R7 define when stages are opened and closed.

4.3 Formal Translation

Having discussed the ingredients of the translation, we present it in its entirety:

Definition 12. *Let* $G = (\mathsf{E}, \mathsf{R}, \mathsf{M})$ *be a DCR graph. We define the* corresponding GSM schema $[\![G]\!] = (Ev, Stg, Mst, R)$ *where*

- $Ev = \mathsf{E}$;
- $Stg = \{s_e \mid e \in \mathsf{E}\}$;
- $Mst = \{m_e^{\mathsf{exec}} \mid e \in \mathsf{E}\} \cup \{m_e^{\mathsf{inc}} \mid e \in \mathsf{E}\} \cup \{m_e^{\mathsf{res}} \mid e \in \mathsf{E}\}$; *and*
- *rules* R *are defined as indicated in Table* 2.

Assume the DCR marking is $\mathsf{M} = (\mathsf{Ex}, \mathsf{In}, \mathsf{Re})$*. We define the* corresponding snapshot $\Sigma_{[\![G]\!]}$ *as follows.*

- $\Sigma_{[\![G]\!]} \models s_e \Leftrightarrow e \in \mathsf{enabled}(G)$*;*
- $\Sigma_{[\![G]\!]} \models m_e^{\mathsf{exec}} \Leftrightarrow e \in \mathsf{Ex}$*;*
- $\Sigma_{[\![G]\!]} \models m_e^{\mathsf{inc}} \Leftrightarrow e \in \mathsf{In}$*;*
- $\Sigma_{[\![G]\!]} \models m_e^{\mathsf{res}} \Leftrightarrow e \notin \mathsf{Re}$.

In words: The events of $[\![G]\!]$ are simply the DCR events of G. Each such event has an associated stage s_e, which is open iff e is enabled in G. Moreover, e also has associated milestones $m_e^{\mathsf{exec}}, m_e^{\mathsf{inc}}$, and m_e^{res}, modelling the executed, included, and response states of e.

Note that the milestone for response is true iff e *does not* have a pending response. This is in accordance with GSM intuition where a milestone is true if we have achieved some goal; if e is pending, we have yet to achieve that goal.

We prove that the translation preserves DCR semantics. First, the next key Lemma states that consistency is guaranteed by the translation.

Lemma 1. *For any DCR graph G, the rules R of $[\![G]\!]$ are consistent.*

We next show that the encoding both preserves and reflects semantics, i.e., the GSM Schema $[\![G]\!]$ has exactly the same behaviour as the DCR graph G.

Lemma 2. *Let G be a DCR graph. Then for any event e of G we have*

1. *If $G \longrightarrow_e G'$ then $\Sigma_{[\![G]\!]} \longrightarrow_e \Sigma_{[\![G]\!]'}$; and*
2. *If $\Sigma_{[\![G]\!]} \longrightarrow_e \Sigma'$ then there exists exactly one G' such that $G \longrightarrow_e G'$ and $[\![G']\!] = \Sigma_{[\![G']\!]}$.*

It follows that trace semantics is preserved:

Theorem 1. *Let G be a DCR graph. G has a trace $\alpha = e_0, e_1, \ldots$ iff $[\![G]\!]$ has a trace $\bar{\alpha} = e_0, e_1, \ldots$.*

Proof. By induction on the length of the trace using Lemma 2.

4.4 Accepting Runs

Note that the notion of "accepting run" from DCR does not have a correspondent in the semantics of GSM, and so cannot be encoded into the GSM semantics: GSM does not make a distinction between "complete" and "incomplete" runs or workflows. It is, however, straightforward to transport the notion of acceptance from DCR to GSM as part of the translation: Simply stipulate that a run is accepting if whenever at step i a milestone m_e^{res} is not achieved yet m_e^{inc} is, then at some subsequent step $j > i$ either m_e^{res} is achieved or m_e^{inc} is not.

Corollary 1. *Let G be a DCR graph. G has an* accepting *trace α iff $\bar{\alpha}$ of Theorem 1 is accepting in the above sense.*

4.5 Prototype

The translation can be improved by removing unnecessary milestones, as explained in an accompanying technical report [9]. A prototype implementation of the translation is available at http://dcr.itu.dk/icsoc16. The prototype allows the user to input a DCR graph G (using existing mechanics of the DCR Workbench), and produces in response a CMMN 1.1 XML serialisation of the translated model $[\![G]\!]$. For readability, this output is generated in terms of the trimmed translation, suppressing semantically pointless milestones. As the non-finalised CMMN 1.1 standard does not support the full rule schemas of GSM, the output assumes certain extensions, e.g., that the expression language allows references to the achieved-state of milestones.

5 Conclusion

In this paper we introduced a formal mapping from DCR Graphs to GSM schemas. We showed that for any DCR Graph, a semantically equivalent GSM schema exists, and that the notion of acceptance of DCR graphs can be recovered in GSM schemas. This means that, when extended with the right acceptance criteria, GSM schemas are at least as expressive as DCR graphs[1]. An important practical application of the mapping is the possibility of deriving consistent GSM schemas from a given set of rules formalised in a DCR model, which provides a clear advantage over existing approaches where consistency of a GSM schema is typically checked after-the-fact by model checking. It also makes an important first step in relating the DCR notation to the CMMN standard, thereby increasing the industrial applicability of DCR Graphs.

In future work we intend to also develop the reverse mapping from GSM schemas to DCR Graphs. Because GSM schemas have a strong data-centric aspect to them, the addition of data concepts to the DCR notation is critical for obtaining a meaningful such mapping. While some initial work on adding data to DCR exists [18,21], many questions remain open. In addition the proposed technique for deriving consistent GSM schemas from a rule-based DCR model needs to be developed in more detail. Finally, to be able to relate DCR Graphs to the CMMN standard, a mapping from GSM to CMMN is required. While such a mapping may appear to be straightforward given the large influence GSM has had over the development of the standard, a lack of a formally defined semantics for CMMN presently hampers its development.

[1] with unique labels, see comments after Definition 1.

References

1. van der Aalst, W.M.P., Pesic, M.: DecSerFlow: towards a truly declarative service flow language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
2. BizAgi, et al.: Case Management Model and Notation (CMMN), v1, OMG Document Number formal/2014-05-05, Object Management Group, May 2014
3. Web Services Business Process Execution Language (BPEL), Version 2.0 (2007). http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
4. Business Process Model and Notation (BPMN), Version 2.0 (2011). http://www.omg.org/spec/BPMN/2.0/PDF
5. Damaggio, E., Hull, R., Vaculín, R.: On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. Inf. Syst. **38**, 561–584 (2013)
6. Debois, S., Slaats, T.: The analysis of a real life declarative process. In: CIDM 2015, pp. 1374–1382 (2015)
7. Debois, S., Hildebrandt, T., Slaats, T.: Safety, liveness and run-time refinement for modular process-aware information systems with dynamic sub processes. In: Bjørner, N., Boer, F. (eds.) FM 2015. LNCS, vol. 9109, pp. 143–160. Springer, Heidelberg (2015)
8. Debois, S., Hildebrandt, T.T., Marquard, M., Slaats, T.: Hybrid process technologies in the financial sector. In: BPM 2015 (Industry track), pp. 107–119 (2015). http://ceur-ws.org/Vol-1439/paper9.pdf
9. Eshuis, R., Debois, S., Slaats, T., Hildebrandt, T.: Deriving consistent GSM schemas from DCR graphs (full version). IT University of Copenhagen (2016). http://itu.dk/people/debois/tr.pdf
10. Eshuis, R., Hull, R., Sun, Y., Vaculín, R.: Splitting GSM schemas: a framework for outsourcing of declarative artifact systems. Inf. Syst. **46**, 157–187 (2014)
11. Eshuis, R., Van Gorp, P.: Synthesizing data-centric models from business process models. Computing **98**(4), 345–373 (2016)
12. Exformatics: Dcrgraphs editor and simulator. http://DCRGraphs.net
13. Gonzalez, P., Griesmayer, A., Lomuscio, A.: Verifying GSM-based business artifacts. In: Proceedings of the 2012 IEEE 19th International Conference on Web Services (ICWS), pp. 25–32. IEEE Computer Society (2012)
14. Heath, F., Vaculín, R., Hull, R.: Barcelona: a design and runtime environment for modeling and execution of artifact-centric business processes. In: Proceedings of the 9th International Conference on Business Process Management, BPM (2011)
15. Hildebrandt, T.T., Mukkamala, R.R.: Declarative event-based workflow as distributed dynamic condition response graphs. In: PLACES, pp. 59–73 (2010)
16. Marin, M., Hull, R., Vaculín, R.: Data centric BPM and the emerging case management standard: a short survey. In: Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 24–30. Springer, Heidelberg (2013)
17. Marquard, M., Shahzad, M., Slaats, T.: Web-based modelling and collaborative simulation of declarative processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM. LNCS, vol. 9253, pp. 209–225. Springer, Heidelberg (2015)
18. Mukkamala, R.R.: A formal model for declarative workflows: dynamic condition response graphs. Ph.D. thesis, IT University of Copenhagen, June 2012
19. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. Int. J. Coop. Inf. Syst. **24**(1) (2015). http://dx.doi.org/10.1142/S021884301550001X

20. Sadoghi, M., Jergler, M., Jacobsen, H., Hull, R., Vaculín, R.: Safe distribution and parallel execution of data-centric workflows over the publish/subscribe abstraction. IEEE Trans. Knowl. Data Eng. **27**(10), 2824–2838 (2015)
21. Slaats, T., Mukkamala, R.R., Hildebrandt, T., Marquard, M.: Exformatics declarative case management workflows as DCR graphs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 339–354. Springer, Heidelberg (2013)
22. Solomakhin, D., Montali, M., Tessaris, S.: Formalizing guard-stage-milestone meta-models as data-centric dynamic systems. Technical report (2012)
23. Swenson, K.D.: Mastering the Unpredictable: How Adaptive Case Management will Revolutionize the Way that Knowledge Workers Get Things Done. Meghan-Kiffer, Tampa (2010)