Supporting Business Process Improvement through Business Process Weakness Pattern Collections

Patrick Delfmann, Steffen Höhenberger

European Research Center for Information Systems (ERCIS),
University of Münster, Germany
{patrick.delfmann|steffen.hoehenberger}@ercis.uni-muenster.de

Abstract. A main task of Business Process Management is to identify weaknesses in business processes that may result in monetary or quality-related drawbacks in order to eliminate them subsequently. A common way to identify weaknesses is to analyze process models. While this has traditionally been done manually, recent work proposes using automatic model query approaches promising time and money savings. Query approaches take (weakness) patterns as input and return all process model subsections that match these patterns, hence may be subject to weaknesses. Although numerous model query approaches have been developed, collections of weaknesses do virtually not exist. To exploit the benefits of model querying, a weakness collection would be highly desirable. In this paper, we provide a first version of a weakness pattern collection, which we identified in an empirical study on several hundreds of weakness-afflicted process models and assess its usefulness through applying it to another process model collection.

Keywords: Business Process Improvement, Business Process Modelling, Model Querying, Process Patterns, Business Process Weakness Detection.

1 Introduction

Business Process Management (BPM) includes identifying *business process weaknesses*. Business process weaknesses are parts of a business process that imply inefficiencies, lack of quality, or legal violations. Examples for business process weaknesses are, for instance, redundant work, automation potential, frequently changing responsibilities, or frequent changes of automatic and manual processing. Identifying such weak parts of business processes supports *improving business processes* according to monetary, quality-related or legal aspects, as once such weaknesses are identified, deciders can take action to eliminate them, for instance, through reorganization.

To deal with the complexity of analyzing business processes oftentimes including several hundreds of interrelated tasks, people, data, products, and application systems [1-2], the use of *business process models* has become a matter of course in corporate reality. In business process models, business process weaknesses exhibit *typical structures and label semantics*. As a consequence, many business process weaknesses can be revealed through analyzing a business process' structure and label semantics.

12th International Conference on Wirtschaftsinformatik,
March 4-6 2015, Osnabrück, Germany

Höhenberger, S.; Delfmann, P. (2015): Supporting Business Process Improvement through Business Process Weakness Pattern Collections, in: Thomas. O.; Teuteberg, F. (Hrsg.): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015), Osnabrück, S. 378-392

However, as with the advancement of BPM, many companies have started to develop and maintain large collections of process models, manual process model analysis becomes more and more cumbersome. Recent studies indicate that process model collections may contain hundreds or thousands of models, each of which may in turn consist of hundreds or even thousands of elements [1-2]. For instance, the process model collection of *Suncorp* contains more than 6,000 models [3]. Other examples include the *BIT Process Library* (approx. 750 models) [4] or the process collection of Dutch municipalities (approx. 500 models) [5].

As the goal of business process improvement includes improving monetary benefits of business processes, it is questionable whether analyzing such a huge amount of information contained in process models manually makes sense. In contrast, it may very likely overcompensate the benefits to be gained through process improvement.

Thus, BPM and conceptual modeling scholars have recently proposed to make use of (process) *model query languages* (for a comprehensive overview, cf. [6]) to analyze process models automatically. Roughly speaking, a query consists of a pattern prescribing which structural and semantic properties a subsection of a process model has to fulfill to match the query. Model querying returns subsections of the queried process models matching the given pattern. This means that a query pattern contains model nodes and attributes and defines which of them are required and which are forbidden to be contained in a match, as well as their relationships via (sometimes several) model edges. An example illustrating model querying is depicted in Figure 1.

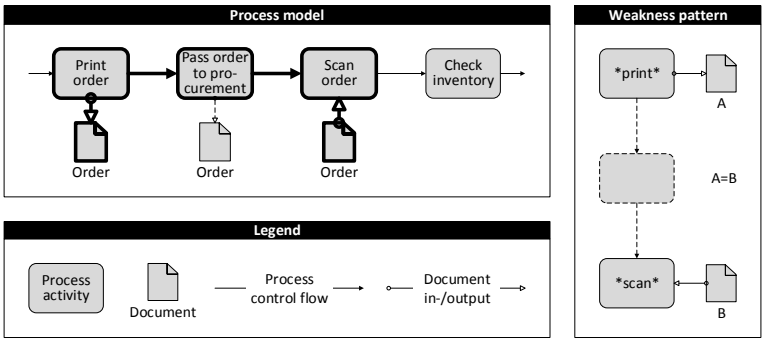

Fig. 1. Querying business process models

The example consists of an excerpt of an order management business process, of which every order that enters the company is printed and passed to the procurement department, where it is scanned for further processing. Obviously, this is a process weakness. Although this weakness seems quite artificial, it is still very common in practice (especially in public administration). The weakness has a typical structure and semantics. In particular, the weakness starts with an activity labelled with a phrase containing the term "print". This activity acts on a document (actually, the document that is printed). Further on in the process model, either directly following the first activity or across a control flow path passing several other activities, another activity acts on the same document and is labelled with a phrase containing the term

"scan". A corresponding weakness pattern hence consists of an activity labelled with the term "print" (plus wildcards) and a process path from this activity to another one labelled with the term "print" (plus wildcards). Furthermore, both activities are connected to a document. The first one produces the document as an output and the second one consumes the document as an input. Both document objects are required to be named identically (*A=B*, cf. weakness pattern in Figure 1). Querying a process model using such a pattern returns those process model sections that match the pattern structurally and semantically (i.e., complying with the specified labels) (cf. process model fraction highlighted bold in Figure 1).

Searching process models for weaknesses this way is promising, as firstly, revealing process weaknesses and subsequently eliminating them may add value to the business process. Secondly, the danger of spending too much effort in searching for weaknesses can be mitigated by considerably decreasing the sheer time for searching. However, in order to make model query approaches really useful for business process weakness detection, they need to be fed with particular weakness patterns. Surprisingly, although model query approaches exist in abundance [6], *collections of particular, commonly occurring, typical weakness patterns do virtually not exist*. In order to exploit the expectable benefits of model query approaches, it would be desirable to establish a collection of common, reusable weakness patterns for business process model querying. Such a collection could be handled similarly to already established collections of software design patterns [7].

Hence, the goal of this paper is to make a step towards establishing a collection of business process weakness patters that can serve as inputs for model query approaches. For this purpose, we conducted an empirical study, in which we examined more than 2000 process models manually that were recorded in several German public administrations. Every weakness found was categorized and formalized using a generic model query language. We could identify seven weakness categories containing in total 111 weaknesses. To assess the common usefulness of the weakness patterns, we applied them to a collection of process models different from the one used for the identification and originating from another business domain. As a technical basis, we used a generic model query approach and a corresponding modeling tool.

The remainder of this paper is structured as follows: In Section 2, we outline related work on business process weakness patterns. As the outcome of our empirical study is a collection of business process weakness patterns, we need an adequate way to represent them, which we present in Section 3. Section 4 reports on the empirical study we conducted to identify typical weakness patterns and gives some examples alongside with their formal representation. In Section 5, we apply the patterns to a process model collection to assess their usefulness. Section 6 provides a conclusion and an outlook towards further research.

2 Related Work

As identifying weaknesses in business processes is one of the main tasks of business process improvement, it has already been referred to by early process management

approaches of the late 1980s [8] and the early 1990s [9-10]. These works already provide so-called best practices providing suggestions how to create business processes with a rather high efficiency, quality, etc.. REIJERS and LIMAN MANSAR concretize these best practices and provide check lists to be reused in reorganization efforts [11]. However, many of these best practices are very general recommendations (e.g., "consider outsourcing a business process in whole or parts of it" [11]). Furthermore, they are not related to particular weaknesses. ZELLNER provides a framework to derive reorganization patterns relating to best practices as mentioned above [12].

To formally specify business process weakness patterns, an abundance of so-called model query languages has been put forth. Such query languages realize the way of searching for patterns as outlined in Section 1. They make use of formal specification techniques, such as Linear Temporal Logic (LTL), Computation Tree Logic (CTL) or graph search techniques. In order not to overload the related work section, we abstain from listing these approaches here. Instead, we refer to a comprehensive survey on query languages contained in [6].

Particular work on weakness patterns is very rare: For instance, BECKER ET AL. present a list of 19 weakness patterns that they apply to process models of the banking sector [13]. WINKELMANN and WEIß apply structural weakness patterns to business processes represented by flow charts [14]. BERGENER ET AL. reuse the patterns of BECKER ET AL. to show how they can be automatically searched for [15].

Surprisingly, we only find very few contributions on business process weakness patterns, although we argue that corresponding collections could be very promising. Hence, we aim at closing this gap with the paper at hand.

3 Representing Weakness Patterns

To represent the weakness patterns we identified in the course of our case study, we make use of a visual model query language that was available from a recent research project [16]. We chose this language due to the following reasons: First, for the sake of comprehensibility and visualization, we decided to use a diagrammatic representation of the patterns (i.e., the patterns look like conceptual models consisting of visual vertices and edges). Only few model query languages exist that support diagrammatic pattern specification (cf. e.g., [6]). Out of these diagrammatic languages, the one we chose exhibits the highest expressive power [16]. Second, as we aimed to apply the identified weakness patterns to a model collection to assess their usefulness, we needed a corresponding implementation in a modelling tool alongside with a matching algorithm and a visualization component that highlights the matches in the models. As we already implemented our own query approach [16], we decided to choose it for this paper (note that we could have used other visual query languages as well – this should not influence our results).

The query language we used allows defining a pattern using a diagrammatic representation. The language's abstract syntax is depicted in Figure 2. Every *pattern* consists of *vertices* and *edges* used to specify the structure of a weakness. Hence, if we would build a triangle of three vertices and three edges, a corresponding model query

would return all subsections of a model that exhibit the structure of a triangle, no matter which model vertex types, edge types or labels occur in the subsection. To further specify the properties of a pattern, each vertex and each edge can be assigned *types* (such as "activity", "event", "entity type", "control flow", etc.) and *captions* (such as "process order", "article", *check*") including wildcards to define which particular vertices and edges of a model are allowed to be mapped to the pattern. Note that *attributes* assigned to process vertices are also regarded as vertices to allow for as much flexibility as possible (e.g., to also allow attributes of attributes). To distinguish attributes from common vertices, we use the *vertex type*.

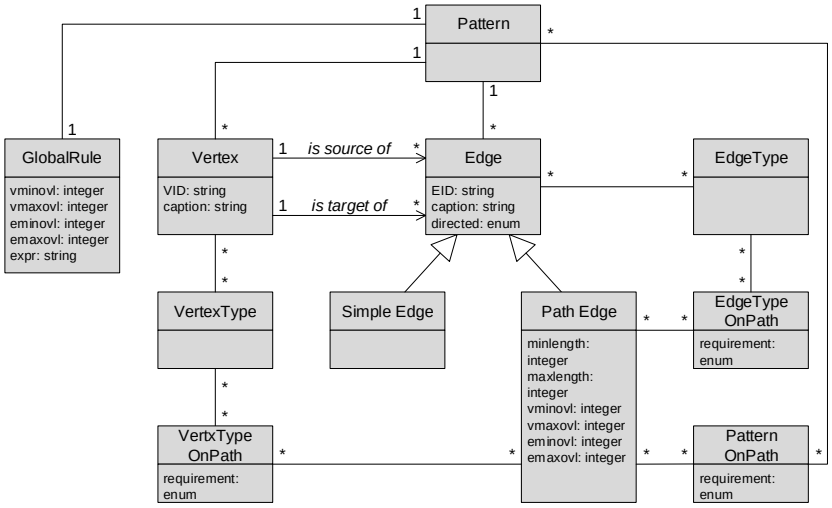

Fig. 2. The abstract syntax of the model query language

For every edge, we can specify its allowed *direction*, that is, if it is directed in the one or the other direction or if it is undirected. Furthermore, combinations are possible. This means that if an edge of the pattern is defined as both undirected and directed, then the matching process would return pattern occurrences in which the pattern edge is mapped to either directed or undirected edges of the model. To search for paths in models (i.e., sequences of several vertices, of which the length is not known before), pattern edges can either be defined as *simple edge* or as *path edge*. While simple pattern edges are always mapped to single edges in a model, path pattern edges are mapped to paths in a model. Several further properties can be defined for path edges: For every path, we can define its *minimum* and *maximum length* and whether or not the path is allowed to cut itself. So, every path has a *minimum* and *maximum* amount of *vertex overlaps* (i.e., the number of places where the path cuts itself via a vertex) and *edge overlaps* (i.e., the number of places where the path cuts itself via an edge). Combining minimum and maximum length, vertex and edge overlaps, we can avoid receiving an infinite amount of pattern matches whenever a model contains loops. The *direction* of edges a path may contain to be mapped is defined similarly to that of simple edges. If a path edge is assigned more than one direction, the edges on the

corresponding path in the model are allowed to have different directions. Every path can additionally be restricted by *requirements*, that is, the edge types, vertex types and other patterns that either (partly) must, are allowed to or are (partly) forbidden to occur on a path. Finally, it is possible to specify pattern-global rules that relate different elements of the pattern by logical expressions. Therefore, every element of a pattern is given an *ID* that can be reused in an expression. For instance, if we wanted to specify a pattern describing a path from a vertex with the ID *X* to another vertex with the ID *Y*, and the vertices' types should be the same, we would define a global rule *X.type=Y.type*. Further global rules define the minimum/maximum cumulated amount of vertex or edge overlaps between all paths of a pattern occurrence.

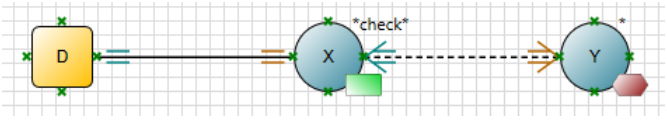

Fig. 3. An example of the concrete syntax of the model query language

The concrete syntax of the query language is quite simple. Pattern vertices are depicted as circles, attributes are depicted as squares, edges as solid lines and paths as dashed lines. As already mentioned, the query language regards attributes as a kind of vertices. However, for a more convenient specification, attributes are depicted other than non-attribute vertices. The information whether or not a vertex is an attribute can be taken from the set of vertex types (see above). The allowed directions are indicated by arrows. If an edge or path should contain undirected edges, this is depicted by two additional parallel short lines adjacent to the vertices. The IDs of the vertices/attributes are placed within the circle/square. If a caption is defined, it is placed at the upper right corner of the vertex. If only one vertex type is allowed for a vertex, its representation (matching the representation of the vertex in a model) is displayed at the lower right corner of the circle (cf. Figure 3). To keep up clarity of patterns, all further specifications are not directly visualized in the pattern, but defined separately. In an according implementation, this means opening a corresponding context menu.

4 Empirical Study

4.1 Identification of Weakness Patterns

To discover typical business process weaknesses, we analyzed a business process model collection from different projects in the area of public administration containing more than 2000 models (due to privacy constraints, we have to keep the details of the projects confidential). The models were provided by a consulting company and had been created with the PICTURE modeling language [17], including not only control flow and activity information but also information about used application systems, responsible organizational units and processed documents/data. The models were as-is models, that is, models describing the current situation prior to any process

improvement efforts. Hence, we expected the models to be suitable for our analysis. The models were analyzed manually by 15 persons with considerable experience in process modelling and improvement. To avoid organizational blindness and loss of creativity, we did not include (few) recommendations from literature into the search that prescribe which types of weaknesses exist. Hence, the identification of possible weaknesses was actually based on experience and common sense (this is how weakness detection is actually done in practice, however, such weaknesses have never been comprehensively documented, categorized and structured). Every process model was analyzed by at least two persons to avoid that potential weaknesses were overlooked.

The weaknesses were documented and categorized. In total, we could identify 280 weaknesses (i.e., distinct weak model sections), which we could generalize as 111 weakness types. (i.e., weaknesses that occurred at multiple places). Furthermore, we defined seven categories to which we assigned weakness types related to similar issues. In the following, we describe the weakness categories we could identify. As we cannot present all 111 weaknesses here, we provide examples for every weakness category and, for some of them, a corresponding weakness pattern.

4.2 Modelling of Weakness Patterns

The patterns we identified should be applied to a model collection in order to assess their usefulness later on. We thus had to specify the weakness patterns according to the modelling language of the model collection. The models of the collection had been built using a domain-specific, terminologically standardized process modeling language called *icebricks* [18]. To terminologically standardize this language, the user has to provide a catalogue of terms that are allowed to be used to denote model elements. To use the standardized terms, every process element of icebricks is assigned to two specific attributes used to express a label: the *business object* and the *business procedure*. Both business object and business procedure consist of terms contained in the catalogue of terms. The combination of both makes a label for a process element (e.g., "check invoice"). Both the models and the catalogues were available to us, so we could use the terms of the catalogue for the specification of the patterns. This way, we could avoid name clashes from the beginning (note that, in absence of a modeling language already including terminological standardization, one would have to employ a corresponding standardization approach, e.g., [19]. Otherwise, automatic weakness detection could result in ambiguous results).

4.3 Weakness Category *Modeling Error*

The category *Modeling Error* contains weakness patterns, which describe situations that do not necessarily arise from an inefficient or wrong execution of the process, but rather from an incorrect way of modeling. This may result from a process modeler's insufficient knowledge of general process modeling or the chosen process modeling language. While weaknesses, which emerged due to the lack of knowledge of general process modeling (for instance overly complicated processes) can be identified more easily as they conflict with general standards of process modeling, language-

dependent weaknesses are more difficult to detect as every modeling language uses a different way of representing certain situations (e.g., the use of interfaces). The latter also applies for the converse: if a weakness is found on the basis of a weakness pattern, it may be no "real" weakness because the language does not allow any other way of modeling this situation. Nevertheless, the weakness patterns in this category are regarded useful to search as they can correct and improve business process models regarding their accuracy, clarity and finally their applicability and analyzability.

The category *Modeling Error* contains 23 weakness patterns, comprising, amongst others, the weaknesses *decision without impact*, *sent document is never received*, and *changing work location without returning*.

The following example shows how we specified the pattern *decision without impact* using the model query approach as outlined in Sections 1 and 3. Decisions without impact occur whenever a process activity describes a decision, but after the activity, no process split occurs, which should normally be the case after a decision.

Figure 4 (a) depicts the graphical representation of the *decision without impact* pattern. It consists of a vertex *A*, which is directly followed by a vertex *B*. In addition, vertex *A* is connected to two attributes called *BO* (business object) and *BP* (business procedure). To express that, in the first vertex, a decision is made, we further specify the contents of the attributes: First, the vertex types of *BO* and *BP* should be the standardized icebricks types *business object* and *business procedure*. Second, *BO* should contain terms that describe a decision. For instance, corresponding with one of the terminological catalogues of icebricks, the related terms were "decide", "approve", "clear", "check", "monitor", or "analyze". Third, we can leave the contents of *BP* open, as we can check, approve, clear, etc., almost any business object. Fourth, the vertex *B* should not be a split object, as the weakness pattern should express that a decision does not have any impact. All these additional specifications are made using the global rules presented in Section 3.

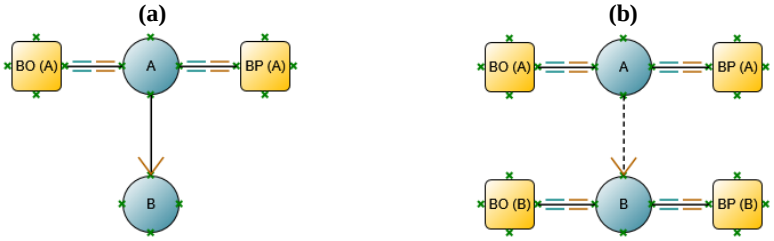

Fig. 4. *Decision without impact (a)* and *check or examination after dispatch or digitalization (b)* patterns

4.4 Weakness Category *Process Flow*

The category *Process Flow* contains weakness patterns regarding an inefficient order or a missing aggregation of activities within processes. The patterns of this category depict situations which contain avoidable waiting time, delayed document transfer or information (transmission) deficits, hence retard the process flow. In combination with the delayed transfer, an information deficit is given, if information is needed but

not available, which may occur due to a delayed transfer or a deficient or inappropriate transmission of information. Additionally, the weakness patterns address an inappropriate handling of business objects, a missing aggregation or decomposition of (redundant) activities as well as inefficient or wrong sequences of activities. An inappropriate handling of business objects is manifold and ranges from unused business objects to wrongly placed activities. Furthermore, redundant or too complex activities decelerate the process and wrong or inefficient orders of activities confuse the process participants. Thus, those activities either have the potential for being aggregated or decomposed or can be re-sequenced for a faster process flow.

The category Process Flow contains 39 weakness patterns, comprising, amongst others, the weaknesses redundant activities, redundant data management, edit document after copying it, and check or examination after dispatch or digitalization.

The following example shows how we specified the pattern *check or examination after dispatch or digitalization* using the model query approach. The weakness describes situations where a (formal) check or (content-related) examination of a document is performed after this document has been sent or digitalized. This should not happen as, since a formal check is necessary, the check should have been done before any further use of the document.

Figure 4 (b) depicts the graphical representation of the *check or examination after dispatch or digitalization* pattern. It consists of a vertex *A*, which is followed by a vertex *B* across a directed path. Both vertices are each is connected to two attributes called *BO* (business object) and *BP* (business procedure), similarly to the pattern described in the last subsection. The first business procedure is assigned terms that describe a transmission or a digitalization, for instance, “hand out”, “forward”, “transfer”, “import”, and “memorize”. The second business procedure is assigned terms that describe a formal check, that is, for instance, “check”, “monitor”, “rate”, and “analyze”. Furthermore, the business objects used by the first and the second vertex have to be the same (i.e., in the global rules, we make the statement *BO(A)=BO(B)*).

4.5 Weakness Category *Information Handling*

The weakness patterns of this category address situations in which the handling of information is inappropriate or inefficient. This includes settings, in which an information is dealt with somehow and later on, this information has to be used in a different way, but firstly has to be transformed in order to use this information appropriately. Another weakness regarding information handling is given if an information is created or stored twice at two different places (e.g., in two different IT systems).

Further weakness patterns do not focus the information itself, but inefficient placement of activities related to some information. Thus, processes may become delayed if information is not provided in time. Those weaknesses also depict unnecessary combinations of activities related to an information (e.g., creation without any use). Also, an improper way of transmission can be detected by analyzing the dispatches and receipts of information and the used communication channels. Finally, inconsistent information storage is identified by patterns depicting either the creation of information within different media or their use which may cause an inconsistency.

The category *Information Handling* contains 14 weakness patterns, comprising, amongst others, the weaknesses *information digitalized twice differently*, *printout sent for signature*, *excessive printing*, and *inappropriate transmission medium*.

4.6 Weakness Category *Technology Switch*

The category *Technology Switch* comprises weaknesses, which denote a change of the used technology for processing an information. This includes all situations where an information is provided by the use of or contained by a certain technology (this may be an IT system, an e-mail, a physical document, etc.) and further not processed by the use of the same one. This change causes an additional activity regarding the transformation of the information, which slows down the process due to extra work. The change of technology may be superfluous if the information is directly provided within the later on used technology and is further only processed with this technology.

Consequently, the weakness patterns are related to an inappropriate archiving of information, the inappropriate usage of software as well as unnecessary printing or an inefficient editing of information.

The category *Technology Switch* contains 6 weakness patterns, comprising, amongst others, the weaknesses *digitalization after printing*, *IT and paper based archiving*, *printing from CD*, and *printing for checking or editing*.

4.7 Weakness Category *Automation*

The category *Automation* contains weakness patterns, which depict any activities or activity combinations that should be automated or supported by automation to improve the activity or the overall process. This comprises an inappropriate usage, creation or handling of information, an inefficient usage of IT systems or a lacking usage of IT systems for the execution of activities. An inappropriate use, creation or handling of information often causes difficulties due to a necessary transformation of the information at later activities, which have to process the information or also may cause unnecessary idle times within a process due to information, which is not provided in time. Furthermore, an inefficient usage of IT systems may slow down a process or at least does not exploit IT functionality. Even worse are situations, in which an activity is performed manually without the usage of an IT system even though an application of an IT system would be useful to simplify or accelerate the process.

The category *Automation* contains 20 weakness patterns, comprising, amongst others, the weaknesses *calculation performed manually*, *digitally available information exchanged personally*, *data transferred manually*, and *information exchanged via fax*.

The following example shows how we specified the pattern *data transferred manually* using the model query approach. The weakness addresses situations, in which some information is received and has to be digitalized afterwards. However, the transfer of the data is performed manually. This is mostly necessary if the data is previously received in a form which cannot be automatically processed. Consequently, the manual transfer of the data is necessary. A special situation is given if the data has to be edited before it is digitalized. In this case, the manual transfer may be justified. In

many other cases, manual digitalization would not be necessary and could become superfluous if the data is received in a digital and automatically processable format.

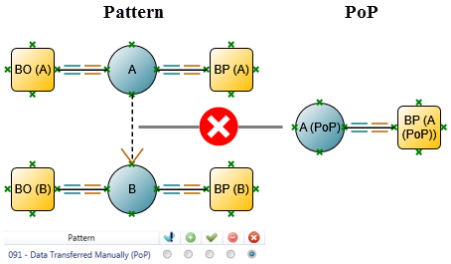

Fig. 5. *Data transferred manually* pattern

The corresponding pattern is shown in Figure 5. It consists of a main pattern and a sub-pattern. The main pattern consists of a directed path between two process vertices *A* and *B*. *A* represents a receive activity (i.e., the activity that accepts the document), and *B* represents the digitalization activity. The business object to be processed is equal for both *A* and *B* (i.e., *BO(A)=BO(B)*). On the path from *A* to *B*, no further activity is allowed that performs an edit task. This means that we specify a sub-pattern that consists of an edit vertex that is assigned a business procedure, to which we assign corresponding terms like “edit”, “change”, or “revise”.

4.8 Weakness Category *Environment*

The category *Environment* contains weakness patterns addressing situations that include external partners but lack well-elaborated communication. Whenever external partners are involved, the partner should not be confused by varying communication channels or varying contact persons. Both are not beneficial for negotiations or agreements and may result in conflicts. In case of varying communication channels, exchanged information may not be centralized in one place and thus, may not be always available when needed. In case of varying contact persons, an agreement made with one contact person has always to be provided to all other contact persons to keep all contact persons informed. If this is not done, confusions within the own company may arise, which can result in confusion and inconvenience of the external partner.

The category *Environment* contains 2 weakness patterns *Varying Communication Channels* and *Multiple Different Negotiations*.

4.9 Weakness Category *Organization*

The category *Organization* includes weakness patterns which occur due to the organizational structure of the company. These weaknesses range from the distribution of responsibilities over the allocation of resources to the behavior of employees grounded in organizational circumstances. An inappropriate or inefficient distribution of responsibilities is given if too many responsibilities are included in one process or two

or more responsibles cannot perform their process tasks in a sequence but have to perform one task and wait for another responsible's task to be completed. In both cases, the process is prone to be delayed or even interrupted if a necessary responsible is on holiday or just overstrained. This congestion can occur as single weakness or – as already stated – in combination with an inappropriate distribution of responsibilities due to a deficient allocation of resources. Furthermore, the given organization of a company may cause an inefficient behavior of employees. In case two or more responsibles have to take part in a process but are spatially dispersed, a high information exchange is necessary or even an unnecessary change of workplace. Additionally, a time difference may also hinder the process flow.

For the detection of situations containing weaknesses as described above, the category *Organization* contains predefined weakness patterns regarding an inappropriate distribution of responsibilities, a deficient allocation of resources and an inefficient employee behavior due to unfortunate location of workplaces.

The category Organization contains 7 weakness patterns, comprising, amongst others, the weaknesses *distributed responsibilities*, *unclear activity responsibility*, *distributed data responsibilities*, and *excessive information exchange*.

5 Evaluation

To assess the usefulness of the identified patterns, we applied them to a collection of business process models taken from real-world consulting projects in *retail*, *supply*, *consulting* and *logistics* businesses (anonymized for privacy reasons). In particular, we evaluated how many of the patters returned matches in the model collection at hand. Afterwards, we analyzed the matches manually if they really were weaknesses or their structure and contents only suggested so. As the basis of identification of the patterns is completely different from the model collection we apply the patterns to (different modeling language *and* business domains), we argue that already a medium matching score could point towards common applicability.

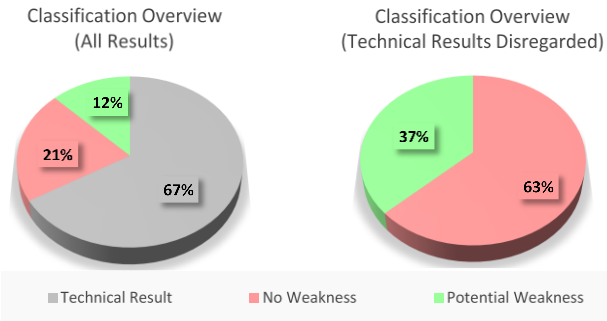

Fig. 6. Amount of weaknesses of the total number of pattern matches

The model collections consisting of four process model databases contained in total 85 business process models, refined through 128 sub-processes and 547 sub-sub-

processes. The models contained 1995 visualized vertices and 5070 process attributes. Hence, the underlying model graph consisted of 7065 vertices in total.

The models had been built using the icebricks modeling language. The four databases were assigned to different modelling projects conducted in different business domains. Hence, their catalogues of terms were different. As already mentioned above, a pattern of the query language we employed has to be adjusted to the properties of a modelling language the models to be searched are designed with. We thus formulated 111 times four patterns. During the specification, we had to exclude some of the patterns due to restrictions either of the different versions of the icebricks language or to restrictions of the query language. Out of the four times 111 patterns, we had to exclude 115 patterns. Hence, we actually searched for a total of 329 patterns.

Altogether, the specification of 329 patterns took 3399 minutes. This makes an average specification time of 10.3 minutes per pattern. 86% of the search runs were finished in about 15 seconds on average.

The search for weaknesses returned 6583 matches in total. Therefrom, we could classify 771 results as actual weakness (12%). 1382 results were classified as no weakness (21%) – their structure and contents only suggested so. Furthermore, we classified 4430 results as technical result (67%) (cf. Figure 6). A technical result occurs, for instance, when the model query returns the same match several times due to graph matching specifics (e.g., if there is an undirected path between two vertices *A* and *B* in a model and if we search for all paths in that model, every path between *A* and *B* is returned twice as the path can either start in *A* or *B*). Assuming that technical results may be avoidable in the future, the actual weaknesses denote a portion of 37%.

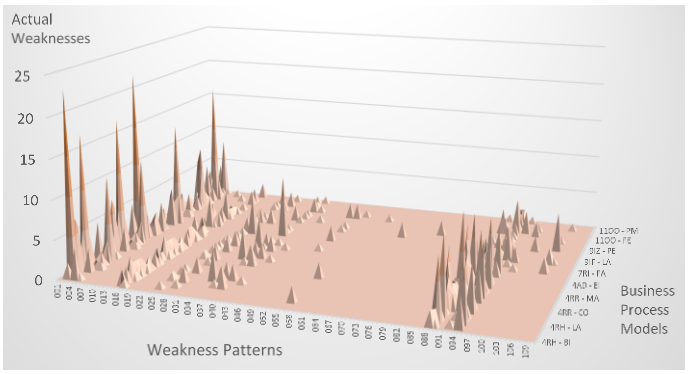

Fig. 7. Weaknesses distributed across weakness patterns and models

Taking a closer look at the results, we see that the detected weaknesses spread widely across 95% of the process models. However, the actual weaknesses we detected were specified by only 35% of the weakness patterns (cf. Figure 7; the abbreviations on the Business Process Models axis denote the main processes of our model collection). The latter may be caused by the fact that the weakness patterns we collected came from public administration processes, hence contained some weakness types that rarely occur in other businesses.

6 Conclusion and Outlook

In this paper, we reported on an empirical study, in which we identified 111 different types of business process weaknesses in a model collection coming from the business domain of public administration. The goal was to establish a commonly usable collection of business process weakness patterns. We see the benefit of such a collection in its reusability for business process improvement, especially when the identification of process weaknesses is supported by process models and automatic weakness detection. To assess the *common* usability of the collection, we applied it to a model collection different from the one we used to derive the patterns, furthermore consisting of models of another modelling language and coming from different business domains. The results show that a considerable amount of the patterns is indeed commonly applicable and returns a high amount of pattern matches, hence potential weaknesses. A further evaluation of the matches showed that part of the potential weaknesses were real weaknesses, whereas part of the matches were actually no weaknesses or technical matches. Although the amount of real weakness matches (12%) seems quite low at a first glance, we argue that identifying 771 weaknesses automatically through reusing a weakness collection is highly promising. If we estimated the time it would take to evaluate the given model collection against only one of the presented weaknesses manually in a model collection consisting of more than 7000 vertices, one can imagine the potential benefit of automatic weakness detection using common collections – especially when comparing it to the average specification time of a pattern (these can be reused!) and the average time it takes to return pattern matches.

Hence, the contribution of our study is as follows: From a research point of view, we contributed a set of documented weakness patterns that can be used as a starting point for efficiently improving business processes. Such weakness collections did not exist up to now, although several approaches to search for them do. Hence, we answered (part of) the question what the input for model query approaches should be. From a practice point of view, a weakness collection as outlined here may help companies to improve business processes and, as a consequence, be more successful.

The following limitations must be reflected, particularly for future research: First, the identified weakness patterns stem from a limited amount of process models, from a defined business domain, and from defined modeling projects. Hence, we cannot claim the weakness collection we presented here being complete. Further studies in different business domains should be conducted to either confirm or falsify the patterns we identified. Second, we applied the weakness patterns to a model collection considering the context of neither the patterns nor the model collection. Although this gives us the chance to assess the common usability of the patterns, the results may be biased as for some patterns, it might not make sense to apply them to that specific model collection. Hence, further studies should consider the business context of the models to be checked. Also, we plan to relate the weaknesses to figures, such as monetary, quality and legal impacts. This way, we could assess the severity of weaknesses and provide recommendations, which of them should be eliminated at first. Finally, we employed a specific model query approach to test our patterns. In further studies, we plan to repeat weakness detection with related approaches.

References

1. Rosemann, M.: Potential pitfalls of process modeling: part A. Business process Management Journal 12(2), 249–254 (2006).
2. Dijkman, R., La Rosa, M., Reijers, H.A.: Managing large collections of business process models – current techniques and challenges. Computers in Industry 63, 91–97 (2012).
3. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Business process model merging: an approach to business process consolidation. ACM Transactions on Software Engineering and Methodology 22, 1–42 (2013).
4. Fahland, D., Jobstmann, B., Koehler, J., Lohmann, N., Hagen, V., Wolf, K.: Instantaneous soundness checking of industrial business process models. In: Proceedings of the 7th International Conference on Business Process Management (BPM 2009). Berlin et al., pp 278–293 (2009).
5. Dijkman, R., Dumas, M., van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: metrics and evaluation. Information Systems 36, pp. 498–516 (2011).
6. Delfmann, P., Steinhorst, M., Dietrich, H.-A., Becker, J., The Generic Model Query Language GMQL – Conceptual Specification, Implementation, and Runtime Evaluation. Information Systems. Accepted for Publication, DOI: 10.1016/j.is.2014.06.003 (2014).
7. Gamma, E., Helm, R., Johnson, R., & Vlissides, J.: Design patterns: elements of reusable object-oriented software. Pearson Education (1994).
8. Imai, M.: Kaizen: The Key to Japan's Competitive Success. McGraw-Hill/Irwin (1986).
9. Davenport, T. Process Innovation: Reengineering work through information technology. Boston (1993).
10. Hammer, M. and Champy, J. A.: Reengineering the Corporation: A Manifesto for Business Revolution. New York (1993).
11. Reijers, H.A., Liman Mansar, S.: Best practices in business process redesign: use and impact. Business Process Management Journal 13(2), pp. 193-213 (2007).
12. Zellner, G.: Towards a framework for identifying business process redesign patterns. Business Process Management Journal 19(4), pp. 600-623 (2013).
13. Becker, J., Bergener, P., Räckers, M., Weiß, B., Winkelmann, A.: Pattern-Based Semi-Automatic Analysis of Weaknesses in Semantic Business Process Models in the Banking Sector. ECIS 2010 Proceedings (2010).
14. Winkelmann, A., Weiß, B. Automatic identification of structural process weaknesses in flow chart diagrams. Business Process Management Journal 17(5), 787-807 (2011).
15. Bergener, P., Delfmann, P., Weiss, B., Winkelmann, A., Detecting Potential Weaknesses in Business Processes – An Exploration of Semantic Pattern Matching in Process Models. To appear in: Business Process Management Journal (2015).
16. Breuker, D., Dietrich, H.-A., Steinhorst, M., Delfmann, P., Outlining a Graphical Model Query Approach Based on Graph Matching. In: Proceedings of the International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2014). LNI P-234, pp. 37-51, Luxembourg (2014).
17. Becker, J., Algermissen, L., Falk, T., Pfeiffer, D., Fuchs, P.: Model Based Identification and Measurement of Reorganization Potential in Public Administrations – the PICTURE-Approach. In: PACIS 2006 Proceedings, p. 114 (2006).
18. Becker, J., Clever, N., Holler, J., & Shitkova, M.: Icebricks. In: Design Science at the Intersection of Physical and Virtual Design. Berlin, pp. 394-399 (2013).
19. Bögl, A., Kobler, M., Schrefl, M.: Knowledge Acquisition from EPC Models for Extraction of Process Patterns in Engineering Domains. In: Proceesdings of the Multikonferenz Wirtschaftsinformatik (2008).