Towards Quality-Aware Translations of Activity-Centric Processes to Guard Stage Milestone

Julius Köpke[1,2](✉) and Jianwen Su[1]

[1] Department of Computer Science, UC Santa Barbara, Santa Barbara, USA
su@cs.ucsb.edu

[2] Alpen-Adria Universität, Klagenfurt, Austria
julius.koepke@aau.at

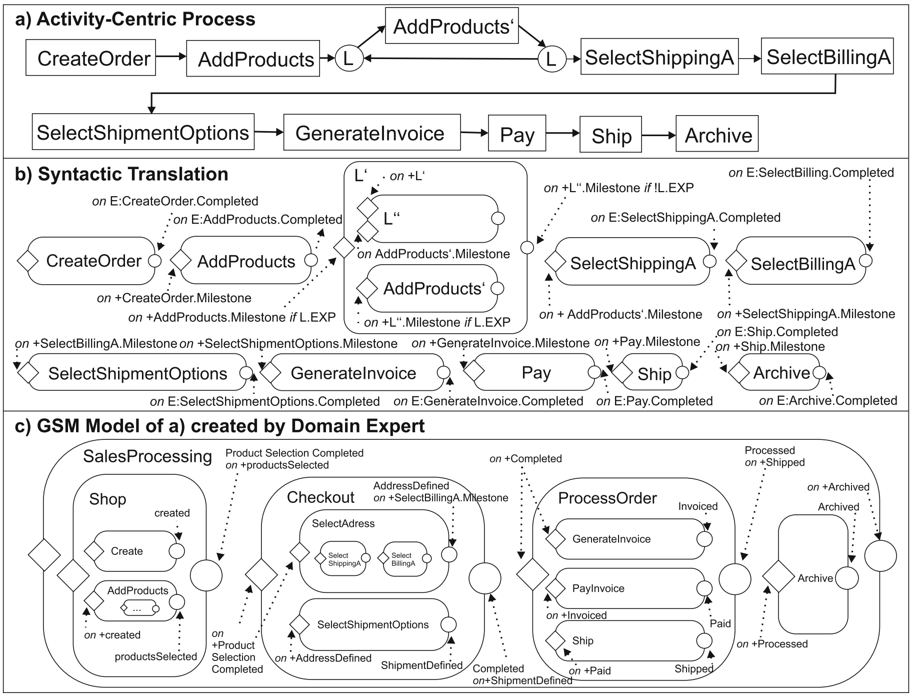
Abstract. Current translation approaches from activity-centric process models to artifact-centric Guard Stage Milestone (GSM) models operate on the syntactic level. While such translations allow equivalent traces (behaviors) of executions, we argue that they generate poor GSM models for the intended audience (including business managers and process modelers). A specific deficiency of these translations is their inability to relate to relevant domain knowledge, especially groupings of activities to achieve well-known business goals cannot be obtained by syntactic translations. Ironically, this is a main principle of GSM models. We developed an initial ontology based translation framework [14] that incorporates the missing knowledge for improved translations. In this paper we further extend this framework with two metrics for the assessment of quality aspects of resulting GSM translations with domain knowledge, propose a novel semantic rewriting algorithm that enhances the quality of GSM translations, and provide an evaluation of the achievable quality for different classes of input processes. Our evaluation shows that maximum quality scores are achievable if semantics and structure of the input processes are well aligned. Given poorly aligned input processes, a translation method can optimize one of the metrics but not both.

Keywords: Process translation · Artifact-centric BPM · Guard Stage Milestone · GSM · Quality metrics

1 Introduction

In contrast to the predominant activity-centric modeling methods (e.g. BPMN) that concentrate on the control-flow between activities, Guard Stage Milestone (GSM) [12] is artifact-centric and defines business processes based on data entities and their declarative life-cycles. With the growing adaption of the artifact-centric modeling paradigm, the need for translations between activity-centric

J. Köpke—Research conducted while visiting UCSB and supported by the Austrian Science Fund (FWF) under grant J-3609-N15.



Note: The syntactic translation algorithm creates a stage L' representing the loop (L). The loop is controlled with an additional control-stage L'' that evaluates the loop condition $L.exp$.

Fig. 1. (a) Input process (b) Syntactic translation (c) Translation of domain expert [14]

and artifact-centric process models gains importance. The fact that GSM provides the basis for the new OMG Case Management Standard (CMMN[1]) further extends its importance. Especially for inter-organizational cooperations, translations between both paradigms become vital. Translation of activity-centric models to artifact-centric models has been studied [7,17]. However, these approaches remain on the syntactic level and create completely flat GSM models not following the basic principles and guidelines of GSM.

Guard Stage Milestone (GSM). We highlight the relevant essentials of GSM here and refer the reader to [5,12] for details. A process is modeled in the form of artifacts, where each artifact has a data schema with data attributes and state attributes, and a life-cycle definition. GSM life-cycles are based on *guards*, *stages*, and *milestones*. In the graphical representation (Fig. 1(b) and (c)), guards are depicted as diamonds, stages as rounded boxes with optional labels, and milestones as circles. A guard defines when a particular stage becomes active, a milestone defines when a stage is completed (e.g. a business goal is reached). Stages can be *atomic* or *composite*. Every atomic stage contains a service that is executed when the stage becomes active. A composite stage contains other stages. The idea of composite stages is to group stages that are executed in

[1] http://www.omg.org/spec/CMMN/.

order to achieve a common goal collectively, i.e. to reach the milestone(s) of their parent stage. Guards and milestones may have labels and are specified as *sentries*. Sentries are defined in the form of Event Condition (over the data schema) Action (ECA) rules of the form "*on event if condition* ", "*on event*", or "*if condition*". Events may be internal (e.g. achieving of a milestone) or external such as the completion event of a service call. Achieving events of sentries are denoted by the prefix "+" and their invalidation by the prefix "−", respectively.

Weaknesses of Syntactic Translations. We discuss weaknesses of syntactic translations with an example. For the activity-centric input process shown in Fig. 1(a), part (b) shows the GSM translation based on a purely syntactic translation algorithm [14]. Part (c) shows a GSM version of (a) potentially created by a domain expert from scratch. The syntactic translation (b) has a number of disadvantages in comparison to (c):

1. Milestones and guards are defined on a solely technical level not relating to any agreed real-world states of data objects nor business goals. For example, the milestone of *Pay* in Fig. 1(b) is defined by the completion of the *Pay* task. In contrast, the domain expert has modeled a stage *PayInvoice* with the milestone *paid* in Fig. 1(c), where *paid* is a well-known state of order objects in the domain and *PayInvoice* is a known activity in the domain.
2. The syntactic translation is mostly flat and lacks nesting of stages based on business goals. In contrast, the domain expert uses nested stages to structure the process based on business goals of the domain. In Fig. 1(c) the activity stages are nested inside the upper-level stages *Shop*, *Checkout*, *ProcessOrder*, and *SalesProcessing*.

The model in Fig. 1(c) not only has advantages for stake-holders, by presenting structured models referring to agreed terms, but also facilitates advanced process monitoring based on abstract stages and business goals of the domain.

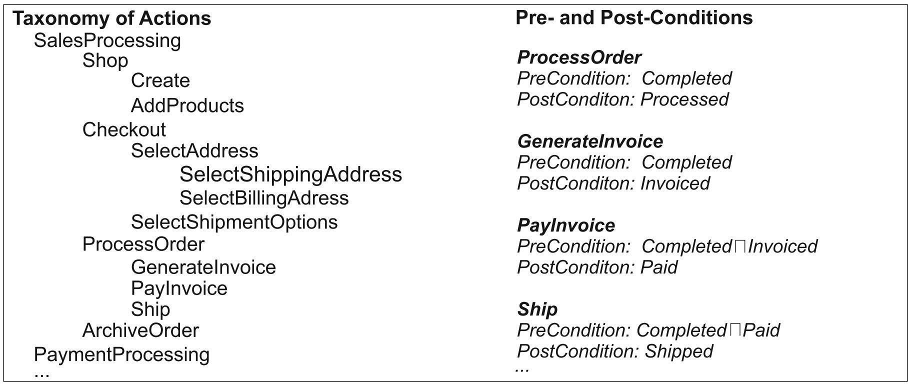
From a general perspective, the model in Fig. 1(c) has a higher "quality" [9,16] than that in Fig. 1(b). A main cause of this quality difference is the different expressiveness of the activity-centric model and the GSM model in the following sense. While GSM allows to define business goals and hierarchies of stages to achieve them, this information is missing in the source models. Consequently, it cannot be added through a purely syntactic translation.

Quality of models [9,16] depends on many aspects, some of which may not have technical formulations. In many application domains, a large part of domain knowledge exists in documents or even with semi-formal languages. Examples include housing management [8], travel industry (e.g. http://www.opentravel.org), the financial/accounting domain [15] and of course the medical domain [2]. For applications in these domains, focusing on ontology "alignment" can improve learning/training of process models, as well as monitoring of *key performance indicators*.

In our earlier work [14] an architecture to tackle this problem was developed. The general idea is to provide the missing domain knowledge for meaningful translations in form of an ontology representing the states of business documents and a taxonomy of state changing actions defining typical part of relations of actions in the domain.

Contributions of this paper. We extend our earlier framework with the following specific contributions:

- Quality metrics for assessing the semantic alignment (Sect. 2) and control-flow complexity (Sect. 3) of GSM translations.
- A rewrite algorithm (Sect. 4) for improving the semantic alignment of a translation.
- Our findings based on the evaluation (Sect. 5) are: Maximum quality metrics of translations are achievable for both metrics if input processes are fully aligned with the domain taxonomy. Having poorly aligned input processes, either semantic alignment or control-flow complexity can be optimized but not both at the same time.



Taxonomy of Actions

- SalesProcessing
  - Shop
    - Create
    - AddProducts
  - Checkout
    - SelectAddress
      - SelectShippingAddress
      - SelectBillingAdress
    - SelectShipmentOptions
  - ProcessOrder
    - GenerateInvoice
    - PayInvoice
    - Ship
  - ArchiveOrder
- PaymentProcessing
- ...

Pre- and Post-Conditions

ProcessOrder
PreCondition: Completed
PostConditon: Processed

GenerateInvoice
PreCondition: Completed
PostConditon: Invoiced

PayInvoice
PreCondition: Completed⊓Invoiced
PostConditon: Paid

Ship
PreCondition: Completed⊓Paid
PostCondition: Shipped
...

Fig. 2. An example taxonomy of actions [14]

2 Semantic Alignment of a GSM Translation

A core element of our translation framework [14] is a *taxonomy of actions* T that defines relationships between actions in the domain. The key idea was to automatically or manually map activities in an input process to the taxonomy T to allow GSM translations with semantic stage nesting. While [14] focused on how an existing process can be matched with the taxonomy, in this paper we assume such a matching given. We first review the taxonomy from [14] and then present a mapping formalism between activity-centric input processes and the taxonomy. Based on the mapping and the taxonomy, we define a novel quality metric for the semantic alignment of GSM translations.

2.1 Taxonomy of Actions

A taxonomy of actions T describes abstract, well agreed actions that result in state changes (e.g. the achievement of business goals) of business entities. The actions are organized in a rooted tree (whose root node is *root*) representing a *part of* hierarchy. The semantics of T is the following: If an action b is defined as a child-action of some action a where $a \neq root$, then b is considered as potentially contributing to achieving the goal of action a (in GSM reaching a milestone). Action b can be used to achieve the goal of a, but it is not required to use b to achieve a in every process or instance. Therefore, T can be considered as a general glossary of actions that can be reused for different processes of the domain. As described in [14] each node in T is additionally annotated with OWL expressions defining pre- and post-states of each action. The annotations are primarily used for matching activities and actions and are out of scope of this paper. An example taxonomy for our previous example is shown in Fig. 2. We assume that the taxonomy is provided as input. It may be created by employing domain ontologies [2,8,15], or it could be derived from GSM process repositories or goal models [4,13].

2.2 Mapping of Activities and Taxonomy of Actions

A mapping between an activity-centric input process G and the taxonomy of actions T is defined by the tuple (M_1, M_2) that is defined below:

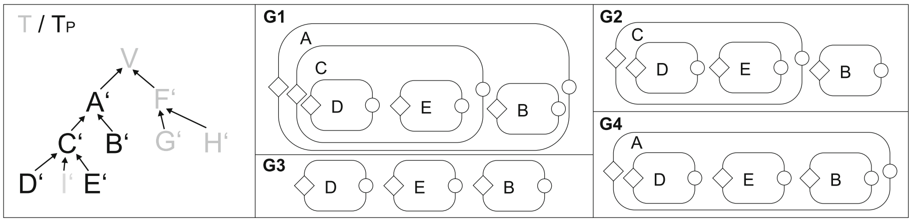


Fig. 3. Taxonomy alignment - correct alignment examples

Definition 1 *(Realizes Mapping).* M_1 is a set of pairs of the form (s, a), where $s \in G.activitySteps$, $a \in T.actions$, $T.actions$ is the set of actions in the taxonomy T and $G.activitySteps$ refers to the set of activities in the input process. A tuple in M_1 defines that the activity s realizes the action a. Each activity can realize zero or one action in T and each action in T can be realized by zero or one activity (partial bijection).

Not requiring a *1:1* mapping is based on the assumption that the taxonomy is not necessarily complete and should be generic to be applicable for different processes. Definition 1 forbids to map more than one activity in the process model to one action that occurs in the same context (same parent) in the taxonomy. However, this limitation can be lifted by extending the taxonomy with additional actions or parent actions.

Definition 2 *(Contributes to Mapping).* The mapping M_2 is also a set of pairs (s, a), where $s \in G.activitySteps$ and $a \in T.actions$, and specifies that a specific activity contributes to the achievement of some action in T. Every activity that realizes an action also contributes to it ($M_1 \subseteq M_2$). M_2 further satisfies the following (quantifiers omitted):

– Activities can only contribute to actions they realize or to ancestors thereof: $(s, a) \in M_2 \Rightarrow (s, a) \in M_1 \vee \exists (s, a') \in M_1$, where a is an ancestor of a'.
– Activities must contribute to common ancestors: $\{(a, a'), (b, b'), (a, a'')\} \subseteq M_2 \Rightarrow (b, a'') \in M_2$ if a'' is a common ancestor of a' and b'.
– No skipping of levels in T: $\{(a, a'), (a, a''')\} \subseteq M_2 \Rightarrow (a, a'') \in M_2$, if $a'' \in T.actions$, a'' is a descendent of a''' and an ancestor of a'.

Definition 3 (*Taxonomy Projection*). For each taxonomy T and each mapping M_2, let *project*(T, M_2) denote a rooted taxonomy (with *root* as the root node) obtained by projecting T onto M_2. A node t of T is also a node of *project*(T, M_2), if $\exists\ (x, t) \in M_2$ for any x. The relative hierarchical order of T is preserved in the projection.

2.3 Assessing Taxonomy Alignment of a Translation

Given a GSM translation of some activity-centric input process with a mapping to a taxonomy of actions, we want to assess how well the stage nesting in the translation corresponds to the taxonomy projection.

General Idea of the Taxonomy Alignment Metric. Figure 3 shows four different GSM translations $G1$–$G4$ of some activity-centric input process (not shown). The taxonomy T and the taxonomy projection T_P are shown on the left. Elements of T that are not in T_P are depicted in grey.

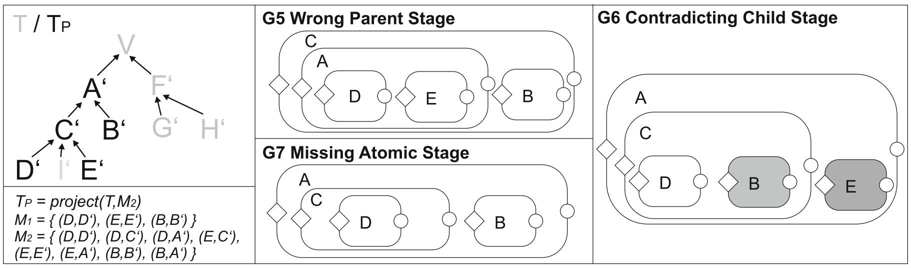

T / TP
V
A'
F'
C'
B'
G'
H'
D'
I'
E'
TP = project(T,M2)
M1 = { (D,D'), (E,E'), (B,B') }
M2 = { (D,D'), (D,C'), (D,A'), (E,C'), (E,E'), (E,A'), (B,B'), (B,A') }
G5 Wrong Parent Stage
C
A
D
E
B
G6 Contradicting Child Stage
A
C
D
B
E
G7 Missing Atomic Stage
A
C
D
B

Fig. 4. Taxonomy alignment - contradictions

Translation $G1$ provides the same nesting as defined in T_P. We consider this as a perfect alignment. $G1$ should get the maximum score of 1. $G3$ does not provide any grouping of atomic stages and should get the minimum score of 0. We do not give credits for the existence of mapped activities since they are trivially part of the translation.

$G2$ and $G4$ are partially aligned but parent actions of T_P are missing in the translations. $G4$ perfectly describes the contribution of B. We therefore assign a credit of 1 for B in $G4$. However, the contributions of D and E are only partially described. We assign a credit of 0.5 for the description of D and of E because only 1 of two abstractions is present in $G4$. $G2$ does not describe the contribution of B at all (no credit for B) and the contributions of D and E are only partially modeled (credits 0.5 for D and for E). We argue that $G4$ should be considered as superior to $G2$ because the contribution of the atomic stages is better described in $G4$. Following this principle we calculate the overall metrics by the average credit for each atomic stage: The metrics of $G4$ is $\frac{0.5+0.5+1}{3} = \frac{2}{3}$ and that of $G2$ is $\frac{0.5+0.5+0}{3} = \frac{1}{3}$.

While the previous example only addressed missing hierarchy levels, Fig. 4 shows example translations, where the nesting in the process model contradicts with the one in the taxonomy projection. In process $G5$ the hierarchy is inverted, where C is a child of A in the taxonomy, A is a child of C in the translation. In $G6$, B is nested under C but it should be nested under A. In $G7$, a mapped atomic stage is missing. Stage E contributes to the achievement of C in the taxonomy but this is not reflected in the process model. We consider this as incorrect since the goal of C may never be achieved without executing E. While credits are assigned separately for each atomic stage, contradictions influence the contributions of multiple atomic stages. We assign a credit of 0 to all atomic stages that are nested in a stage that contains contradictions.

Calculating Alignment Metrics. We first provide preliminary definitions: Corresponding Taxonomy Action of an Atomic Stage: Let G' be a GSM translation of an activity-centric process G with a mapping $M = (M_1, M_2)$ to a taxonomy T. Let s be an atomic stage in G' implementing an activity a of G. If $(a, t) \in M_1$, then t is the corresponding action of s. The corresponding taxonomy action of a composite stage is defined by equivalent stage labels and labels of the actions in the taxonomy. Taxonomy Tree of a GSM translation G', $\mathit{tree}(G')$, is a taxonomy tree representing the stage nesting of G'. The nodes of $\mathit{tree}(G')$ are the corresponding actions of the stages union additional nodes for non-mapped stages of G'. The hierarchy in $\mathit{tree}(G')$ equals the hierarchy in G'. The Hierarchy Path $\mathit{hp}(n, T)$ of a node n in a tree T is a sequence of nodes defined by the path from n to the root, excluding n and the root node. The projection of a hierarchy path a and a hierarchy path b, $\mathit{projectP}(a, b)$ denotes a hierarchy path a', where a' only contains the elements of b while the relative order of elements in a is preserved.

Definition 4 (*Correct Nesting* of a mapped atomic stage s in a GSM translation G' under T and $M{=}(M_1, M_2)$). Let t be the corresponding taxonomy node of the atomic stage s and T_P the taxonomy projection of T under M_2. The atomic stage s is correctly nested into parent stages if $\forall$ action $a \in \mathit{hp}(t, \mathit{tree}(G'))$ where a is an action in $T_P \Rightarrow a$ is an action in $\mathit{hp}(t, T_P)$ and the relative order of actions is equivalent in both paths.

Definition 5 (*Contradiction* of composite stages in G' with the taxonomy projection). A composite stage s_1 contradicts with a taxonomy projection if it contains incorrectly nested atomic stage (Definition 4) or if atomic stages are missing: Let $t_1 \in \mathit{project}(T, M_2)$ be the corresp. taxonomy node of s_1. An atomic stage is missing in s_1 if there exists a descendent t_2 of $t_1 \in \mathit{project}(T, M_2)$, a tuple $(x, t_2) \in M_1$ for some x, but $\nexists s_2$ as a substage of s_1 in G' such that t_2 is the corresp. taxonomy node of s_2.

For calculating the taxonomy alignment score of a (mapped) atomic stage, we assign the value of 0 if the atomic stage is part of a contradicting composite stage. Otherwise, the score is based on the fraction of existing parents in the translation and the number of parents in the taxonomy projection:

Definition 6 (*Taxonomy Alignment* of an atomic stage s). Let G' be a GSM translation of an activity-centric process G and t the corresponding action of s in T under the mapping M_2. If s is part of a composite stage that does not contradict with the taxonomy projection (Definition 5), the score is the fraction of the number of existing mapped abstractions of t in $\mathit{tree}(G')$ and the number of abstractions of t in the taxonomy projection: $\mathit{SemMetricAtomic}(s) = \frac{|\,\mathit{projectP}(\mathit{hp}(t, \mathit{tree}(G')), \mathit{hp}(t, \mathit{project}(T,M_2)))|}{|\,\mathit{hp}(t, \mathit{project}(T,M_2))|}$ otherwise, the score is 0.

Definition 7 (*Taxonomy Alignment* of a GSM translation). The taxonomy alignment metrics of a translation is the mean of the metrics scores of all mapped atomic stages.

Example Calculating the taxonomy alignment for D of $G4$ in Fig. 3:
$\mathit{hp}(D', \mathit{tree}(G4)) = \langle A \rangle, \mathit{hp}(D', \mathit{project}(T, M_2)) = \langle C, A \rangle$
$\mathit{projectP}(\langle A \rangle, \langle A, B \rangle) = \langle A \rangle \rightarrow \mathit{SemMetricAtomic}(D) = \frac{|\langle A \rangle|}{|\langle A,B \rangle|} = \frac{1}{2}$.
In analogy to D: $\mathit{SemMetricAtomic}(E) = \frac{1}{2}$, $\mathit{SemMetricAtomic}(B) = \frac{1}{1} = 1$.
The taxonomy alignment score of $G4$ is: $(\frac{1}{2} + \frac{1}{2} + \frac{1}{1})/3 = \frac{2}{3}$.

Properties of the Metrics: The purpose of the taxonomy alignment metrics is to compare translations of the same input process. The metrics is based on assessing the degree of alignment of each atomic stage. When a translation a achieves better average taxonomy alignment scores for all atomic stages than another translation b, then a gets a better score than b. When the taxonomy projection is balanced this metrics is equivalent to an alternative metrics, which is the number of all provided abstractions of atomic stages divided by the number of possible abstractions of all atomic stages. The result of the alternative approach is different for unbalanced taxonomy projections because atomic stages that are deeper nested have stronger positive or negative impact on the overall alignment score. This behavior should be considered when the metrics is applied to unbalanced taxonomy projections. Which metrics better describes the desired alignment depends on the usage scenario.

We define the quality of a stage nesting (unordered tree) relative to the taxonomy projection (unordered tree). This also makes general tree similarity

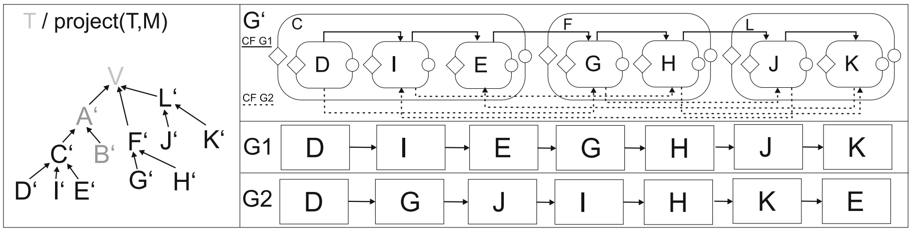


Fig. 5. GSM translation G' for input processes $G1$ and $G2$

approaches such as the tree edit distance (e.g. [1]) possible candidates for metrics. However, beside the problem that the calculation of the minimal tree edit distance is NP-hard for unordered trees, it does not directly produce the desired results: In the example in Fig. 3, the non-weighted tree edit distance between the taxonomy and $G2$ and between the taxonomy and $G4$ are both 1 (adding one node). However, $G4$ better matches the desired stage nesting. Therefore, the edit operations would still need to be weighted based on the number of affected atomic stages and potential contradictions (see Definition 5).

3 Control-Flow Complexity

The previous metrics assesses the existence of stage nestings relative to a taxonomy while ignoring the control-flow between composite stages. However, the control-flow may limit the usefulness of a given stage nesting: frequent switches between sub-processes negatively impact on the understandability of (behaviors of) process models [23,24]. Additionally, the utility of translation for monitoring purposes is limited because numerous stages remain opened at the same time without actually performing tasks in parallel. In GSM switching between sub-processes (composite stages) exists if there is control-flow between atomic stages of different (active) composite stages. To address this, we introduce a "control-flow complexity" metrics of translations. It is based on the usual fan-in and fan-out [11] of modules (stages). We are specifically interested in fan-out of composite stages that are not linked to their closing (non-exit fan-out) and fan-in into already opened composite stages (non-entry fan-in).

Example Figure 5 shows a GSM stage hierarchy G' perfectly aligned with the taxonomy projection on the left. If G' is the result of a translation of $G1$, the control-flow is completely in-line with the stage progression of G' (solid arrows in the top part of G' in Fig. 5). The composite stages C, F, and L are executed in a sequence. There is no non-entry fan-in nor non-exit fan-out control-flow. (For the sake of simplicity the example does not contain control-blocks.) If $G2$ is the input for G', the control-flow (dashed arrows in the bottom part of G' in Fig. 5) is scattered over multiple composite stages that are open in parallel without actually executing tasks in parallel. We denote the parent stage of an atomic stage in subscript. C is opened and (D_C, G_F) opens F not closing C. (G_F, J_L)

opens L not closing F. (J_L, I_C) resumes C. (I_C, H_F) resumes F not closing C. (H_F, K_L) closes F and resumes L. Finally, (K_L, E_C) resumes C and closes L. For stage C as one example this leads to the non-entry fan-in control-flows $(J_l, I_C), (K_L, E_C)$ and the non-exit fan-out control-flows $(D_C, G_F), (I_C, H_F)$.

Control-Flow Complexity for GSM. We define fan-in and fan-out of a stage based on the control-flow graph of the activity-centric input process G. A *control-flow* from an activity a to another b exists if a is a predecessor of b in the graph representation of G and there exists a path from a to b that does not contain any other activity.

Definition 8 (*Non-Entry Fan-In of a Stage* S). Let G' be the GSM translation of an activity-centric process G and S be a composite stage of G'. A *fan-in* of S is a control-flow (a, b), where a corresponds to an atomic stage $\notin S$ and b corresponds to an atomic stage $\in S$. A fan-in (a, b) of S is a *non-entry* if for all permissible instantiation of G some activity $\in S$ is executed before a. The set of all non-entry fan-ins of S is denoted $\mathit{NonEntryFanIn}(S)$. *Fan-out* of S and $\mathit{NonExitFanOut}(S)$ are defined correspondingly.

$\mathit{AvgNonEntryFanIn}(G')$ is the arithmetic mean of $|\mathit{NonEntryFanIn}(S)|$ of all composite stages $S \in G'$. $\mathit{AvgNonExitFanOut}(G')$ is defined similarly. The calculation of the control-flow complexity is realized in analogy to coupling metrics [6]. The values are in the interval of $[\geq 0, < 1]$, where 0 indicates no unwanted switching between active composite stages, near 1 indicates very high numbers of switches on average.

Definition 9 (*Control-Flow Complexity* of a GSM translation G').
$\mathit{controlComplex}(G') = 1 - \frac{2}{1+\,\mathit{AvgNonEntryFanIn}(G')+1+\,\mathit{AvgNonExitFanOut}(G')}$

EXAMPLE In the example in Sect. 3, when considering $G2$ as the input process of G': The non-entry-fan-in of C in G' is $|\{(J, I), (K, E)\}| = 2$, for F and L, we get 1. The non-exit-fan-out of C in G' is $|\{(D, G), (I, H)\}| = 2$, for F and L, we get 1. This leads to an average non-entry-fan-in and non-exit fan-out of $\frac{4}{3}$. Thus, $\mathit{controlComplex}(G') = 1 - 2/(1 + \frac{4}{3} + 1 + \frac{4}{3}) = 0.572$.

When $G1$ is the input we have $\mathit{controlComplex}(G') = 1 - \frac{2}{1+0+1+0} = 0$.

Properties of the Metrics: The purpose of the control-flow complexity metrics is to compare translations with different stage nestings of the same input process regarding unwanted dependencies between active composite stages. Therefore, higher total numbers of non-entry fan-in and non-exit fan-out relative to the number of composite stages must lead to higher complexity values of the metrics. This is guaranteed. The metrics does not assess the control-flow complexity [3,10] of the input process. However, control-blocks in the input process have impact on the potential fan-in and fan-out of composite stages in the translation.

A control-flow is considered non-entry if the stage has certainly been opened before. A more pessimistic and more complex approach would be to calculate

Algorithm 1 . Semantic Rewrite of a GSM Translation

```
1: Method rewriteTranslation
Input: TaxonomyNode node
2: if (node is not root node) then
3:     GsmStage commonAncest = getCommonAncestor(
       getMappedAtomicStages(node));
4:     for all (GsmStage s $\in$ getMappedAtomicStages(node)) do
5:         topStageBefore = ancestorBefore(s,commonAncest);
6:         nestingCandidates $\uplus$ topStageBefore;
7:         checkAtomic $\uplus$ topStageBefore.getAllAtomicStages();
8:     end for
9:     if (allNestable(checkAtomic,getMappedAtomicStages(node)) then
10:        nestStages(nestingCandidates,node,commonAncest);
11:    end if
12: end if
13: for all (TaxonomyNode n $\in$ node.getChildren()) do
14:    rewriteTranslation(node);
15: end for
```

non-entry fan-in based on the probability that some stage has already been opened before and to compute non-exit fan-out correspondingly. However, what metrics better describes problematic control-flows still needs to be decided based on a user-study.

4 Semantic Rewrite Algorithm

Based on the taxonomy alignment metrics, we present an algorithm that rewrites a syntactic translation of an activity-centric input process to enhance its metrics score. The algorithm takes as input an activity-centric process G, a (possibly nested) syntactic translation G' of G, a taxonomy of actions T, and a contributes-to mapping M_2.

The core method *rewriteTranslation*(*TaxonomyNode*) is shown as Algorithm 1. It is first called with the root node of the taxonomy projection and visits the nodes of the taxonomy projection in a depth-first traversal. Unless the current node is the (virtual) root node of the projection it retrieves the common ancestor *commonAncest* stage of all atomic stages that are mapped to the current node of the projection in G'. It then creates the sets *nestingCandidates* and *checkAtomic*, where *nestingCandidates* contains the top-level ancestor stage of each atomic-stage below *commonAncest* and *checkAtomic* contains all atomic stages nested into each stage in *nestingCandidates*. The set *checkAtomic* is used to check if a nesting is possible.

According to the alignment metrics, a nesting is correct if it contains all required atomic stages and it does not contain atomic stages that are not mapped to the current node but to other taxonomy nodes. This check is realized by the Boolean method *nestable*(). If *nestable* returns *true*, a new stage with the label of the current taxonomy node is created as a child stage of the common ancestor and all nodes in *nestingCandidates* are assigned as child stages of the new stage. Finally, guards and milestones are generated for the new stage.

Example Applying Algorithm 1 on $G3$ in Fig. 3, *rewriteTranslation* (project(T, M_2)) $\rightarrow$ *rewriteTranslation*(A'): *CommonAncestor* of D, E, and B is $G3$ itself. The loop in lines 4 to 8 produces the sets *nestingCandidates* $= \{D, E, B\}$ and *checkAtomic* $= \{D, E, B\}$, *allnestable*$(\{D, E, B\}, \{D, E, B\})$ returns *true*. The new stage A' is created under the common ancestor $G3$. The *nestingCandidates*, $\{D, E, B\}$, are set as its child stages. Next, *rewriteTranslation*(C') is called and processed in analogy to A'. Finally, $G3$ equals $G1$ in Fig. 3.

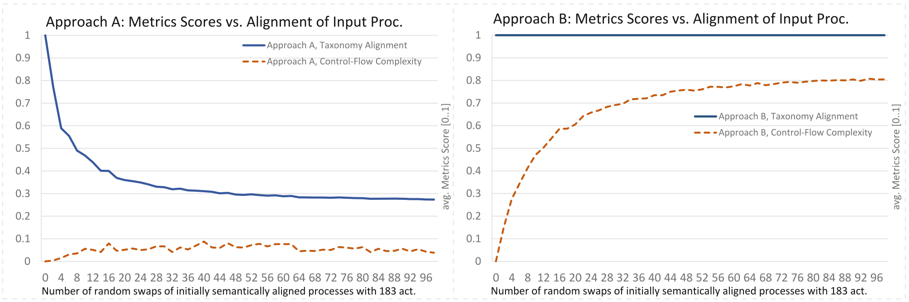


Fig. 6. Metrics scores vs. alignment of input processes. Left: App. A, Right: App. B

Setting Guards and Milestones. Since the rewrite algorithm must not change the permitted traces of executions it should guarantee that every stage that could be opened before the new stage was introduced can still be opened after the new stage is introduced. In GSM, a child stage cannot be opened if the parent stage is closed. Therefore, the new stage must be opened before any of the potentially first executed nested atomic stages may get opened. In principle, we could add a copy of the guards of each potentially first opened stage to the new stage.

When taking a block-structured syntactic translation from [14] as input, control-blocks (xor, par, loops) of the activity-centric input process are represented as composite stages and in a block-structured activity-centric process there is always one block that is evaluated first. Therefore, we use the sentry expression of the substage that represents the first block as the (single) guard sentry expression of the new stage.

For milestones, we apply a similar strategy. In principle, the new stage is completed, e.g. some milestone of it is reached, when no atomic stage within the new stage is open and no atomic stage within the new stage can get opened anymore. However, this may depend on future decisions during the runtime of the process, resulting in potentially complex milestone expressions. In contrast when using the nested syntactic translation of [14] as input there is always one last stage. We use the achievement sentry expression of its milestone as the sentry expression of the new stage's milestone.

Finally, we beautify the generated guards and milestones by rewriting equivalent expressions of child and parent guards/milestones. If a child stage has the exact sentry expression for guards as its parent stage, we update the sentry of the guard to the opening event of the parent stage. If a parent milestone has the exact same sentry condition as a milestone of a child stage, we rewrite the parent milestones sentry to the achieving event of the child milestone.

5 Evaluation

We present an evaluation of achievable metrics scores of the rewrite algorithm (Sect. 4) to assess (1) the influence of existing hierarchy on taxonomy alignment, (2) the influence of rewriting on control-flow complexity, and (3) the balance between control-flow complexity and semantic alignment. We conducted experiments with the rewrite algorithm having two different syntactic translation approaches as input. The combination of our block-based translation approach [14] with the semantic rewrite algorithm is referred to as "Approach A". The combination of the semantic rewrite approach with a simple flat translation is referred to as "Approach B". While Approach A creates complete and potentially enactable translations, Approach B generates partial translations discarding guards and milestones. This is sufficient for the assessment of the achievable quality since the control-flow metrics is based on the control-flow defined in the input processes. Potentially enactable implementations can be based on existing flat translation approaches such as [17,18].

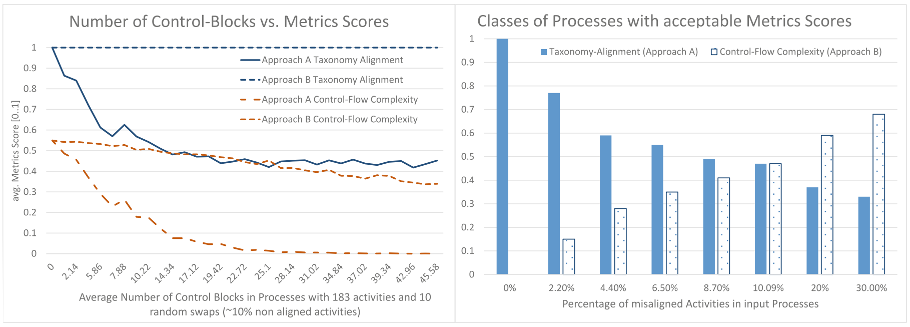


Fig. 7. Left: #control-blocks vs. metrics scores, Right: %-random vs. metrics scores

For our experiments, we have generated semantically aligned block-structured processes and taxonomy mappings based on the *Food Products Chapter* of the well balanced *UN Central Product Classifications Taxonomy.*[2] A block structured input process G is aligned with a taxonomy when each control-block (par, xor, loop) c that contains mapped activities only contains

[2] CPC Ver.2.1 http://unstats.un.org/unsd/cr/registry/regdnld.asp?Lg=1.

mapped activities if all are mapped to the same most specific common ancestor action a in the taxonomy (recursively) and only brothers or descendants of c may also contain activities mapped to a.

Experiment 1. We assess the influence of taxonomy alignment of the input process and the achievable metrics scores for Approaches A and B. We have conducted experiments with varied number of misaligned activities by repeatedly swapping two random activities. For each number of swaps (0 to 96 = completely random), we randomly generated 50 fully semantically aligned initial processes with mappings and applied the swaps. Each process contains 183 activities and on average 19 control-blocks (par, xor, loop). The average metrics scores of 50 processes in relation to the number of swaps for Approach A (on the left) and Approach B (on the right) are shown in Fig. 6.

Experiment 2. In the second experiment we investigate the influence of control-blocks in the input processes on taxonomy alignment and control-flow complexity scores. We generated input processes with 183 activities, 0 control-blocks (only sequences) to 183 activities, 59 control-blocks (par, xor, loop). For each number of control-blocks we generated 50 processes with 10 swapped activities (approx. 20 changed activities or 10 % of the activities are not aligned with the taxonomy). The average metrics scores of 50 processes in relation to the number of control-blocks are shown on the left side of Fig. 7.

5.1 Findings

Finding 1: *Maximum taxonomy alignment scores are achievable*
Both approaches can achieve maximum taxonomy alignment scores. Approach A achieves an alignment score of one, if the input processes are fully aligned with the taxonomy (see 0 swaps at left side of Fig. 6) or if the input processes only contain sequences (see left side of Fig. 7). Approach B constantly produces the maximum alignment score of 1 (see right side of Fig. 6 and left side of Fig. 7). In contrast to Approach A, the non-nested GSM translation used as input for Approach B does not impose any restrictions on the required nesting.

Finding 2: *Optimizing taxonomy alignment scores increases control-flow complexity*
When input processes are not aligned with the taxonomy, Approach B still produces perfect taxonomy alignment scores of 1 (right side of Fig. 6 and left side of Fig. 7). However, the semantic grouping results in an increase of control-flow complexity. As shown on the right side of Fig. 6, the control-flow complexity of the translation results of Approach B grows logarithmical with the percentage of misaligned activities in the input processes. A rough estimate for the control-flow complexity is $complex = 0.1936 \ln(x) + 0.1074$, where x is the percentage of misaligned activities in the input processes. This behavior of increased control-flow complexity score due to semantic nesting also applies for Approach A if the processes contain (mostly) of sequences (left side of Fig. 7). The reason is that nested syntactic translation approach does not perform nesting for sequences.

Another interesting behavior is that the control-flow complexity given a fixed number of swapped activities decreases, when the number of control-blocks grows in the processes (left side of Fig. 7, dotted line). The reason for this behavior is that the number of non-entry fan-in and non-exit fan-out decreases, when more (potential) entry fan-ins and exit fan-outs exists due to conditions.

Finding 3: *Optimizing control-flow complexity decreases taxonomy alignment scores*
Where Approach B produces constantly perfect taxonomy alignment scores, Approach A produces very low control-flow complexity scores (left side of Fig. 6). By not modifying existing nestings of the syntactic translation that translates control-blocks to single-entry, single-exit composite stages, the control-flow complexity stays at a very low level (dotted line in Fig. 6). However, near optimal control-flow complexity comes with strongly reduced taxonomy alignment scores of Approach A, if randomness is added to the input processes. The taxonomy alignment scores decrease potentially with a rough estimation of $score = 0.8081x^{-0.299}$, where x is the percentage of misaligned activities in the input processes. This exponential decrease is induced by the growing misalignment of two trees: The nesting of the syntactic translation and the best-case semantic nesting defined by the taxonomy projection.

By combining Findings 2 and 3 we conclude that given non-perfectly aligned input processes, a translation approach producing control-flow preserving translations (e.g. permitting the same traces of executions) can either optimize taxonomy alignment scores (as approach B) or minimize control-flow complexity scores but cannot achieve both at the same time.

5.2 Input Processes that Achieve Acceptable Alignment/Complexity Scores

We assume that for real-world applications, a process will mostly follow the domain taxonomy. The right side of Fig. 7 shows the average taxonomy alignment scores for approach A and the average control-flow complexity scores for approach B depending on the percentage of randomly assigned activities in the processes (data from Exp. 1). 2.2 %[3] randomly assigned activities results in a still very good taxonomy alignment score of 0.77 for approach A and in a very low control-flow complexity score of 0.15 for approach B. 6.5 % random activities results in a still reasonable score of 0.55 for A and 0.35 for B. When 10 % of the activities are randomly assigned, the metrics score is 0.47 for both approaches. To conclude, both approaches still provide good scores (>0.5 for taxonomy alignment and <0.5 for control-flow complexity) when less than approx. 8 % activities in the input processes are not aligned with the taxonomy. We suspect that this class covers a wide range of real-world processes since it is very likely that activities that belong semantically together are also structurally related in the input processes. However, the assumption of acceptable scores (>0.5 / <0.5) requires further empirical validation with experts or practitioners.

[3] 2.2 % corresponds to 2 swaps resulting in 4 misaligned activities out of 183 in Fig. 6.

We have created variants of the taxonomy with deeper and flatter hierarchies. Our experiments show, that all findings also apply for these variants. Only the classes of acceptable quality are influenced by the taxonomy depth.

6 Related Work

Translations of activity-centric processes to declarative GSM models have been studied [7,14,17,18]. The approach in [7] generates from UML activity diagrams with data objects and state information as input state machines for data objects, and then translates the state machines into flat GSM models. The translation of Petri nets to GSM was addressed in [17] and applied to mining GSM processes in [18]. The approach is based on calculating pre-condition sets for each activity in order to generate guards of atomic stages. The resulting GSM models are completely flat. To the best of our knowledge, the syntactic translation approach in [14] is the only approach that generates nested GSM models, with nesting based on the block structure of the input process.

A key component of our work is the mapping between input processes and taxonomies. Such a mapping could be obtained by matching pre- and and post-conditions of activities [14]. As an alternative approach to obtain a mapping, processes are matched with a taxonomy based on label similarity [20]. However, this would not take into account (explicit) business goals, which is a key idea of abstractions in GSM. Approaches combining activity-centric modeling and goal modeling such as [4,13] lead to richer input models, which potentially allow to derive the taxonomy and mapping.

The broader context of our quality metrics is framed by the work on quality of conceptual models in general [9,16] and quality of business process models [10,21] in particular. Metrics for business process models were inspired by metrics from software engineering [6,11], namely coupling, cohesion, complexity, modularity and size. Coupling and cohesion in the context of business processes were addressed in [19], where the major goal is to find a proper granularity of activities. The control-flow complexity (CFC) of activity-centric models was studied in [3], where the complexity is measured based on different gateway types and the potential number of states.

There are no quality metrics for GSM processes. We have made a first step with our taxonomy alignment metrics following the basic principles of stage nesting in GSM. The metrics is accompanied with a control-flow complexity metrics to assess unwanted communications between active composite stages. However, in contrast to [3,10] our control-flow complexity metrics does no assess the control-flow complexity of the input process. The core of both metrics developed here is counting existing abstractions or unwanted control-flow between composite stages. This naturally satisfies all 9 properties of Weyuker measures [22] for software programs. The normalized metrics are in-line with all relevant properties of Weyuker measures given that their purpose is to compare different translations of the same input process.

7 Conclusions and Future Work

GSM models allow to group stages based on the fulfillment of business goals. The absence of goals in activity-centric models leads to undesirable syntactic translations. We presented two novel metrics for GSM translations assessing the quality of stage nesting relative to domain taxonomies and assessing the control-flow complexity induced by stage nesting. We developed a semantic rewrite algorithm to enhance the taxonomy alignment metrics of syntactic translations. Experiments show that rewritten translations can achieve reasonable to perfect metrics scores if input processes are well aligned with the domain taxonomies. When input processes are poorly aligned a translation can either achieve optimal alignment scores or low complexity scores but not both.

While we argue that adding semantics nestings to GSM translations will also enhance understandability of the models, this hypothesis still needs to be addressed in further evaluations with end-users. Such a study could also reveal details on the interpretation of the metrics values: What values can be considered good? At which scores do processes actually get uncomprehensible?

Other fields of future work include the study of translations of process models that already partially include (typically structural) groupings (e.g. BPMN sub-processes). Given such models, the taxonomy and mapping creation process may exploit existing groupings. The translation itself can be realized as presented in this paper. Finally, we suppose that processes, where the translations have high control-flow complexity might already have deficits in their activity-centric representation. On the one hand they might themselves by hardly understandable, on the other hand the control-flow may still have room for optimizations. Both questions are interesting future work.

References

1. Bille, P.: A survey on tree edit distance and related problems. Theor. Comput. Sci. **337**(13), 217–239 (2005)
2. Bodenreider, O.: Biomedical ontologies in action: role in knowledge management, data integration and decision support. Yearb. Med. Inf. 67–79 (2008)
3. Cardoso, J.: Control-flow complexity measurement of processes, Weyuker's properties. Int. J. Math. Comput. Phys. Electr. Comp Eng. **1**(8), 366–371 (2007)
4. Cortes-Cornax, M., Matei, A., Dupuy-Chessa, S., et al.: Using intentional fragments to bridge the gap between organizational and intentional levels. Inf. Softw. Tech. **58**, 1–19 (2015)
5. Damaggio, E., Hull, R., Vaculín, R.: On the equivalence of incremental and fixpoint semantics for business artifacts with guard-stage-milestone lifecycles. Inform. Syst. **38**(4), 561–584 (2013)
6. Dhama, H.: Quantitative models of cohesion, coupling in software. J. Syst. Soft. **29**(1), 65–74 (1995). Oregon Metric Workshop
7. Eshuis, R., Van Gorp, P.: Synthesizing data-centric models from business process models. Computing **98**, 345–373 (2015)
8. City Office for Property Management of Hangzhou: 2014 rental subsidies for low income families: processing guidelines, July 2014 (in Chinese)

9. Gemino, A., Wand, Y.: A framework for empirical evaluation of conceptual modeling techniques. Requirements Eng. **9**(4), 248–260 (2004)
10. Gruhn, V., Laue, R.: Complexity metrics for business process models. In: International Conference on Business Information Systems - BIS, vol. 85, pp. 1–12 (2006)
11. Henry, S., Kafura, D.: Software structure metrics based on information flow. IEEE Trans. Softw. Eng. **SE–7**(5), 510–518 (1981)
12. Hull, R., Damaggio, E., De Masellis, R., et al.: Business artifacts with guard-stage-milestone lifecycles: managing artifact interactions with conditions and events. In: Proceedings of DEBS, pp. 51–62. ACM (2011)
13. Koliadis, G., Ghose, A.K.: Relating business process models to goal-oriented requirements models in KAOS. In: Hoffmann, A., Kang, B.-H., Richards, D., Tsumoto, S. (eds.) PKAW 2006. LNCS (LNAI), vol. 4303, pp. 25–39. Springer, Heidelberg (2006)
14. Köpke, J., Su, J.: Towards ontology guided translation of activity-centric processes to GSM. In: Reichert, M., Reijers, H. (eds.) BPM Workshops 2015. LNBIP, vol. 256, pp. 364–375. Springer, Heidelberg (2016). doi:10.1007/978-3-319-42887-1_30
15. McCarthy, W.E.: The REA accounting model: a generalized framework for accounting systems in a shared data environment. Acc. Rev. **57**(3), 554–578 (1982)
16. Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data Knowl. Eng. **55**(3), 243–276 (2005)
17. Popova, V., Dumas, M.: From petri nets to guard-stage-milestone models. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 340–351. Springer, Heidelberg (2013)
18. Popova, V., Fahland, D., Dumas, M.: Artifact lifecycle discovery. Int. J. Coop. Inf. Syst. **24**, 44 (2015). http://dx.doi.org/10.1142/S021884301550001X. 1550001
19. Reijers, H.A., Vanderfeesten, I.T.P.: Cohesion and coupling metrics for workflow process design. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 290–305. Springer, Heidelberg (2004)
20. Smirnov, S., Dijkman, R., Mendling, J., Weske, M.: Meronymy-based aggregation of activities in business process models. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 1–14. Springer, Heidelberg (2010)
21. Vanderfeesten, I., Cardoso, J., Mendling, J., Reijers, H., van der Aalst, W.M.P.: BPM and workflow handbook, chapter quality metrics for business process models, p. 179 (2007)
22. Weyuker, E.J.: Evaluating software complexity measures. IEEE Trans. Softw. Eng. **14**(9), 1357–1365 (1988)
23. Zugal, S., Pinggera, J., Weber, B., Mendling, J., Reijers, H.A.: Assessing the impact of hierarchy on model understandability – a cognitive perspective. In: Kienzle, J. (ed.) MODELS 2011. LNCS, vol. 7167, pp. 123–133. Springer, Heidelberg (2012)
24. Zugal, S., Soffer, P., Haisjackl, C., et al.: Investigating expressiveness and understandability of hierarchy in declarative business process models. Softw. Sys. Model. **14**(3), 1081–1103 (2013)