On the Equivalence of Incremental and Fixpoint Semantics for Business Artifacts with Guard-Stage-Milestone Lifecycles

Elio Damaggio[1], Richard Hull, Roman Vaculín[2]

[1] University of California, San Diego(elio@cs.ucsd.edu)
[2] IBM T.J. Watson Research Center ({hull,vaculin}@us.ibm.com)

Abstract. Business artifacts (BAs, or artifacts) are used to model conceptual entities that are central to guiding the operations of a business, and whose content changes as they move through those operations. The recently introduced Guard-Stage-Milestone (GSM) meta-model for artifact lifecycles is declarative in nature, and allows concurrent execution of long-running (possibly human-executed) activities. Modularity is incorporated through the use of hierarchical clustering of activities. The GSM operational semantics is based on a variant of Event-Condition-Action (ECA) rules, which are used to control the start and termination of individual and composite activities. This paper introduces, in an abstract setting, three different and provably equivalent formulations of the GSM operational semantics. The semantics is specified in terms of how a single external event is incorporated into the current "snapshot" (i.e. full description) of a running execution of an artifact model. The "incremental" formulation corresponds to the sequential application of the ECA-like rules in response to the event; the "fixpoint" formulation characterizes the mathematical properties of pairs of snapshots corresponding to the full impact of incorporating the event; and the "closed-form" formulation captures the fixpoint one in terms of first-order logic. The paper introduces a formally specified well-formedness condition on GSM models that guarantees the equivalence of the three formulations while permitting all of the commonly arising patterns for using GSM constructs to model business operations.

1 Introduction

There is increasing interest in frameworks for specifying and deploying business operations and processes that combine both data and process as first-class citizens. One such approach is called Business Artifacts (BA) (or simply "artifacts"), and has been studied by a team at IBM Resarch for several years [21, 6, 20]. Artifacts are key conceptual entities that are central to the operation of part of a business and that change as they move through the business's operations. An artifact type includes both an *information model* that uses attribute/value pairs to capture, in either materialized or virtual form, all of the business-relevant data about artifacts of that type, and a *lifecycle model*, that specifies the possible ways that an artifact of this type might progress through the business, and the ways that it will respond to events and invoke external services, including human activities. The recently introduced [15] Business Entities[3] with Guard-Stage-Milestone Lifecycles meta-model[4] (abbreviated simply with "GSM") provides a substantially declarative approach for specifying artifact lifecycles that supports parallelism and modularity, with an operational semantics based on a variant of Event-Condition-Action (ECA) rules. The current paper presents the formal foundations for the GSM operational semantics. This includes specification of a well-formedness condition on GSM models, and proving the equivalence of three different formulations of the semantics.

As described in [15], the core motivation leading to GSM has been to create a meta-model for specifying business operations and processes that:

1. Will help business-level stakeholders gain insight and understanding into their business operations;
2. Is centered around intuitively natural constructs that correspond closely to how business-level stakeholders think about the operations of their business;
3. Can provide a high-level, abstract view of the operations, and gracefully incorporate enough detail to be executable;
4. Can support a spectrum of styles for specifying business operations and processes, from the highly "prescriptive" (as found in, e.g., BPMN) to the highly "descriptive" (as found in Adaptive Case Management systems); and
5. Can serve as the target into which intuitive, informal, and imprecise specifications of the business operations (e.g., in terms of "business scenarios") can be mapped.

The fundamental research challenge underlying the development of GSM has been to find a meta-model that on the one hand incorporates the business-level constructs in an intuitive and flexible manner, and on the other hand supports a precise semantics that can support both implementation and mathematical investigation. The current paper introduces the mathematical basis for the GSM meta-model, and presents three provably equivalent formulatons for the core operational semantics of GSM.

There are four key elements in the GSM meta-model: (a) *Information Model* for business artifacts, as in all variations of the artifact paradigm; (b) *Milestones*, which correspond to business-relevant operational objectives, and are achieved (and possibly invalidated) based on triggering events and/or conditions over the information models of active artifact instances; (c) *Stages*, which correspond to clusters of activity intended to achieve milestones; and (d) *Guards*, which control when stages are activated, and as with milestones are controlled through triggering events and/or conditions. Multiple stages of an artifact instance may

[3] The notion of Business Entity (with Lifecycle) in [15] and other recent papers is the same as one of Business Artifact the we use in this work.

[4] Following the tradition of UML and related frameworks, we use here the terms 'meta-model' and 'model' for concepts that the database and workflow research literature refer to as 'model' and 'schema', respectively.

be active at the same time, which enables the modeling of parallel activity such as when two human performers are simultaneously conducting different tasks in connection with the same artifact instances. Hierarchical structuring of the stages supports a rich form of modularity.

The operational semantics for GSM is specified in terms of how a single "incoming event" is incorporated into the current "snapshot" (i.e., description of all relevant aspects at a given moment of time) of a GSM system. This semantics is based on a variant of the Event-Condition-Action (ECA) rules paradigm, and is centered around *GSM Business steps* (or *B-steps*), which focus on the full impact of incorporating the incoming event. In particular, the focus is on what milestones are achieved or invalidated, and what stages are opened and closed, as a result of this incoming event. Changes in milestone and/or stage status are treated as "internal events", and can trigger further status changes in the snapshot. Intuitively, a B-step corresponds to the smallest unit of business-relevant change that can occur to a GSM system.

The semantics for B-steps have three equivalent formulations, each with their own value. These are:

Incremental: This corresponds roughly to the incremental application of the ECA-like rules, provides an intuitive way to describe the operational semantics of a GSM model, and provides a natural, direct approach for implementation.

Fixpoint: This provides a concise "top-down" description of the effect of a single incoming event on an artifact snapshot. It is useful for developing alternative implementations for GSM, and optimizations of them; something especially important if highly scalable, distributed implementations are to be created.

Closed-form: This straight-forward rewriting of the fixpoint formulation provides a characterization of snapshots and the effects of incoming events using a first-order logic formula. This permits the application of previously developed verification techniques to the GSM context. (The previous work, [4, 10, 7], assumed that services were performed in sequence, whereas in GSM services and other aspects may be running in parallel.)

This paper formally defines these three formulations of the semantics, and shows that they are equivalent for GSM models that satisfy a well-formedness condition. The development is in some ways reminiscent of logic programming [17], and the well-formedness condition, which is based on an acyclicity condition, can be viewed as providing a kind of stratification on the ECA-like rules. For ease of presentation, and without fundamentally compromising the applicability of the results, this paper uses a common restriction that puts the focus on a single artifact instance. The exposition here is brief; full details of these results, in the context of multiple BA types and instances, are presented in [8] (see also [16].)

Organizationally, Section 2 introduces the abstract GSM framework used in the current paper, through both an informal example and a series of formal definitions. Section 3 presents the notion of B-steps, the well-formedness condition, and the three formulations of GSM operational semantics. Section 4 discusses how the equivalence theorem can be used to reduce GSM-based verification problems to existing techniques for sequential BA meta-models, [4, 10, 7] may be applied. Section 5 overviews related work and Section 6 concludes the paper.

2 Framework

In this section we introduce the theoretical framework for the abstract GSM meta-model (that focuses on a single BA instance), and a motivating example.

Example 1. The motivating example models a design process which includes activities like requirement gathering, engineering design and legal review. We consider five actors in this process: customer, sales manager, request manager, engineering manager, and legal manager. The activities performed are as follows:

Requirement gathering starts when a request with customer's requirements is received from the sales manager. It is considered completed when the customer signs off the requirements and the request manager approves them. The activity might be restarted if the customer wants to change the requirements.

Engineering design It starts when the requirements are ready and the country restrictions has been evaluated as part of the legal review. It is completed when the engineering manager approves the design. It might be suspended and resumed if the customer wants to change some of the requirements.

Legal review It is formed by two main sub-activities: Evaluating country restrictions, and Preparing import documents. The first starts as soon as the request from the sales manager is received and terminates after the approval of the legal manager. Preparing import documents start when the engineering design is completed and can be suspended by the legal manager. □

2.1 Intuitive Framework for GSM Systems

In the general setting, a *BA Service Center* (*BSC*) is used to maintain artifact instances. The BSC acts as a container and supports conventional SOA interfaces (using both WSDL and REST) to interact with an (*external*) *environment*. The most significant part of the environment for the discussion here is its ability to support 2-way service calls, which may be short-lived (as with most automated activities) or long-lived (as with most human-performed activities). The environment can also send 1-way messages into the BSC, and can request that the BSC create new artifact instances.

GSM, as with most BPM, case management, and workflow systems, is intended to support the *management* of business-related activities, but not support the details of executing those activities. Thus, most of the "actual work" in connection with a GSM model is typically performed by actors in the environment.

In this paper we do not address any of the details of the BSC, instead we focus uniquely on the GSM semantics, which, intuitively, specifies how a GSM model reacts to incoming messages and coordinates the process' activities.

Example 2. In Figure 1 we show a graphical representation of the artifact model described in Example 1. At the bottom we have the information model, which includes data attributes and status attributes. Status attributes are associated to the achievement status of a milestone ($true$ or $false$) and to the activity state of stages (open or close). In the top half we have the lifecycle specification. Stages are represented as rounded boxes and are associated to simple or

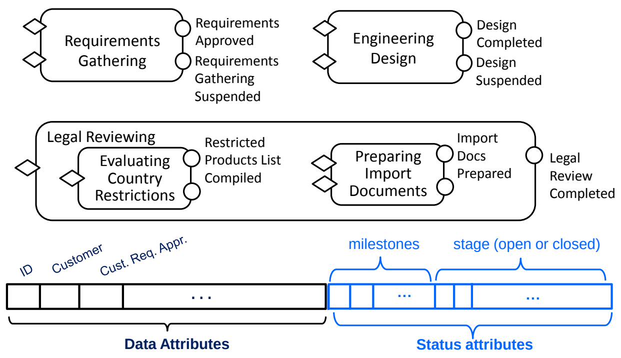

Fig. 1. Graphical representation of the Engineering Requirement BA model

complex activities. The ‘Requirement gathering’ and ‘Engineering design’ activities are modeled as atomic stages and ‘Legal review’ as a composite stage. This choice is due to nature of the ‘Legal review’ activity that includes two business relevant subactivities, ‘Evaluating country restrictions’ and ‘Preparing import documents’. Note how in this case the hierarchical structure of the GSM meta-model allows to express this relationship between activities in the model.

For every stage we show guards as diamonds and milestones as labeled circles Note how, intuitively, one of the ‘Engineering design’ guards can simply be the condition $\mathsf{RequirementsApproved} \wedge \mathsf{RestrictedProductListCompiled}$ that mentions two other milestones in the model. Conditions in guards and milestones can also mention data attributes (e.g. $\mathsf{CustomerRequirementsApproved}$) and incoming events (e.g. $\mathsf{RequirementManagerApproved}$). For instance, the milestone $\mathsf{RequirementsApproved}$ has the following condition, $\mathit{LatestEventType} = \mathsf{RequirementManagerApproved} \wedge \mathsf{CustomerRequirementsApproved}$, and stages ‘Requirement gathering’ and ‘Legal review’ have a guard with condition $\mathit{LatestEventType} = \mathsf{SalesManagerRequest}$, which simply wait for the event that models the request of the Sales manager to start the process. □

2.2 Formal framework

This subsection introduces the formal notion of GSM model that we use to derive this paper’s results.

Definition 1. *A* GSM model *Γ has the form $\Gamma = (Att, \mathcal{S}, M, \mathcal{L})$ where the following hold.*

- *Att is the set of attributes of this type. Att is partitioned into the set Att_{data} of* data attributes *and Att_{status} of* status attributes *(see below).*
- *$\mathcal{S}$ is the set of* stage names, *or simply,* stages.
- *M is the set of* milestone names, *or simply,* milestones.
- *$\mathcal{L}$ is the* lifecycle model *(defined below).*

The following must also hold for $(Att, \mathcal{S}, M, \mathcal{L})$.

1. *The sets $Att, \mathcal{S}, M$ are pairwise disjoint.*
2. *Att includes a attribute called $\mathit{LatestEventType}$ that holds an event type in the set $\mathcal{E}$ (see below). Intuitively, this attribute holds the type of the latest event received by the* BSC.

3. *For each milestone $m \in M$, there is a* milestone status value *attribute in Att_{status}, denoted as m. Intuitively, attribute m indicates whether milestone m is currently true or false.*
4. *For each stage $S \in \mathcal{S}$, there is a* stage status value *attribute in Att_{status}, denoted as $active_S$. Intuitively, attribute $active_S$ indicates whether stage S is currently active (open) or inactive (closed).*

For ease of presentation, we consider attributes to have a single common domain *Dom*, that includes an *undefined* symbol $\perp$, a set of elements that respresent event types, and the two boolean constants $\{true, false\}$. Without loss of generality, when it is clear from the context we might assume that certain attributes are booleans, or event types.

In order to define lifecycles we need to have a language to specify logical conditions for guards and milestones. We consider a condition language $\mathcal{C}$ that refers to attributes in *Att* as variables. For instance, let $\mathsf{completed}$ and price be attributes in *Att* then $\mathsf{completed} = true \wedge \mathsf{price} = \$50$ is a condition. In the rest of the paper, when it is clear from the context that an attribute a holds a boolean value we use the notation a and $\neg\mathsf{a}$ to mean sin $a = true$ and $\mathsf{a} = false$, respectively. For the purposes of the present paper we do not need to specify the exact expressive power of $\mathcal{C}$; we assume, though, that it is a flavor of first-order logic. possibly with restictions on the use of quantifiers. Also, $\mathcal{C}$ can refer to arithmetic constraints and/or to external database relations in the spirit of [7].

The structure of artifact lifecycles is defined next.

Definition 2. *Let $\Gamma = (Att, \mathcal{S}, M, \mathcal{L})$ be an artifact model. The lifecycle model $\mathcal{L}$ of Γ has structure $(Substages, Task, Owns, Guards, Ach, Inv)$ and satisfies the following properties.*

- *Substages is a function from $\mathcal{S}$ to finite subsets of $\mathcal{S}$, such that the relation $\{(S, S') \mid S' \in Substages(S)\}$ creates a forest. The roots of this forest are called* top-level stages*, and the leaves are called* atomic stages.
- *Task is a function from the atomic stages in $\mathcal{S}$ to* tasks *(see below).*
- *Owns is a function from $\mathcal{S}$ to finite, non-empty subsets of M, such that $Owns(S) \cap Owns(S') = \emptyset$ for $S \neq S'$. (This is used to define the hierarchy.)*
- *Guards is a function from $\mathcal{S}$ to finite subsets of boolean formulae of $\mathcal{C}$ with no free variables and that refer to any milestone m of the same stage $S \in \mathcal{S}$ only as a top-level conjunct and in negated form (e.g. $\neg m \wedge \gamma$).*
- *Ach is a function from M to the set of boolean formulae of $\mathcal{C}$ with no free variables. For milestone m, the element of $Ach(m)$ is called the* achieving condition *of m.*
- *Inv is a function from M to the set of boolean formulae of $\mathcal{C}$ with no free variables. For milestone m, each element of $Inv(m)$ is called the* invalidating sentry *of m.*

The notions of *parent*, *child*, *ancestor*, and *descendant* stages are defined as usual. Also, note that the additional requirements for the conditions used in guards is just a technicality stemming from the close relationship between guards and milestones of the same stage.

We previously said how atomic stages are able to execute activities through the ability to call 2-way external services from the *BSC* to the outside world. We model these service calls as *tasks*. Tasks model both computer-executed tasks as well as human-perfomed ones.

Definition 3. *A* task *is a tuple $\langle I, O, \psi \rangle$, where $I \subseteq Att_{data}$ (*input attributes*), $O \subseteq Att_{data}$ (*output attributes*), and ψ (*postcondition*) is a logical formula in $\mathcal{C}$ referring to attributes in I with normal variables and to attributes in O of the successive snapshot with primed variables.*

Intuitively, postconditions model the effects of the execution of the task on the information model. They model both (i) deterministic tasks (e.g. performing a computation: $x' = x+1$), and (ii) black-box services or human performed tasks (e.g. an attribute is only guarateed to be filled: $x \neq \bot$).

GSM models handle two kinds of incoming events: termination events and message events. The first notify the termination of active atomic stages, the second are unsolicited messages that come from the environment.

Definition 4. *We define:*

- $\mathcal{E}_{term}$ *as the set of* termination event types $\{S_{term} | S$ *is an atomic stage*$\}$;
- $\mathcal{E}_{msg}$ *as the set of* message types, *pairs $\langle O, \psi \rangle$ where $O \subseteq Att_{data}$ and $\psi \in \mathcal{C}$ referring to attributes in O;*
- $\mathcal{E} = \mathcal{E}_{term} \cup \mathcal{E}_{msg}$, *as the set of* event types.

Note that, analogously to tasks, postconditions in message types define (in a non-deterministic way) the payload of the event. Also, we assume that the elements in $\mathcal{E}$ are also element of the domain *Dom*.

3 Semantics

In order to formally define the semantics of a GSM model we have to formalize the concepts of snapshot and environment. The environment is used to model the part of the real-world environment that is necessary to ensure the consistency of the GSM model states in the process. For instance, only tasks that have been started can terminate, and their output has to correspond to the input values with which they were invoked. In the current GSM meta-model, each atomic stage can have at most one open occurrence at a given time (although it may have multiple occurrences that occur sequentially through time). It follows that the environment can just keep a copy of the values of the input attributes for each open atomic stage in the model.

Definition 5. *The* environment *for GSM model Γ is the set of attributes $\Omega = \{x_S \mid S$ is an atomic stage and x is an input attribute of the task in $S\}$*

In order to streamline the presentation, in the rest of the paper we assume that the environment is correctly updated throughout the execution of the GSM model. Until now, we informally referred to the process as an evolution of the GSM model through a series of states. We formalize this idea using the concept of snapshot.

Definition 6. *A* snapshot *of a GSM model Γ is an assignment function from $Att \cup \Omega$ to values in D, where (i) Att_{status} are assigned only to $\{true, false\}$, (ii) LatestEventType is assigned to values in $\mathcal{E}$, and (iii) it satisfies the following invariants:*

- GSM-1: Milestone false for active stage. *If a stage S owns a milestone m, then if $active_S = true$ then m has to be false.*
- GSM-2: No activity in closed stage. *If a stage S has a child stage S', and $active_S = false$, then $active_{S'} = false$.*

(In some variations of the GSM meta-model [16, 14], a third invariant *GSM-3* is assumed, which states that for each stage S, at most one of its milestones can be true. This invariant is enforced by some form of syntactical restriction on the milestone achieving conditions. To streamline presentation here, and without loss of generality, we do not consider *GSM-3* here.)

3.1 GSM Business steps (B-steps)

The operational semantics for GSM are focused on the notion of B-steps, which capture impact of a single incoming event occurrence e on a snapshot Σ of a GSM model Γ. In particular, a *GSM Business step*, or *B-step*, is a 3-tuple having the form (Σ, e, Σ'), where

1. Σ is the *previous* snapshot.
2. e is an ground occurrence of an incoming event type associated with Γ.
3. Σ' is the *next* snapshot.

and where certain restrictions apply (given below).

Intuitively, Σ' corresponds to the result of incorporating e into Σ, and incorporating all changes to Σ implied by the guards, milestone achievers, and milestone invalidators of Γ. Also, we assume that the artifact instance will send task invocation messages to the environment corresponding to the atomic stages that were opened as a result of the incorporation of e. Generation of events is not further detailed in the description of the semantics.

Although the creation of Σ' from Σ and e may take a non-empty interval of clock time, in the formal model this is considered instantaneous.

Intuitively, B-steps are considered to be natural "units of business-relevant change". In naturally arising cases, a B-step will capture all of the effects of incorporating a single incoming event, including changes to milestone and stage status. While the construction of a snapshot Σ' from snapshot Σ and event e will be treated as a "black box" from the perspective of the business, the values of Σ and Σ' are considered business relevant. This perspective on B-steps will lead to some requirements on how they are defined (given below).

Example 3. In this example we will describe the intuitive meaning of B-steps on the BA model in Example 2. Upon receiving an event of the type SalesManagerRequest the first guard (with condition *LatestEventType* = SalesManagerRequest) of 'Requirement gathering' opens the stage and invokes the associated task. In the same B-step, the composite stage 'Legal review' is opened by one its guards with condition *LatestEventType* = SalesManagerRequest. This makes all stages inside 'Legal review' eligible. In particular, the guard of the stage 'Evaluating country restriction' with condition

$\neg$RestrictedProductListCompiled opens its stage since the mentioned milestone is false, and invokes the associated task. The result of the B-step is the opening of three stages and the launching of two tasks. A single B-step can affect even more part of the model as a result of a single event. Let us assume that 'Requirement gathering', 'Engineering design', and 'Evaluate country restriction' are all completed, and 'Preparing import document' is running. Now an event of type CustomerChangeRequest arrives, which triggers a guard of 'Requirement gathering'. As we will see later, no milestone can be achieved when a stage is open, so the milestone RequirementsApproved is invalidated. The milestones Designcompleted and ProductListCompiled both have $\neg$RequirementsApproved as an invalidating condition, which result in their invalidations. It follows that the milestone PreparingImportDocumentsSuspended is achieved though a condition $\neg$DesignCompleted. Note how a single event now put the process in a state where three milestones have to be achieved again and a running activity has been suspended because it depends on invalidated work.

We provide now the intuitive framework for the incremental formulation of the GSM semantics. Suppose that Σ, e are given. A sequence $\Sigma = \Sigma_0, \Sigma^e = \Sigma_1, \Sigma_2, \ldots, \Sigma_n = \Sigma'$ of pre-snapshots is constructed, where (a) Σ^e corresponds to the "immediate effect" of e on Σ (defined shortly); (b) each Σ_i, $i > 1$ corresponds to the application of an ECA-like rule; and (c) Σ' corrresponds to the snapshot obtained at the end of the B-step, Each step in this computation is called a *micro-step*.

Remember that GSM models handle two kinds of *incoming events* (termination events and message events), each with its own kind of payload. Also, a system may only receive termination events that refer to stages that are currently open, and the payload is assumed to satisfy the postcondition of the invoked target. A more rigorous formalization of this notion is in [8].

Intuitively, the "immediate effect" of an incoming event e on snapshot Σ, is to update the attribute *LatestEventType* and the values of all data attributes directly affected by the payload of e. Formally:

Definition 7. *Event $e = \langle E, p \rangle$ is* applicable *to snapshot Σ if either*

1. *$E \in \mathcal{E}_{msg}$, or*
2. *$E = S_{term}$ is a termination event for an atomic stage S whose associated service is $\langle I, O, \psi \rangle$, and $\Sigma, p \models active_S \wedge \psi[I/I_S]$, with Σ assigning values to non-primed attributes, p to primed ones, and ψ' be ψ where each variable $x \in I$ is substituted with x_S (i.e. the input in the environment for stage S).*

Definition 8. *The* immediate effect *of an applicable event $e = \langle E, p \rangle$ with output attributes O on snapshot Σ is the snapshot Σ^e derived from Σ as follows: For each $a \in O$, $\Sigma^e(a) = p(a)$, and for each $a \notin O$, $\Sigma^e(a) = \Sigma(a)$*

In this paper we provide three different formulations for the B-step relation (Σ, e, Σ') introduced above. B-steps are combined to create the possible executions of a BA model defined as follows.

	Basis	Prerequisite	Antecedent	Consequent
Explicit rules				
PAC-1	Guard: for each stage S, for each guard φ of S. (Include term $active_{S'}$ if S' is parent of S.)	$\neg active_S$	$\varphi \wedge active_{S'}$	$+active_S$
PAC-2	Milestone achiever: For each milestone m of stage S with φ the achieving condition for m.	$active_S$	φ	$+m$
PAC-3	Milestone invalidator: For each milestone m of stage S with φ the invalidating condition for m.	m	φ	$-m$
Invariant preserving rules				
PAC-4	Guard invalidating milestone: For each guard φ of a stage S, for each milestone m not occuring in a top-level conjunct $\neg m$. (Include term $x.active_{S'}$ if S' is parent of S.)	$\neg active_S$	$\varphi \wedge active_{S'}$	$-m$
PAC-5.	For each milestone m of a stage S.	$active_S$	m	$-active_S$
PAC-6.	For every stage S child of S'.	$active_S$	$\neg x.active_{S'}$	$-active_S$

Fig. 2. Templates for PAC rules associated with a GSM model Γ

3.2 Prerequisite-Antecedent-Consequent Rules

We now introduce the ECA-like rules, which form the basis for all three formulations of the GSM operational semantics.

Definition 9. *A PAC rule is a triple $\langle \pi, \alpha, \gamma(a) \rangle$, where:*

- π *(*prerequisite*), is a formula in $\mathcal{C}$ on attributes in Att,*
- α *(*antecedent*), is a formula in $\mathcal{C}$ on attributes in Att, and*
- $\gamma(a)$ *(*consequent*), is a* change *in the form $+a$ or $-a$, with $a \in Att_{status}$.*

With an abuse of notation we say that a snapshot $\Sigma \models +a$ (resp. $\Sigma \models -a$) if $\Sigma \models a$ (resp. $\Sigma \models \neg a$).

Figure 2 shows templates for the six kinds of PAC rules associated with a GSM model Γ. The explicit rules correspond directly the sentries in Γ, while the implicit rules are used to maintain the GSM invariants.

The set of PAC rules for a GSM model Γ is denoted rules_Γ. A PAC rule $\langle \pi, \alpha, \gamma(a) \rangle$ in rules_Γ is *applicable* to snapshots Σ, Σ' if $\Sigma \models \pi$ and $\Sigma' \models \alpha$. In this case, the result of *applying* the rule is Σ'', which is constructed from Σ' by changing the value of status attribute a according to $\gamma(a)$.

The PAC rule is *applicable* to the sequence $\Sigma = \Sigma_0, \Sigma^e = \Sigma_1, \Sigma_2, \ldots, \Sigma_i$, if it is applicable to Σ, Σ_i; in this case the application of the rule adds Σ_{i+1} to the sequence where Σ_{i+1} is the result of applying the rule to Σ, Σ_i.

As formalized below, the incremental formulation includes restrictions on the ordering of rule application to prevent certain counter-intuitive behaviors. Space limitations prevent a more full discussion (see [8]), but we give brief intuitions here. One desired property of a B-step (Σ, e, Σ') is that it be "intertial", in the sense that a status attribute of Σ does not change in Σ' unless there is a reason for the change, or more specifically, there is a PAC rule that justifies that change. Further, this "justification" should be visible by examining just Σ and

Σ', not intermediate pre-snapshots that might have been involved in computing Σ'. (This forms part of the fixpoint formulation of the GSM semantics.)

Suppose now that stage S_1 owns milestone m_1 with achieving sentry m for some other milesone m; stage S_2 owns milestone m_2 with achieving sentry m_1; and stage S_3 has a guard of form $m_1 \wedge \neg m_2$. Suppose further that at the beginning of some B-step S_1 and S_2 are open, S_3 is closed, and that in some event e has already lead to m. A possible firing sequence for PAC rules would be: achieve m_1; close S_1; achieve the guard of S_3 (since right now m_1 is true and m_2 is false) and open S_3; achieve m_2; and finally close S_2. This is counter-intuitive and violates the inertial property.

An ordering of PAC rules will be devised to prevent such non-inertial results of the incremental application of PAC rules; this in turn will require an acyclicity condition that will be imposed on GSM models, which is captured in the well-formedness condition. It is also desirable to enforce the intuitive *Toggle-Once* principle, which states that in an incremental construction of Σ', a given status attribute may change value once, but may not revert back to the original value. This is one motivation for the Prerequisite component of PAC rules.

3.3 Polarized Dependency Graph and Well-formedness

The Polarized Dependency Graph is intended to capture dependencies between the PAC rules in rules_Γ, and will be used to order the application of rules during the incremental construction of B-steps. This graph incorporates different aspects of the GSM semantics.

Definition 10. *We call* polarized graph *$PDG(\Gamma)$ of a BA model Γ is defined as follows. For each status attribute a in Γ, we have two nodes $\langle +, a\rangle$ and $\langle -, a\rangle$. For each stage S and each of its guards γ, we have a node $\langle S.\gamma\rangle$. For edges we have:*

- *For each PAC-1 rule $\langle \neg active_S, \gamma, +active_S\rangle$ in rules_Γ, let b be a status attribute mentioned in the guard γ, then we have two directed edges $(\langle +, b\rangle, \langle +, S.\gamma\rangle)$ and $(\langle -, b\rangle, \langle +, \gamma\rangle)$, and*
- *For each guard γ of stage S, we have an edge $(\langle +, S.\gamma\rangle, \langle +, active_S\rangle)$. Also, if γ contains a top-level conjunct $\neg m$ with m a milestone of S, we have a directed edge $(\langle -, m\rangle, \langle +, S.\gamma\rangle)$; an edge $(\langle +, S.\gamma\rangle, \langle -, m\rangle)$, otherwise.*
- *For each PAC rule $\langle \pi, \alpha, \gamma(a)\rangle$ with templates PAC-2 or PAC-3 in rules_Γ, let b be a status attribute mentioned in α, and σ be the sign of $\gamma(a)$, then we have two directed edges $(\langle +, b\rangle, \langle \sigma, a\rangle)$ and $(\langle -, b\rangle, \langle \sigma, a\rangle)$, and*
- *For each PAC-5 rule $\langle active_S, m, -active_S\rangle$ in rules_Γ, we have a directed edge $(\langle +, m\rangle, \langle -, active_S\rangle)$, and*
- *For each PAC-6 rule $\langle active_S, \neg active_{S'}, -active_S\rangle$ in rules_Γ, we have a directed edge $(\langle -, active_{S'}\rangle, \langle -, active_S\rangle)$.*

Definition 11. *We say that an artifact system Γ is* well-formed *if $PDG(\Gamma)$ is acyclic.*

In some cases it is helpful to use a more lenient notion of well-formed, that is based on the acyclicity of all of the *event-relativized* PDGs. For an event type

E, the event-relativized PDG for $\mathit{\Gamma}$ and E is constructed in the same manner as $PDG(\mathit{\Gamma})$, except that a rule (π, α, γ) is not considered if π is an incoming event type different from E. (Although not considered here, the same results of the present work hold for this lenient notion of well-formed.)

3.4 Incremental formulation

We can now formally define the incremental formuation for well-formed systems. Every PAC rule with consequent $\gamma(a)$ is associated with the corresponding node of the polarized graph (e.g. $\langle +, a\rangle$ for a PAC with $\gamma(a) = +a$). Since the polarized graph is acyclic, its topological sort provides a partial order $\prec$ on PAC rules. The main idea of the incremental semantics is that we apply the PAC rules following this partial order.

Definition 12. *Let $\mathit{\Sigma}, \mathit{\Sigma}'$ be snapshots of a well-formed GSM model $\mathit{\Gamma}$ and e an incoming event. Then $(\mathit{\Sigma}, e, \mathit{\Sigma}')$ is in the* incremental transition relation*, denoted $(\mathit{\Sigma}, e, \mathit{\Sigma}') \in \rightarrow_{\mathit{\Gamma}}$ or $\mathit{\Sigma} \rightarrow^{e}_{\mathit{\Gamma}} \mathit{\Sigma}'$ if:*

1. *e is applicable to $\mathit{\Sigma}$,*
2. *for each data attribute a of $\mathit{\Gamma}$, $\mathit{\Sigma}'(a) = \mathit{\Sigma}^e(a)$,*
3. *$\mathit{\Sigma}'$ is the result of applying PAC rules of $\mathsf{rules}_{\mathit{\Gamma}}$ to $\mathit{\Sigma}^e$ in an order compatible with $\prec$, and*
4. *no PAC rule in $\mathsf{rules}_{\mathit{\Gamma}}$ can be applied to $\mathit{\Sigma}, \mathit{\Sigma}'$.*

Well-formedness guarantees the following important result.

Theorem 1. *For well-formed GSM model, given a snapshot $\mathit{\Sigma}$ and an applicable event $\hat{e}$, there always exists a unique $\mathit{\Sigma}'$ s.t. $\mathit{\Sigma} \rightarrow^{e}_{\mathit{\Gamma}} \mathit{\Sigma}'$.*

Proof sketch. We assume by contradiction that there exists two different outcomes of the B-step. It is clear from Definition 12 that the only way they can differ is on the status attributes (point 5 and 4). Since both snapshots are derived from the same $\mathit{\Sigma}^e$, we consider the earliest different attribute according to $\prec$. But since that attribute value is changed only by PAC rules that depend only on attributes that are smaller w.r.t. $\prec$ and thus are the same in both snapshots, we have a contradiction. $\square$

3.5 Fixpoint formulation

The fixpoint formulation for the GSM semantics is analogous to the fixpoint characterization used in logic programming. (Intuitively, our framework has some similarities to stratification. It also has similarities to logic programming without negation, because we enforce the Toggle-Once principle.)

The formulation is based on two properties of triples $(\mathit{\Sigma}, e, \mathit{\Sigma}')$.

Definition 13. *Let $\mathit{\Gamma}$ be well-formed. The triple $(\mathit{\Sigma}, e, \mathit{\Sigma}')$ is* compliant *with respect to $\mathsf{rules}_{\mathit{\Gamma}}$ if*

- *$\mathit{\Sigma}'$ and $\mathit{\Sigma}^e$ agree on all data attributes, and*
- *for each PAC rule $(\pi, \alpha, \gamma(a))$ in $\mathsf{rules}_{\mathit{\Gamma}}$, if $\mathit{\Sigma} \models \pi$ and $\mathit{\Sigma}' \models \alpha$, then $\mathit{\Sigma}' \models \gamma(a)$.*

Definition 14. *Let $\mathit{\Gamma}$ be well-formed. The triple $(\mathit{\Sigma}, e, \mathit{\Sigma}')$ is* inertial *with respect to $\mathsf{rules}_{\mathit{\Gamma}}$ if for each status attribute a: if $\mathit{\Sigma}(a) \neq \mathit{\Sigma}'(a)$ then there is a PAC rule $(\pi, \alpha, \gamma(a))$ in $\mathsf{rules}_{\mathit{\Gamma}}$ such that $\mathit{\Sigma} \models \pi$; $\mathit{\Sigma}' \models \alpha$; and $\mathit{\Sigma}' \models \gamma(a)$.*

Definition 15. *Let Γ and (Σ, e, Σ') be as above. Then the triple is in the* fixpoint formulation relation*, denoted $(\Sigma, e, \Sigma') \in \rightarrowtail_\Gamma$ or $\Sigma \rightarrowtail^e_\Gamma \Sigma'$ if*

1. *e is applicable to Σ,*
2. *for each data attribute a, $\Sigma'(a) = \Sigma^e(a)$*
3. *(Σ, e, Σ') is inertial and compliant w.r.t.* rules_Γ

3.6 The Closed-Form Formulation

The closed-form formulation of the GSM semantics is based on the observation that the properties of compliance and inertial can be captured in a first-order formula (extended to work with relation types and path expressions) that refers to structures having the form (of some suitable "flattening" of) (Σ, e, Σ'). The construction of the overall formula is reminiscent of constructions used for logic programming with negation, and in particular, when characterizing "negation as failure" [17]. Due to space limitations, we do not present the construction of the logic formula Θ_Γ here; please see [8].

Definition 16. *Let Γ and (Σ, e, Σ') be as above. Then the triple is in the* closed-form formulation relation*, denoted $(\Sigma, e, \Sigma') \in \twoheadrightarrow_\Gamma$ or $\Sigma \twoheadrightarrow^e_\Gamma \Sigma'$ if (Σ, e, Σ') respect to* rules_Γ *and*

1. *e is applicable to Σ,*
2. *for each data attribute a, $\Sigma'(a) = \Sigma^e(a)$*
3. *$(\Sigma, e, \Sigma') \models \Theta_\Gamma$*

We can now state the main result.

Theorem 2. *For well-formed systems, let Σ, Σ' and e as above, the following are equivalent: (i) $\Sigma \rightarrow^e_\Gamma \Sigma'$, (ii) $\Sigma \rightarrowtail^e_\Gamma \Sigma'$, and (iii) $\Sigma \twoheadrightarrow^e_\Gamma \Sigma'$.*

Proof sketch. First we note that (*ii*) and (*iii*) are syntactic manipulations of the same logical formulas. In order to prove (*i*) $\leftrightarrow$ (*ii*), we assume (*i*) and (*ii*) are different, and we focus on the smaller differing status attribute a with respect to $\prec$. It is then easy to prove that either (a) a PAC rule changes a in (*i*) iff the compliance principle is satisfied, or (b) no PAC rule changes a in (*ii*) iff the inertial principle is satisfied. $\square$

4 Discussion

We already noted how the fixpoint and closed-form formulations are very close. However, while the fixpoint formulation is best suited to specify execution engines for artifact models, enabling the possibility of sound optimization techniques, the closed-form formulation bridges the gap between previous data-centric business process meta-models and artifact models. This is especially important to exploit previous results in the field of automatic verification of business process models [4, 10, 7]. The cited works assumed that the artifact lifecycles can start tasks only in a sequential manner. Tasks are called *services* and are equipped with a precondition that determines when that task is 'eligible' to start, and a postcondition that, analogously to the tasks' postconditions in the GSM models, prescribes a way to modify the values in the business artifact snapshot. This is in contrast to GSM models where all closed atomic stages with a guard evaluating to true are started in a single B-step, and, as a result, many tasks can run at the same time.

If we look at the closed-form formulation, however, we note how the transition relation $\Sigma \twoheadrightarrow^{e}_{\Gamma} \Sigma'$, identifies what is effectively a service in this sequential model. Let us consider snapshots of the form (Σ, e), at every step we can evolve the system by applying a service that has a precondition that forces point 1. of Definition 16 (i.e. e is applicable by Σ) and a postcondition that forces points 2., and 3. (i.e. $\forall_{a \in Adata} \Sigma'(a) = \Sigma^{e}(a)$, and Θ_{Γ}), that can be easily expressed as formulae in Σ, e, Σ'.

As a result, the closed-form formulation provides a direct way to translate GSM models into 'sequential' business process models. This enables researchers focusing on more theoretical aspects to use a simplified model, while keeping intact the scope of their results.

5 Related work

The GSM meta-model is a natural evolution from the earlier practical artifact meta-models [21, 5, 22], but using a declarative basis, and supporting modularity and parallelism within artifact instances. GSM draws on previous work on ECA systems [19], and develops a specialized variant useful for data-centric BPM.

There is a strong relationship between the artifact paradigm and Case Management [26, 9, 27]; both approaches focus on conceptual entities that evolve over time, and support *ad hoc* styles of managing activities. The GSM framework provides a formal operational semantics, which may be relevant to some Advanced Case Management systems.

Formal analysis of artifact-based business processes in various contexts has been reported in [11, 12, 4, 10, 7]. Unlike the GSM meta-model, all of these assume that the external services are performed in a sequential fashion. Notably, [10] permits infinite data domains with order, and an underlying (static) database; and [7] extends to include arithmetic operations. Both works characterize bounds on expressive power that support decidability of LTL-FO properties.

The AXML Artifact model [2, 18] supports a declarative form of artifacts using Active XML [1] as a basis. Hierarchy based on the XML structure is used in AXML; this contrasts with the stage hierarchy in GSM. Automatic verification for AXML is studied in [3]. DecSerFlow [25] is inherently more declarative than GSM, which can be viewed as a reactive system that permits the use of a rich rules-based paradigm for determining, at any moment in time, what activities should be performed next. The work in [13] falls in the same category. There is a loose correspondence between the artifact approach and proclets [24]; both approaches factor a BPM application into "bite-size" pieces that interact through time. Proclets do not emphasize the data aspect, and support only message-based proclet interaction. GSM also permits interaction of artifact instances through condition testing and internal event triggering.

6 Conclusion

The Business Artifact paradigm provides a compelling data-centric approach for modeling and deploying business operations and processes, that is now incorporated into IBM products and service offerings. The Guard-Stage-Milestone

(GSM) meta-model for artifacts represents a substantial extension of previous artifact meta-models, that supports a declarative style, parallelism within artifact instances, and modularity through hierarchical constructs. In addition to implementing the an execution engine prototype, the IBM Research team has created several GSM models, including one that is being used for an IBM internal pilot application [23]. As a result, we believe that the the core constructs of the GSM meta-model, and the essential aspects of the GSM operational semantics, are robust and will not undergo significant changes.

This paper provides the core mathematical foundations for GSM in an abstract setting. The paper introduces a well-formedness condition for GSM models that supports naturally arising patterns of business operations while helping to ensure key mathematical properties. The paper demonstrates the equivalence of three formulations for the GSM operational semantics, and describes how the result can be used to adapt verification results obtained for sequential artifact meta-models to the GSM context.

The development and results in this paper provide the foundation for a number of research and practical investigations into the use of declarative artifact-based frameworks, including: (1) extension to multiple artifact types and instances (see [14, 16]); (2) incorporating business-level transactional guarantees; (3) incorporating roles, performers, teams, accountability, and delegation; (4) development of practical verification systems for GSM; (5) development of optimized implementations for GSM, including highly distributed, massively scalable ones; and (6) the study of variations and customization of business processes.

References

1. S. Abiteboul, O. Benjelloun, and T. Milo. The Active XML project: An overview. *Very Large Databases Journal*, 17(5):1019–1040, 2008.
2. S. Abiteboul, P. Bourhis, A. Galland, and B. Marinoiu. The AXML Artifact Model. In *Proc. 16th Intl. Symp. on Temporal Representation and Reasoning (TIME)*, 2009.
3. S. Abiteboul, L. Segoufin, and V. Vianu. Static analysis of active XML systems. In *Proc. Intl. Symp. on Principles of Database Systems (PODS)*, 2008.
4. K. Bhattacharya, C. E. Gerede, R. Hull, R. Liu, and J. Su. Towards formal analysis of artifact-centric business process models. In *Proc. Int. Conf. on Business Process Management (BPM)*, pages 288–304, 2007.
5. D. Cohn et al. Siena: From powerpoint to web app in 5 minutes. In *Intl. Conf. on Services Oriented Computing (ICSOC)*, 2008.
6. D. Cohn and R. Hull. Business artifacts: A data-centric approach to modeling business operations and processes. *IEEE Data Engineering Bulletin*, 32:3–9, 2009.
7. E. Damaggio, A. Deutsch, and V. Vianu. Artifact systems with data dependencies and arithmetic constraints. In *Proc. Intl. Conf. on Database Theory (ICDT)*, 2011.
8. E. Damaggio, R. Hull, and R. Vaculín. On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles (full version), 2011. available upon request.
9. H. de Man. Case management: Cordys approach, February 2009. `http://www.bptrends.com/deliver_file.cfm?fileType=publication&file Name=02-09-ART-BPTrends%20-%20Case%20Management-DeMan%20-final.doc.pdf`.

10. A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. Automatic verification of data-centric business processes. In *Proc. Intl. Conf. on Database Theory (ICDT)*, 2009.
11. C. E. Gerede, K. Bhattacharya, and J. Su. Static analysis of business artifact-centric operational models. In *IEEE International Conference on Service-Oriented Computing and Applications*, 2007.
12. C. E. Gerede and J. Su. Specification and verification of artifact behaviors in business process models. In *Proceedings of 5th International Conference on Service-Oriented Computing (ICSOC)*, Vienna, Austria, September 2007.
13. T. Hildebrandt and R. R. Mukkamala. Distributed dynamic condition response structures. In *Programming Language Approaches to Concurrency and Communication Centric Software (PLACES)*, 2010.
14. R. Hull, E. Damaggio, R. De Masellis, F. Fournier, M. Gupta, F. Heath III, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, and R. Vaculín. Business entities with guard-stage-milestone lifecycles: Managing entity interactions with conditions and events. In *Distributed Event-Based Systems (DEBS) 2011*, 2011.
15. R. Hull et al. Introducing the guard-stage-milestone approach for specifying business entity lifecycles. In *Proc. of 7th Intl. Workshop on Web Services and Formal Methods (WS-FM 2010), Revised Selected Papers*, Lecture Notes in Computer Science. Springer, 2010.
16. R. Hull et al. A formal introduction to business entities with guard-stage-milestone lifecycles, Version 0.7, March 14, 2011. Draft IBM Research internal report, available at `http://researcher.watson.ibm.com/researcher/view_page.php?id=1710`.
17. John. W. Lloyd. *Foundations of Logic Programming, 2nd Edition.* Springer, 1987.
18. B. Marinoiu, S. Abiteboul, P. Bourhis, and A. Galland. AXART – Enabling collaborative work with AXML artifacts. *Proc. VLDB Endowment*, 2010.
19. Dennis R. McCarthy and Umeshwar Dayal. The architecture of an active data base management system. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989*, pages 215–224. ACM Press, 1989.
20. P. Nandi et al. Data4BPM, Part 1: Introducing Business Entities and the Business Entity Definition Language (BEDL), April 2010. `http://www.ibm.com/developerworks/websphere/library/techarticles/1004_nandi/1004_nandi.html`.
21. A. Nigam and N. S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.
22. J.K. Strosnider, P. Nandi, S. Kumarn, S. Ghosh, and A. Arsanjani. Model-driven synthesis of SOA solutions. *IBM Systems Journal*, 47(3):415–432, 2008.
23. R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukavirirya. Declarative business artifact centric modeling of decision and knowledge intensive business processes. In *Enterprise Distributed Object Computing Conference (EDOC)*, 2011.
24. W. M. P. van der Aalst, P. Barthelmess, C.A. Ellis, and J. Wainer. Proclets: A framework for lightweight interacting workflow processes. *Int. J. Coop. Inf. Syst.*, 10(4):443–481, 2001.
25. Wil M. P. van der Aalst and Maja Pesic. Decserflow: Towards a truly declarative service flow language. In *The Role of Business Processes in Service Oriented Architectures*, 2006.
26. W.M.P. van der Aalst and M Weske. Case handling: a new paradigm for business process support. *Data Knowl. Eng.*, 53(2):129–162, 2005.
27. W.-D. Zhu et al. Advanced Case Management with IBM Case Manager. Published by IBM. Available at `http://www.redbooks.ibm.com/redpieces/abstracts/sg247929.html?Open`.