Towards Vertical Alignment of Process Models - A Collection of Mismatches

Matthias Weidlich[1], Gero Decker[1], Mathias Weske[1], and Alistair Barros[2]

[1] Hasso Plattner Institute, Potsdam, Germany
{matthias.weidlich,gero.decker,mathias.weske}@hpi.uni-potsdam.de
[2] SAP Research, CEC Brisbane, Australia
alistair.barros@sap.com

Abstract. Process models are created to answer a specific purpose. Multiple models created in the context of the same scenario, but for different purposes, differ with respect to their level of abstraction. However, it is a common observation that differences between these models related not only to the applied abstraction level. Aspects that are of no relevance regarding the purpose of the model are neglected, whereas other aspects that are relevant to answer the purpose are added. These *mismatches* violate existing consistency notions between models of different abstraction levels, as these notions focus on strict refinement relations between models. We argue that these models are not *wrong*. On the contrary, the mismatches are a natural result of the different focus of these models. In this paper, we present a categorisation and informal description of these mismatches.

1 Introduction

The wide variety of drivers for business process management leads to a set of different process models describing a certain scenario. Depending on the concrete purpose of the model, there is a huge difference in the appropriate level of abstraction of processes for all involved stakeholders. On a very high level solely the major activities of an enterprise are modelled. Here, an intuitive overview of the major processing steps and their performance indicators (KPIs) is in the centre of interest. These high level models are often independent of any concrete organisational or technical environment. Therefore, they might also be applied in order to define the scope of a process-related project.

At the other end of the line, processes are specified in a fine-grained manner. They might aim at capturing technical requirements. Thus, aspects such as the treatment of exceptional cases or data formats are specified. Furthermore, low-level models often also focus on the relation between the process and its execution environment. Organisational units that are mandated to execute the tasks and information systems that support their execution are assigned to certain parts of the process. Strengthened process awareness and effective change management in an organisation are the driving factors for the application of these types of process models.

Evidently, common scenarios require multiple levels of abstraction, as real-world processes are too complex to be captured by a single high-level and a single low-level model. Driven by a specific objective, an *appropriate* process model is created, which incorporates a reasonable level of detail, focus on certain properties, and neglects unrelated aspects. Albeit complicated by the usage of different modelling approaches, vertical alignment of process models is a generic problem, independent of any language or technique. As stated before, a process model, which is appropriate in a particular context, highlights certain aspects, whereas others are neglected or even ignored. Consequently, the assumption that these process models can always be derived through hierarchical refinement appears to be unrealistic. A variety of mismatches between these models is, indeed, a more realistic scenario. From our point of view, these mismatches are in the nature of process models emphasising dedicated aspects. Thus, avoidance of mismatches might not only be impossible in certain scenarios, it might also be undesired. That is to say that a resolution of these mismatches might impact the adequacy of a process model in a negative manner.

This paper presents an informal description of common mismatches as a first step to overcome the discussed disconnect of process models. The remainder of this paper is structured as follows. Section 2 introduces a classification framework for mismatches between process models of different levels of abstraction. The broad field of related work is discussed in section 3. Finally, we conclude in section 4.

2 Mismatches between process models of different levels of abstraction

After the previous section introduced various mismatches by means of an exemplary process, this section gives a general overview of existing mismatches between process models of different levels of abstraction. We group the mismatches according to the perspective that is affected. Thus, mismatches in the context of the process, activity, control flow, data, and resource perspective are discussed separately.

2.1 Process Perspective

The most coarse-grained mismatches cannot be traced back to the basic building blocks of a process model, namely activities, data elements, and resources, but related to the notion of processes in general. We identified three major mismatches for the process perspective, which are listed in table 2.1.

The first mismatches (P.1) describes the fact a process might be split up into several processes in another model. Thus, the *process coverage* between two sets of processes of different process models is the same, whereas the *process slicing* is different. A set of processes in one model is represented by a set of processes of the other model, which is also illustrated in figure 1. Here, the process in the upper part corresponds to the set of processes containing the two processes on

Table 1. Mismatches in the process perspective

	ID	Mismatch
Processes	P.1	Process Fragmentation There is no relationship between single processes, but *sets* of processes of two models.
	P.2	Process Case Relation A case for process in one model does not represent a case for the corresponding process in the other model.
	P.3	Fragmented Process Case Relation Processes are fragmented differently. In addition, the relation between a process and a case is different for at least one of the process fragments.

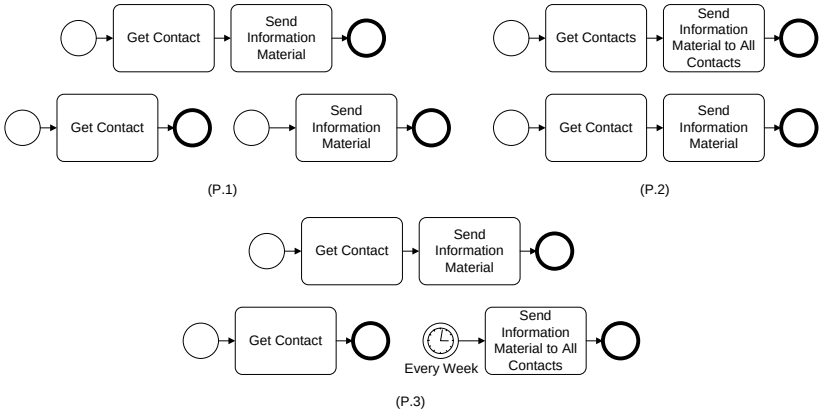

Fig. 1. Examples for the mismatches related to the process perspective

the lower part. Please note that as a consequence of this mismatch, all other mismatches defined on the *process* level might also occur on the level of *sets* of processes.

It is worth mentioning that mismatch P.1 requires full correspondence, while we do not introduce a mismatch for partial correspondence processes. In case two sets of processes correspond to each other solely to a partial extent, it is sufficient to capture this mismatch on the level of activities, data elements, or resources, respectively. However, we capture a special case of partial equivalence. That is, a process in one model corresponds to a process in another model in the sense that the difference in the process coverage can completely be traced back to the number of executions (P.2). In other words, one instance of a process corresponds to multiple instances of a dedicated process in the other model. Figure 1 shows an example for this case. The upper process handles a set of contacts, while the lower process handles only a single contact. Therefore, the notion of a case is different for both processes.

The third mismatch (P.3) can be derived from the previous ones. Process are fragmented differently in both models, while there is also a difference in the notion of a case for at least one of the process fragments. Again, an example can be found in figure 1. The upper process describes the handling of a single contact. The same holds true for the lower left process. In contrast, the lower right process corresponds to multiple instances of the upper process as multiple contacts are treated in the same instance.

Against the background of model alignment, we argue that it is also reasonable to neglect any non-correspondence between processes. Therefore, we avoid to introduce a dedicated mismatch that captures sets of processes of one model that have not even a partial counterpart in the other model.

2.2 Activity Perspective

Activities are the elemental processing steps of every business process. Not surprisingly, various mismatches between process models of different abstraction levels are related to these elemental processing steps. We list five mismatches along with a short explanation in table 2.2.

The first mismatch (A.1) resembles the one defined also on the process level. It describes the case that the *activity coverage* between two sets of activities of different process models is the same, whereas the *activity slicing* is different. Thus, a set of activities in one model is represented by a set of activities of the other model, whereas the contained activities or subsets of activities are without a direct counterpart. An example is shown in figure 2. Assuming that the sequence of the upper two activities describes exactly the same amount of work or computation as the sequence of the lower three activities, this example shows different activity fragmentation. As in case of the process mismatch, all other mismatches defined on the *activity* level might also occur on the level of *activity sets*.

In contrast to the first mismatch, there is a difference in the activity coverage in mismatch A.2. Thus, the constraint of the first mismatch, i.e. the amount of

Table 2. Mismatches in the activity perspective

	ID	Mismatch
Activities	A.1	Activity Fragmentation There is no relationship between single activities, but *sets* of activities of two models.
	A.2	Partial Activity Equivalence An activity in one model is only *partially covered* by the corresponding activity in the other model.
	A.3	Non-Covered Activity An activity in one model is *not covered* by any activity in the other model.
	A.4	Activity Iteration An activity is executed (in a sequential order) more often in one model than the corresponding activity in the other model.
	A.5	Activity-Case Relation A case for an activity in one model does not represent a case for the corresponding activity in the other model.

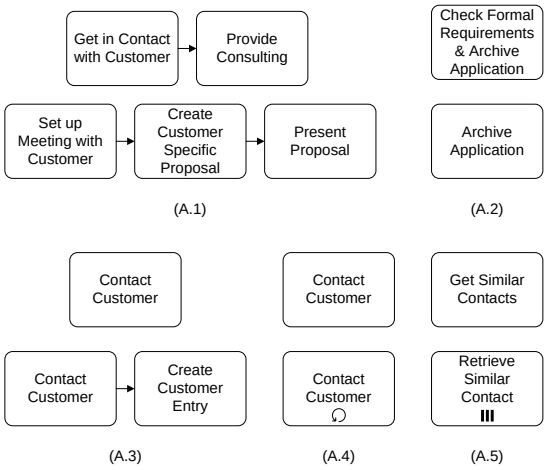

Fig. 2. Examples for the mismatches related to the activity perspective

work or computation described by two corresponding activities is the same, is not fulfilled. An activity in one model (or a set of activities, respectively) is represented only *partially* in the other model. The example in figure 2 shows such a scenario.

The third mismatch (A.3) logically follows the first two ones. An activity in one model is *not* represented in the other model.

The last two mismatches are related in the sense that both are specialisations of mismatch A.2. Both assume an activity in one model that is partially represented by an activity in the other model. However, the difference in the activity coverage can completely be traced back to the number of executions in the scope of a process instance. Therefore, these mismatches are also closely related to mismatches P.2 introduced above.

Mismatch A.4 describes the scenario, in which an activity in one model is executed more often than the related activity in the other model. In particular, one activity is represented partially by the other activity if the execution of one *activity* instance is the basis of the comparison. However, one activity is completely represented by the other activity if the execution of all activity instances is taken into account. Therefore, one activity represents the other activity in case the *process* instance is the basis for the comparison. Further on, mismatch A.4 assumes these activity instances to be executed in a sequential order. An example is illustrated in figure 2. Here, an immediate iteration of activity *Contact Customer* is realised. Nonetheless, this is not required according to our definition of the mismatch. *Contact Customer* might as well be repeated with other activities happening in between. On the contrary, mismatch A.5 describes a non-matching number of related activity instances that are executed concurrently. There is a mismatch with respect to the definition of the case for two related activities. For a single instance of an activity in one model, we create multiple concurrent instances of the related activity in the other model. However, the single instance of the first activity is completely represented by the set of concurrent instances of the second activity.

It was already mentioned that mismatches A.4 and A.5 are closely related to the mismatch P.2 defined between processes. In particular, the existence of mismatch P.2 implies the existence of the mismatches A.4 and A.5 for all activities contained in the respective process.

2.3 Control Flow Perspective

In table 2.3, the first control flow mismatch (C.1) describes different causal dependencies between activities. It is based on the assumption that we already identified corresponding activities (or sets of activities, respectively). However, these corresponding activities are executed in a different order in two models. They might be in sequential order in one model, whereas they are executed concurrently in the other model. Another example would be an interchanged order of two corresponding activities as it is illustrated in figure 3.

Mismatch C.1 has been defined in a very generic way and refers to a variety of differences between process models. In the context of models of different ab-

Table 3. Mismatches in the control flow perspective

	ID	Mismatch
Control Flow	C.1	Different Causal Dependencies Equivalent activities have different execution dependencies. Common examples are sequentialisation, parallelisation, and an interchanged order.
	C.2	Rerouting A sequential execution dependency between two activities in one model is not preserved in the other model due to rerouting of control flow.
	C.3	Alternative Merge A sequential execution dependency between two activities in one model is not preserved in the other model as control flow might merge from an alternative preceding branch in the other model.
	C.4	Decision Distribution A decision point in one model is represented by multiple decision points in the other model.

straction levels, several specialisations of this mismatch are of particular interest. Mismatch C.2 refers to rerouting of the control flow. That is, in one model there are two activities that have a sequential execution dependency in sense that one of them is executed whenever the other is executed. In the second model, there are corresponding activities for both activities. Nevertheless, the sequential execution dependency is not preserved. Thus, the control flow might be rerouted after execution of the first activity. An example can be found in figure 3. Further on, C.3 represents the counterpart for C.2. Again, a sequential execution dependency between corresponding activities is not preserved. The control flow might merge from an alternative branch, so that the execution of the second activity might happen without an execution of the first one.

Further on, mismatch C.4 describes scenarios, in which a decision point in one model is represented by different decision points in the other model. Thus, a single decision is fragmented into multiple decisions. It might also be the case that not only one but multiple coarse-grained decisions are split up into multiple fine-grained decisions. Instead of an one-to-many relationship, this would result in an many-to-many relationship between the decision points. However, against the background of processes on different abstraction levels, we consider these cases to be of only theoretical interest. Therefore, we focus on the one-to-many relationship as it is shown in the example in figure 3. It is worth mentioning that mismatch C.4 does not necessarily involve a mismatch with respect to activity coverage as it is shown in the example.

2.4 Data Perspective

The first mismatches with respect to the data perspective resemble some of the mismatches defined on the activity layer. As shown in table 2.4 there are

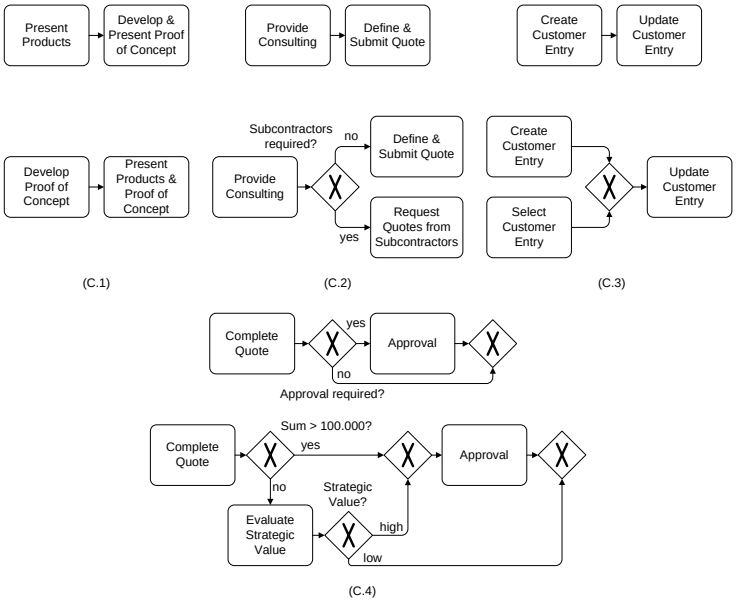

Fig. 3. Examples for the mismatches related to the control flow perspective

three different types of relations between data elements or sets of data elements, respectively. Firstly, data elements can be fragment differently in two models (D.1). Secondly, a data element of one model might only be partially covered by the other model (D.2), whereas a counterpart might also be completely missing (D.3).

Similar to the activity perspective, mismatches might also result from a differing notion of an instance of related data elements. Mismatch D.4 describes the case that a data element in one model corresponds to multiple instances of a data element in the other model. Therefore, it is a specialisation of mismatch D.2 as there must always be a partial coverage between these two data elements. The modelling of lists of certain entities instead of the entity itself is a common example for mismatch D.4. It is illustrated in figure 4. Again, the mismatch on the instance level is closely related to the process mismatch P.2, as the existence of mismatch P.2 implies the existence of the mismatch D.4 for all data objects contained in the respective process.

Based on a identified relations between data elements and activities, the last mismatch D.5 describes a difference in the way data is accessed in two models. In the course of abstracting or refining a process model, data access might be restricted or expanded. Figure 4 illustrates a simple example for this mismatch.

Table 4. Mismatches in the data perspective

	ID	Mismatch
Data Handling	D.1	Data Element Fragmentation There is no relationship between single data elements, but *sets* of data elements of two models.
	D.2	Partial Data Element Equivalence An data element in one model is only *partially covered* by the corresponding data element in the other model.
	D.3	Non-Covered Data Elements A data element in one model is *not covered* by any data element in the other model.
	D.4	Different Amount of Data Instances A data element in one model represents multiple instances of a data element in the other model.
	D.5	Different Data Access There is a difference in the data access between corresponding pairs of data elements and activities of two models.

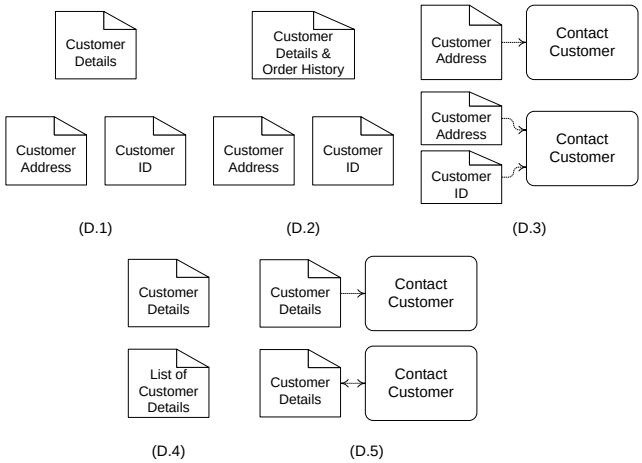

Fig. 4. Examples for the mismatches related to the data perspective

2.5 Resource Perspective

As introduced for the other perspectives, three mismatches related to relations between resources of different process models. As listed in table 2.5 they might be fragment in a different way (R.1) or they only be partially related (R.2). In addition, a resource in one model might not be represented in the other model (R.3). Again, examples for these mismatches are illustrated in figure 5.

Table 5. Mismatches in the resource perspective

	ID	Mismatch
Resources	R.1	Resources Fragmentation There is no relationship between single resources, but *sets* of resources of two models.
	R.2	Partial Resources Equivalence A resource in one model is only *partially covered* by the corresponding resource in the other model.
	R.3	Non-Covered Resources A resource in one model is *not covered* by any resource in the other model.
	R.4	Contradicting Resource Assignments An activity or data element is assigned to a resource in one model, but not to the corresponding resource in the other model.
	R.5	Additional Resource Assignments An activity or data element is assigned to multiple resources in different models and the resources are not related.

Beyond mismatches related to the relation of resources of different process models, mismatch R.4 and R.5 describe differences in resource assignments. Obviously these mismatches are based on identified relations between different models regarding their resources and activities or data elements. We distinguish non-matching resource assignments that are contradicting (R.4) from those that are compliant (R.5). In the first case an activity or data element is assigned to a resource in one model, but not to the corresponding resources in the other model. In the second case, an activity or data element is also assigned to two different resources. Nonetheless, these two resources are independent of each other. As illustrated in figure 5, this mismatch often results from different perspectives that are adopted by the process models. It is worth mentioning that mismatch R.5 covers scenarios, in which the resource assignment is specified solely in one of the investigated process models.

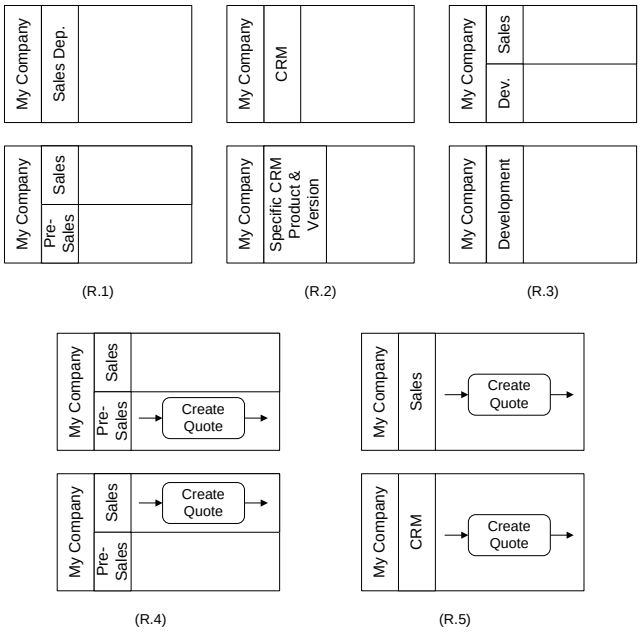

Fig. 5. Examples for the mismatches related to the resource perspective

3 Related Work

Our work relates to a variety of research fields from which we pick up certain concepts, i.e. process integration, process change management, process similarity, and process abstraction mechanisms.

In this paper we primarily raise the awareness for the fact that process models on different levels of abstraction focus on different aspects. For instance, a model might focus on the business conducted, whereas a another more fine-grained model specifies the available IT functionality. The gap between these models has been described in various publications [1–4]. One of the paper's authors proposed a process support layer realising common mismatch patterns to bridge this gap [5]. Although it seems questionable, whether the idea to wrap the technical models in order to enact the business models is feasible in real world scenarios, most of the pattern identified in [5] are subsumed by the mismatches presented in this work. Addressing the gap between business models and technical models, Henkel et al. [3] introduced *realisation types* that are applied to transform a business model into a technical model in order to overcome the disconnect. Further on, the realisation types are grouped according the process aspects, i.e. the functional, behavioural, informational, organisational, and transactional perspective. Albeit more abstract, these realisation types influenced the definition of our mismatches. Our work is also influenced by the business-driven development (BDD) paradigm as introduced by Koehler et al. [4]. The BDD

methodology postulates the necessity to distinguish between a business-centred *analysis model* and an it-centred *design model* and aims at seamless transition from the first to the latter.

Building on top of the multi-viewpoint paradigm, the authors of [6] show how enterprise and computational views can be related. However, their work is based on a strict notion of composition that cannot be applied in our scenarios. Broadening the focus to a generic multi-viewpoint approach, a framework in order to handle inconsistencies was also presented already in 1994 [7]. In this work, the authors show how to apply temporal logic in order to detect and resolve inconsistencies in multi-viewpoint specifications.

Differences between process models are also in the centre of interest of process integration. In that field it is assumed that process models originate from different sources. Therefore, they are different yet similar. Recently, Dijkman published a classification of mismatches between similar processes [8] and showed how they can be detected [9]. These *mismatch patterns* are similar to the mismatches we presented in this paper. Several of our mismatches have a counterpart in the framework presented in [8]. As mentionend before, Dijkman focus on mismatches of very similar processes that differ only slightly. A process that is enacted in different departments of the same organisation is a typical example for such a scenario. Typically, these processes are on the same level of abstraction. Therefore, these mismatch patterns are more fine-grained than our mismatches. Common problems arising with processes on different abstraction levels (e.g. the distribution of a decision point) are out of scope for the mismatch patterns, while they are covered by our framework.

Research was also conducted on process integration in the field of statecharts [10], composite business processes [11], and EPCs representing different views [12]. However, these approaches assume a high similarity between the processes and concentrate on the definition of a merge operation. A technique to merge an existing process model with a related reference model based on model correspondences has also been presented in the context of business-driven development [13]. This approach concentrates on n-ary activity and subprocess correspondences and does not tackle mismatches related to the control flow structure.

Resolution of small differences that exist between process models is also a key aspect of process change management. In this context, differences are between multiple *versions* of a process model. Evolution of a process model is realised through elementary change operations. These operations can be aggregated to more abstract *change patterns* as proposed by the authors of [14]. Thus, change patterns also classify common mismatches between process models. In case of the absence of a change log, process differences can be detected and resolved based on existing correspondences between process model versions [15]. However, in contrast to our work, process change management assumes that differences are a result of a transformation of a process based on clearly defined change operations.

Existing work in the field of process similarity is also related to our work. Although approaches targeting the verification of processes, such as trace equivalence [16] and bisimulation [17], are restricted to Boolean answers with respect

to the similarity of processes, other approaches provide more nuanced notions. The authors of [18] present a method to measure similarity between process models based on the enforced execution constraints. On the other hand, a similarity measure might also be grounded on the aforementioned change operations as it was shown in [19]. These approaches relate to our work, as they might inspire a consistency notion that is able to cope with our mismatches.

Shifting the focus from the differences between related process models to issues that result from abstractions, inspiring work has been published in the field of behaviour inheritance. Here, the main idea is the adaptation of inheritance notions known for structural aspects in the context of object-oriented design to behavioural aspects. In [20] Basten et al. present a full-fledged framework based on labelled transition systems and branching bisimulation. The authors introduce different notions of behaviour inheritance, namely *protocol inheritance*, *projection inheritance*, and combinations of these basic notions. In the case of protocol inheritance, a model inherits the behaviour of a parent model, if it shows the same external behaviour when all actions that are not part of the parent model are *blocked*. On the other hand, projection inheritance is fulfilled, if a model shows the same external behaviour as the parent model, if all actions that are not part of the parent model are *hidden*. In [21] the authors show the application of these notions for UML activity diagrams, sequence diagrams, and statechart diagrams.

Similar ideas have been presented in [22, 23]. Here, the authors distinguish *invocation consistency* and *observation consistency* depending on whether a model inherits the interface of a parent model or its externally observable behaviour. In [22] both notions are defined based on labelled transition systems. Based thereon, the application of these notions in the context of UML statechart diagrams is shown in [23]. As discussed in [20], invocation consistency and observation consistency directly correspond to the aforementioned notions, i.e. protocol inheritance and projection inheritance.

Concentrating on the actions that are invocable, Stumptner et al. introduce *weak invocation consistency* and *strong invocation consistency* in [24]. While weak invocation consistency corresponds to the invocation consistency and protocol inheritance, strong invocation consistency enforced additional constraints. That is, it must be possible to use a model in the same way as its parent, even if added actions (not inherited from the parent) are used. Both notions are defined for object/behaviour diagrams (OBD), while the authors also show how UML statechart diagrams can be mapped to OBDs.

Although the ideas on behavioural inheritance are highly related to our work, none of the introduced notions can be applied in our scenarios. In general, the existing notions are too restrictive for a scenario as introduced in section **??**. All notions support only a limited variety of mismatches. The authors of the most liberal notion, namely *life-cycle inheritance*, list a set of *inheritance preserving transformation rules* in [20]. The insertion of activities between existing ones or the addition of loops containing new actions are examples for these rules. However, everything that goes beyond these rules, for instance differences in the

process instantiation mechanism, does not preserve consistency. We can summarise that all of these consistency notions assume that behaviour is *added* in a *structural way* (e.g. iteration, choice, sequential or parallel composition) in the course of refinement of process models. This assumption does not hold for scenarios that show our mismatches.

Several of our identified mismatches are also addressed in the context of abstraction mechanisms for process models. Given a complex and full-fledged model, abstraction approaches apply reduction algorithms to generate a simplified model. Such an abstraction might be driven by the aim to limit reasoning effort or to extract the main process logic for manual analysis. Grounded on research conducted on generic graph transformation rules, various abstraction approaches have been presented. Zerguini [25] defines elemental Petri net transformations that realise an aggregation of *reducible subflows* and preserves the soundness criterion. Other approaches focus on the generation of abstract views of process models. The authors of [26] identify *single-entry-single-exit* regions and apply graph aggregation and graph reduction techniques to these regions. In [27] it is shown, how aggregation of process models can be extended with *customisation*, i.e. a user selects process parts that should not be aggregated. Another aggregation algorithm for processes is introduced in [28]. Ensuring preservation of dedicated properties of the process model, the authors of [29] and [30] introduce aggregation patterns and discuss their influence on these properties. Although the use-case of process abstraction approaches is different compared to ours – we want to align multiple *existing* models on different abstraction levels instead of generating them – the patterns applied in these approaches also inspired the definition of some of our mismatches.

4 Conclusion

In this paper we elaborated on the disconnect between process models of different abstraction levels. From our point of view, this disconnect is in the nature of models that are created for different purposes. Nevertheless, there is a pressing demand for better alignment of these models.

As a first step towards a framework that allows for vertical alignment of process models, we presented an informal description and classification of common mismatches between these models. As discussed above, these mismatches go well beyond the differences between process models that are typically handled in the field of process integration.

References

1. Grover, V., Fiedler, K., Teng, J.: Exploring the Success of Information Technology Enabled Businessprocess Reengineering. IEEE Transactions on Engineering Management **41**(3) (August 1994) 276–284
2. Rolland, C., Prakash, N.: Bridging the Gap Between Organisational Needs and ERP Functionality. Requirements Engineering **5**(3) (October 2000) 180–193

3. Henkel, M., Zdravkovic, J., Johannesson, P.: Service-based processes: Design for business and technology. In Aiello, M., Aoyama, M., Curbera, F., Papazoglou, M.P., eds.: ICSOC, ACM (2004) 21–29
4. Koehler, J., Hauser, R., Küster, J.M., Ryndina, K., Vanhatalo, J., Wahler, M.: The Role of Visual Modeling and Model Transformations in Business-driven Development. Electr. Notes Theor. Comput. Sci. **211** (2008) 5–15
5. Decker, G.: Bridging the Gap between Business Processes and existing IT Functionality. In: Proceedings of the 1st International Workshop on Design of Service-Oriented Applications (WDSOA), ICSOC, Amsterdam, The Netherlands (December 2005) 17–24
6. Dijkman, R.M., Quartel, D.A.C., Pires, L.F., van Sinderen, M.: A Rigorous Approach to Relate Enterprise and Computational Viewpoints. In: EDOC, IEEE Computer Society (2004) 187–200
7. Finkelstein, A., Gabbay, D.M., Hunter, A., Kramer, J., Nuseibeh, B.: Inconsistency Handling in Multperspective Specifications. IEEE Transactions on Software Engineering **20**(8) (1994) 569–578
8. Dijkman, R.M.: A Classification of Differences between Similar Business Processes. In: EDOC, IEEE Computer Society (2007) 37–50
9. Dijkman, R.M.: Feedback on Differences between Business Processes. Working Paper WP-234, Eindhoven University of Technology, Eindhoven, The Netherlands (2007)
10. Frank, H., Eder, J.: Towards an Automatic Integration of Statecharts. In Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E., eds.: ER. Volume 1728 of Lecture Notes in Computer Science., Springer (1999) 430–444
11. Grossmann, G., Ren, Y., Schrefl, M., Stumptner, M.: Behavior Based Integration of Composite Business Processes. In van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F., eds.: Business Process Management. Volume 3649. (2005) 186–204
12. Mendling, J., Simon, C.: Business Process Design by View Integration. [31] 55–64
13. Küster, J.M., Koehler, J., Ryndina, K.: Improving Business Process Models with Reference Models in Business-Driven Development. [31] 35–44
14. Weber, B., Rinderle, S., Reichert, M.: Change Patterns and Change Support Features in Process-Aware Information Systems. In Krogstie, J., Opdahl, A.L., Sindre, G., eds.: CAiSE. Volume 4495 of Lecture Notes in Computer Science., Springer (2007) 574–588
15. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and Resolving Process Model Differences in the Absence of a Change Log. In: to appear. (2008)
16. Hidders, J., Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M., Verelst, J.: When are two Workflows the Same? In Atkinson, M.D., Dehne, F.K.H.A., eds.: CATS. Volume 41 of CRPIT., Australian Computer Society (2005) 3–11
17. van Glabbeek, R.J., Weijland, W.P.: Branching Time and Abstraction in Bisimulation Semantics. Journal of the ACM (JACM) **43**(3) (1996) 555–600
18. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring Similarity between Business Process Models. In Bellahsene, Z., Léonard, M., eds.: CAiSE. Volume 5074 of Lecture Notes in Computer Science., Springer (2008) 450–464
19. Li, C., Reichert, M., Wombacher, A.: On Measuring Process Model Similarity based on High-level Change Operations. In: Proceedings of the 27th International Conference on Conceptual Modeling (ER). (2008)
20. Basten, T., van der Aalst, W.M.P.: Inheritance of Behavior. Journal of Logic and Algebraic Programming (JLAP) **47**(2) (2001) 47–145

21. van der Aalst, W.M.: Inheritance of Dynamic Behaviour in UML. In Moldt, D., ed.: Proceedings of the Second International Workshop on Modelling of Objects, Components, and Agents (MOCA). Number PB-561 in DAIM, Aarhus, Denmark (August 2002) 105–120
22. Ebert, J., Engels, G.: Observable or Invocable Behaviour - You Have to Choose. Technical Report 94-38, Department of Computer Science, Leiden University (December 1994)
23. Engels, G., Heckel, R., Küster, J.M.: Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model. In Gogolla, M., Kobryn, C., eds.: UML. Volume 2185 of Lecture Notes in Computer Science., Springer (2001) 272–286
24. Stumptner, M., Schrefl, M.: Behavior consistent inheritance in uml. In: ER. (2000) 527–542
25. Zerguini, L.: A Novel Hierarchical Method For Decomposition And Design Of Workflow Models. Journal of Integrated Design & Process Science **8**(2) (2004) 65–74
26. Bobrik, R., Reichert, M., Bauer, T.: View-Based Process Visualization. In Alonso, G., Dadam, P., Rosemann, M., eds.: BPM. Volume 4714 of Lecture Notes in Computer Science., Springer (2007) 88–95
27. Eshuis, R., Grefen, P.W.P.J.: Constructing Customized Process Views. Data & Knowledge Engineering **64**(2) (2008) 419–438
28. Liu, D.R., Shen, M.: Workflow Modeling for Virtual Processes: an Order-Preserving Process-View Approach. Information Systems **28**(6) (2003) 505–532
29. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of Service for Workflows and Web Service Processes. Journal of Web Semantics **1**(3) (2004) 281–308
30. Polyvyanyy, A., Smirnov, S., Weske, M.: Business Process Model Abstraction (2008) to appear.
31. Eder, J., Dustdar, S., eds.: Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006, Proceedings. In Eder, J., Dustdar, S., eds.: Business Process Management Workshops. Volume 4103 of Lecture Notes in Computer Science., Springer (2006)