Grass-Root Enterprise Modeling: Issues and Potentials of Retrieving Models from Powerpoint

Achim Reiz[1], Kurt Sandkuhl[1], Alexander Smirnov[2], Nikolay Shilov[3]

[1]Rostock University, Rostock, Germany
[2]ITMO University, St. Petersburg, Russia
[3]St. Petersburg Institute of Informatics and Automation, St. Petersburg, Russia
[achim.reiz, kurt.sandkuhl]@uni-rostock.de, [smir, nick]@iias.spb.su

Abstract. Enterprise modeling (EM) is an established practice in many organizations, but the majority of stakeholders in organizations who produce content relevant for EM use drawing or presentation tools instead of formalized EM techniques. The model-like content of such drawings or presentations often is very valuable for enterprises which calls for a way of integrating it with "real" models and other structured knowledge sources in organizations. This paper investigates how the model-like content of Powerpoint presentations can be extracted and transformed to EM. The main contributions of the paper are (a) an approach for model extraction from Powerpoint, (b) identification of heterogeneities to be tackled during the extraction process and (c) a prototype implementation demonstrating the approach based on ADO.xx.

Keywords: Enterprise Modeling, Grass-Root Modeling, information extraction.

1 Introduction

Enterprise modeling (EM) is an established practice in many organizations and used for various purposes, such as business model development, visualization of the current situation, strategy development, business and IT alignment, and enterprise architecture management. Many application scenarios and experience reports on the use of EM were published during the last decade, for example in automotive industry [2], manufacturing [1], oil industry [3] or healthcare [4]. However, a number of researchers from the EM community argue that EM is more an "elitist discipline" than common practice [5] because the majority of stakeholders in enterprises who produce content or knowledge suitable for EM use drawing or presentation software (e.g., Visio, Powerpoint, Omnigraffle) instead of EM techniques or tools (e.g., ADO.IT, Troux Architect, ARIS). The model-like content of such drawings or presentations often is very valuable for enterprises which calls for a way of integrating it with "real" models and other structured knowledge sources in organizations.

One result of the debate about how to extend the reach of EM in organizations is a recently published research roadmap [6]. This roadmap includes the topic of grass-

root EM which basically describes the vision of people doing EM as part of their daily work, without explicitly noticing formalized modelling approaches and techniques. The general idea of grass-root modelling is related to the concept of natural modelling [26] as flexibility in modeling language or symbols is important in both approaches. One way to implement this idea would be to accept drawings as "local representations" of models for certain stakeholder groups and creating ways of integrating expert modeling and grass-root modeling. This paper aims at contributing to the research roadmap implementation by investigating how the model-like content of Powerpoint presentations can be extracted and transformed, as Powerpoint frequently is used in organizational practice [25]. The main contributions of the paper are (a) an approach for model extraction from Powerpoint, (b) identification of heterogeneities to be tackled during the extraction process and (c) a prototype implementation demonstrating the approach based on ADO.XX.

The rest of the paper is organized as follows. Section 2 gives a theoretical background for understanding diagrams. Section 3 develops these theoretical approaches further to a practical comparing algorithm. This algorithm will be evaluated exemplary in section 4. Section 5 shows current limitations and challenges for the developed approach, section 6 gives a conclusion and an outlook for further research.

2 Background

Diagram theory (section 2.1) and existing work on diagram recognition (section 2.2) form the background for our work and will be discussed in this section. Furthermore, our work is also based on background knowledge from EM. We assume that an enterprise model is captured in an enterprise modeling language with a defined meta-model and a visual notation or diagrammatic representation. More information about EM languages, meta-models and tools is available in textbooks about EM (e.g., [7]).

2.1 Diagram Theory

Visual notations are widely used in enterprise models as diagrams offer significant advantages compared to text: They give an overview about a topic with a high level of abstraction, which brings it closer to the problem domain. They also structure and group information together just by the location of concepts and can easily add perceptual inferences, which are easy to understand for humans and are more memorable than text [8]. Further, the dual channel theory states that diagrams (visual content) and text (verbal content) are processed entirely different within the brain and concludes that information that is processed within both layer can generate a more sustainable understanding and learning [10].

A graphical notation consists of semantics and syntax. The visual semantic contains the constructs that are included and their meaning, the syntax how to represent these constructs. The semantic itself is independent from the notation and could also be represented in a mathematical way [12, p.67]. The syntax though contains the visual variables of the representation: horizontal position (x), vertical

position (y), size, brightness, color, texture, shape and orientation [11]. The more these variables are used, the more information a diagram can carry and the readability increases. By using these variables, data gets represented in a notation.

All topics shown above dealt with the so called primary notation: The formal meaning of representations that is described within the semantics. Every attribute of a variable represents a concept. But often, additionally to the formally necessary concepts, other, free variables are used to display informal information to clarify the meaning. This is called the secondary notation. Examples are the color or the placement of objects: If they are close to each other, it may imply a connection between them that is not modelled formal, or the color indicates the affiliation to a group even though from a notational perspective color is not a meaningful attribute.

The analysis of diagrams is not trivial as the reader has to know different aspects to interpret a diagram the right way: The first and most obvious is the notation itself. If it is not clear what the difference between shapes or a straight and dotted line means, the diagram cannot be understood. Also, to understand a complex, detailed diagram, it is necessary to have certain domain knowledge to put the model into a context. At last, a novice reader tends to misunderstand the secondary notation. Even though the formal concepts are understood, the secondary notation might bear knowledge that is important to interpret. As a result to all these preconditions for model understanding, the modeler often has to create more than one diagram for the same construct to fit the target audience, from a novice reader or a management summary to detailed models for fellow colleagues [9, 11 pp. 772-773].

Models are part of the language. Every language is used for communication and consists of meaningful language elements [12, pp. 64-65]. While humans can interpret a language without a formal notation, computers need an underlying fixed concept. But even though people tend to believe that diagrams are less formal than textual language, this is a widely distributed misconception: It just highly depends on the underlying syntax and semantics to ensure a high degree of formality [12, pp. 69-70].

2.2 Methods of Diagram Recognition

Diagram retrieval is not a new topic for the research community. This section gives an overview about existing approaches for the automated model creation on the basis of documents. Many of the approaches originate from image recognition. While there are approaches that consider the transfer from drawings or pictorial representations to modelling languages with the help of human interaction, like the PICTMOD method [24], this section is dedicated to the fully automated document analysis.

The idea of analyzing graphics into a structured, digital representation engaged since the early 1990's to further fill the sensory gap between the real world object and the computational description as well as to fill the semantic gap between the information that a visual data can give a user and the information a computer can retrieve from the given graphics [13, p.5]. In 1995, Yu et. al. already described the need to convert archives of paper based designs and diagrams to an object-oriented format that is easier to access, update, understand and manipulate [14]. To achieve this goal, image recognition uses pattern recognition and image processing techniques

like vectorization, symbol recognition, analysis documents with diagrammatic notations like electrical diagrams, architectural plans or maps [15, p. 4]. Further, image recognition interacts with other disciplines like multimedia, machine learning, information retrieval, computer vision, and human-computer interaction to build better recognition systems [13, p. 48]. While image recognition in general covers even more aspects like biometrical face recognition [16], image annotation [17] or even handwriting recognition [18], the following overview is limited to diagram recognition as a technique closer to the research topic.

Blostein [19] developed a process for an image diagram retrieval process. At first in the early processing, all unnecessary objects within the image has to be identified and excluded from the analysis process. In the segmentation phase, the distinction between the different symbols has to occur. This is especially challenging if symbols overlap with each other. The last part of the symbol recognition is the recognition itself. This includes shapes, segments of lines that can belong to the shapes in the form of a relation as well as textual elements. The symbol-arrangement analysis covers the relationship between the identified symbols. While the spatial analysis just depends on the position of the objects, the last two steps align the analyzed picture with knowledge about a formal notation. Flowcharts are especially in the focus of image recognition [14, p. 791, 20, pp. 215-216]. They contain Diagram elements as well as logical relations or associations between them, represented by lines which can be directed or undirected [20, p. 216]. As the image recognition systems are getting more and more advanced, it is now possible to also detect handwritten flow charts [21] or to analyze large engineering drawings [14, p. 794]. In the future, new technologies like Deep neural networks are promising to achieve new breakthroughs in the field of image recognition [22, p. 770].

At the one hand, there are a lot of promising approaches towards the analysis of diagrams out of images. At the other hand, for this kind of work, image recognition lays out an additional layer of complexity: a PowerPoint file itself contains not a picture, but shape objects. Prior to the analysis, it would be necessary to convert the slides to an image file format. This would lead to a loss of information: the .pptx file itself stores attributes like connections in form of start and endpoint of a line and form, color and content of shapes. It is therefore not helpful to drop this information and try to retrieve on an image level, but better to analyze the PowerPoint data structure directly if it is available.

3 Model Extraction from Powerpoint

Our approach for model extraction from Powerpoint followed the principal idea that it should be applicable for all kinds of enterprise models and be suitable for as many variations in Powerpoint slide decks as possible. Applicability for all kinds of EM basically implicates that the target meta-model is not pre-defined but can be loaded dynamically during run-time, including the possibility to check what meta-model(s) would fit best to the content of the slide deck. Suitability for Powerpoint variations means no assumptions are made about presentation styles or slide structures.

This principal idea basically results in the need to transform both, Powerpoint content and EM meta-model, into an internal shape-oriented representation which at the same time serves as intermediate format. “Shape-oriented” in this context means that this internal representation is designed for comparing and matching the shapes included in the Powerpoint content and the shapes making up the visual notation of the EM language. The model extraction process consists of four main steps:

- *Retrieving data from Powerpoint*: Analysis of the Powerpoint document and extraction of diagram data and shape information.
- *Retrieving data from meta-model:* Parsing of the meta-model and retrieving of information about the visual notation.
- *Matching:* Analysis of the content retrieved from Powerpoint and the data retrieved from the meta-model with the purpose to find exact or sufficiently similar matches between shapes in Powerpoint and meta-model. This step includes two sub-steps: (a) Structural analysis what shapes occur in slides and if they fit to the shapes in the EM’s visual notation; (b) Semantic analysis if the relationships between the discovered shapes fit to the meta-model of the EM.

The steps introduced above will be elaborated in the next sections.

3.1 Retrieving Information out of PowerPoint

PowerPoint, the SlideShare Program invented in 1984 was at the time a milestone in communication and is a product and trademark of Microsoft. This section first gives an overview about the development of the diagram-retrieval algorithm from Powerpoint as well as the internal storage of the visual attributes in the software prototype developed. Furthermore, aspects of the implementation are shown.

If a Powerpoint file is loaded into the prototype, each slide is opened individually and searched for a suitable diagram. A diagram is marked as usable if it has shapes that are interconnected with each other by a line. The line has to be connected to two shapes, otherwise a valid relationship is not assumed. If such is found, the analyzer converts the diagram into an internal data representation. This internal representation is programmed for the needs of a further analysis. The internal data representation includes a set of Diagrams. A diagram stores a string with the name of the diagram, extracted from the slide title, as well as sets of the object “shape” and “relation”. A shape contains the location and size, as well as the type (e.g. rectangle, ellipse), the stored text – if there is any – and the id. PowerPoint gives each shape a presentation wide unique id which can be used for further identification. A relation contains two shape objects, which maps the start and end point.

Technical Implementation: To retrieve the information out of the .pptx file, the apache POI framework is used with the POI-XLSF component. The XMLSlideShow contains all information from the PowerPoint like the Masterslide-attributes or functions to search for specific data. It contains also a list of slides. The slides are stored in a specific data fragment called XSLFSlide. By iterating over this list, every slide can be accessed. The slide object already provides a lot of function for accessing data as well as altering items. It is possible to create shapes, tables and group items. General information to the used theme, used master slide, layout, title or slide number

can be crawled directly. For more information on the content, the item XSLFSlide has a method getShapes() for getting all placeable data. By iterating the shapes with the type XSLFShape, almost all necessary information can be retrieved (e.g. size, type):

```
diagram.addshape(shape.getAnchor().getX(),
shape.getAnchor().getY(),shape.getAnchor().getWidth(),
shape.getAnchor().getHeight(), shape.getShapeType().name(),
shape.getShapeId());
diagram.addShapeTextById(shape.getShapeId(), shape.getText());
```

With these short statements, the shape is stored in the internal diagram representation. At last, the relations between objects have to be set up. In a PowerPoint file, the connector between shapes is not a relationship itself but also a special kind of shape. By validating the type by checking

```
for (XSLFShape sh : slide) {
  if (sh instanceof XSLFConnectorShape) {
        XSLFConnectorShape line = (XSLFConnectorShape) sh;
```

The algorithm identified the specialized object "line". The XSLFConnectorShape contains not all necessary data out of the box. To identify the connectors of the lines, it is obligatory to traverse the inner XML of this data fragment.

```
XSLFConnectorShape line = (XSLFConnectorShape) sh;
XmlObject xml = line.getXmlObject();
```

After creating an Element object with the saxBuilder, XML can be traversed.

```
Namespace ns_a = lineXML.getNamespace("a");
Namespace ns_p = lineXML.getNamespace("p");
Element connectors = lineXML.getChild("nvCxnSpPr", ns_p).
getChild("cNvCxnSpPr", ns_p);
String id =
 connectors.getChild("stCxn", ns_a).getAttribute("id").getValue();
```

The node `nvCxnSpPr/cNvCxnSpPr` contains `stCxn` and `endCxn` for the start and end ID of the shapes. As this value is already stored in the internal data representation, the corresponding relationship can be set up.

```
public void addRelation (int idStart, int idEnd) {
        this.relations.add(new Relation(getShapeById(idStart),
        getShapeById(idEnd)));  }
```

The example above shows the creation of a new relation. Giving the individual IDs for the start and end shape, a function crawls all existing shapes and returns the Object "Shape" (getShapeById(id) with the right ID.

If all line shapes are converted to object relations, the line shapes can be deleted:

```
for (XSLFShape sh : slide) {
        if (sh instanceof XSLFConnectorShape)
                diagram.deleteShape(sh.getShapeId());    }
```

If the examined item is a group itself, the algorithm assumes an enclosed meaning within the group and extracts the information by retrieving all shapes and running the algorithm recursive: "`diagrams.addAll(handleDiagrams(diagrams,shape.getShapes(), "Group-Shape"));`" The effect is an own diagram object for the group.

3.2 Using ADOxx Libraries as Meta Models

ADOxx is a meta-modeling tool provided by BOC. ADOxx is not specialized for one modelling language, it just provides the underlying construct for developing any

modelling language by creating a meta model. This meta model contains all elements like concepts and the corresponding relations that can be included later in the diagrams. It is also possible to add additional model functionality or validation by programming routines in ADOscript, the proprietary internal script language. With this tool set, it is possible to accurately describe every kind of models like UML, Entity Relationship, BPMN etc. The library containing the meta model is used in this work for diagram analysis. This section shows how ADOxx Libraries can be crawled to retrieve the formal description of the contained Meta Models for further use.

The purpose of the developed software is to be as general as possible. Part of this concept is the idea that every kind of model can be used for analyzing and mapping PowerPoint slides. It is represented in an extensive XML-File, containing all objects with the associated attributes. In the very beginning, the views are identified and internally created. A view is a set of objects that can be put on one diagram type. According to the created view identified by a unique name, the objects are stored internally together with necessary attributes like name and graphical representation. In this state, relations are not yet distinct from objects but a specialized relation object. To separate them, relations within the objects are identified, deleted and the corresponding relation will be stored as a linkage between diagram objects. With this, all necessary information of the meta model is stored and ready for a further analysis.

Technical Implementation: As mentioned above, the meta-model-XML is very detailed and long. While the used example library – a simple Entity Relationship representation – contains already 11555 lines of code, more detailed libraries get even longer: The 4EM-library is stored in 18896 lines of code. The official ADOxx UML library, available in the ADOxx application library [1] is powerful and detailed and has 63854 lines of code in its XML data representation. Even though the size differs strongly, the overall structure of the file does not change. This allows to crawl the documents in a uniformed way. To ensure a convenient and fast document handling, XPATH is used for all queries on the XML-document. First of all, all contained views are identified. In the example case of the ER-diagram, the result is just one item, the "ER diagram". After all views are identified and the internal "view" objects are created, all concepts that belong to a view are crawled. The following exemplary XPATH-Query shows how this is carried out, figure 1 illustrates the result:

```
/library/attributes/attribute[@name="Modi"]/value/leo/*/@val
```

Name	Value	Type
val	ER diagram	Attribute
val	Entity	Attribute
val	Type constructor	Attribute
val	Attribute	Attribute
val	Relation	Attribute
val	has attribute	Attribute
val	links	Attribute
val	Note	Attribute
val	has Note	Attribute
val	Standard	Attribute
val	Documentation	Attribute

Figure 1: XPATH - Views and concepts (Screenshot)

[1] https://www.adoxx.org/live/adoxx-application-library-code-repository

If it is known that “ER-Diagram” is a view, all items following this view object have to be the concepts in this view. In this stage, just the concept object with the attribute name is created. Further attributes are added in the next stage. Now that the names of the concepts are known, it is possible to query the XML-document for the specific data. A challenge is the accuracy of the form description. PowerPoint for example provides precise names for shapes – RECT (rectangle), ELLIPSE, ROUND_RECT (round rectangle)”, TRAPEZOID, PARALLELOGRAMM, TRIANGLE and RHOMBUS – these are just a few examples. A shape type can be easily accessed. In ADOxx, the amount of shape types is much more limited – just rectangle, round rectangle, ellipse and pie do exist as simple, directly accessible types. Shapes that are not found within these categories have to be built by a “polygon” attribute, an attribute that can represent every different form by modelling lines and curves. Its graphical vocabulary is comparable to the one in the scalable vector graphics format (.svg). For further comparing towards a PowerPoint shape, the mathematical descriptions of the form have to be accessed and analyzed. The example below shows the examination for two graphical forms with 4 coordinates.

```
if (x.get(0) == -1 * x.get(2) && y.get(1) == -1 * y.get(3))
        cm.addUniqueGraphtype("RHOMBUS");
else if (x.get(0) == -1 * x.get(2) && x.get(1) == -1 * x.get(3)
                && y.get(0) == y.get(1) && y.get(2) == y.get(3))
cm.addUniqueGraphtype("PARALLELOGRAM");
```

These checks – here are just 2 shown, the software itself provides more - assure that complex PowerPoint shapes can be matched to a class stored in the Meta Model. Very complex forms though or those who are not integrated in the checking algorithm trigger a fallback towards a generic “polygon” representation. Every PowerPoint slide has a representational counterpart in the Meta Model analysis. If not and the PowerPoint shape is very complex, the PowerPoint will also be stored as “polygon” within the internal representation. Within a considerable amount of these kind of invalid shapes, the document can still be analyzed and matched.

3.3 Comparing Diagrams and Models

To measure PowerPoint diagrams against a meta models, the diagrams will be checked in two different ways – structural and semantical. While the structural analysis is limited to the form of the shapes, the semantical is more complex and also considers possible interconnections between them. The following Proof of Concept is carried out at the example of an entity relationship diagram. The result will be written to a .csv file for further analysis and testing of the algorithm.

3.3.1 Structural Analysis

Even though the concepts between shapes in PowerPoint and ADOxx are completely different, the similarities regarding to the look of different shapes can be assessed. Figure 2 shows the different representations of shapes. Even though almost all parameters can be altered within a shape, it is normally possible to identify one top level category of a form. In the structural analysis, the software takes these top-level categories and compares if the shape types in the PowerPoint slide are also found within the objects in a view of an meta model.

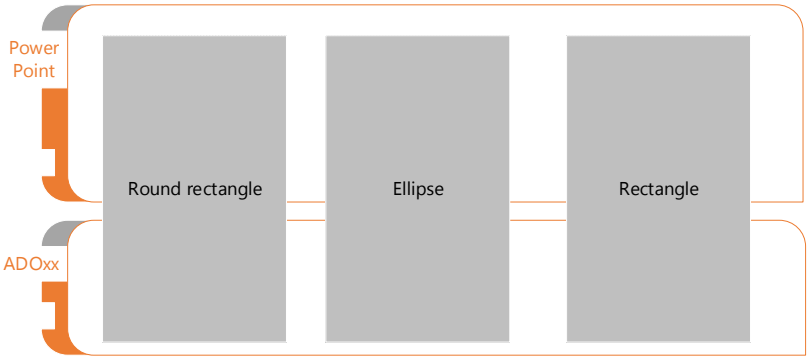

Figure 2: Shapes in PowerPoint and ADOxx

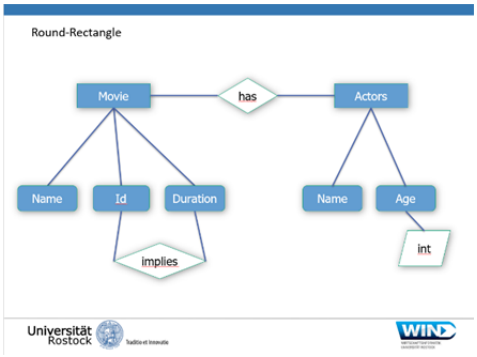

Figure 3: Slide loaded into Structural Analysis

In the example of an Entity Relationship diagram, there are 3 important types of forms: An entity (rectangle), a relation (rhombus) and an attribute (round rectangle). The structural analysis now crawls through every diagram and compares if the item contains any form that is not included within the meta model. The output consists of a .csv file that prints out the shapes that do not fit. In Figure 3, the example slide loaded into the structural analysis is shown. The Entities “Movie” and “Actor” are connected with a relation. Age is wrongfully connected with a parallelogram to outline the data type, but there is a connection between “Id” and “Duration” with a relation as well. This is also not possible in an Entity Relationship diagram.

The algorithm detected that the parallelogram is not a valid content for an ER-diagram. Compared to the goal of an accurate analysis of a diagram fit, it is more and more clear that the focus just on the occurrence of shape forms is not enough. Even though the distinction for forms works properly, the structural analysis does not detect incorrect relationships like the relation between the attributes “ID” and “Duration”. The check for relations between shapes is carried out in the semantical analysis.

3.3.2 Semantical Analysis

The structural analysis works especially for a check towards small meta models. With an increase of the amount of possible and considerable shape forms, coming from more meta models to check or larger libraries, the structural analysis is more

likely to predict a false outcome. The semantic analysis offers a solution to this kind of problem by not only considering the form of the shapes, but also the interconnection between them. The semantic analysis will be explained at the ER-Example. An entity can be connected with the attribute and the relation shape, but there is no connection possible between attribute and relation. Also, all shapes can be connected with the same kind, relations with relations, attributes with attributes, entities with entities. This information is stored in the meta model as well. Combined with the structure, it can be identified that rectangle and rectangle, rectangle and rhombus, rhombus and rhombus, rhombus and round rectangle, round rectangle and round rectangle as well as round rectangle and round rectangle can be interconnected.

Crawling through a diagram frame, the software analyses the connections between the different forms and counts those, who have a valid connection. While the structural analysis could not find a problem with the connection between the attribute and relation, the semantical analysis does: not only the "int" shape, but also the "implies" shape is identified as an object without the proper meaning. The results are now accurate and provide the right results for the given PowerPoint.

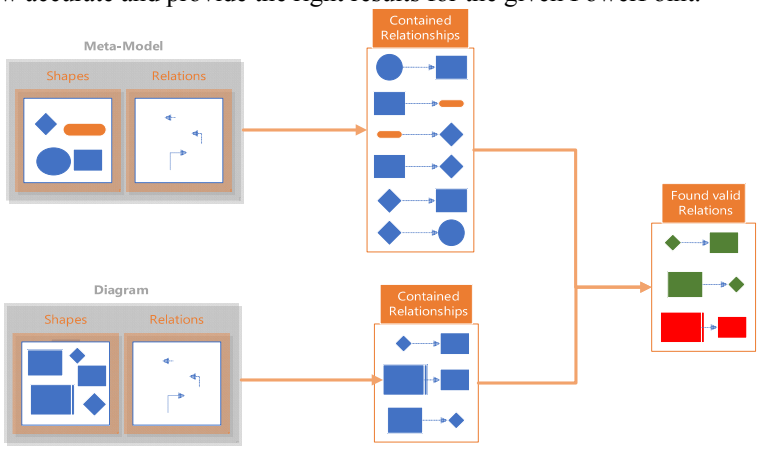

Figure 4: Structural Analysis - Comparison between Diagram and Meta-Model Relations

4 Experimental Evaluation

We evaluated the software by testing various scenarios containing ambiguities and heterogeneities. This sharpens the understanding of what the algorithm is capable of and where the limitations are. Exemplary, the processing of groupings is included in this section to give an understanding how a minor change for modelling purpose can create major changes in the data structure.

As the grouping implies structure and meaning and also alters the data structure, the processing currently comes with the cost of the loss of information. Figure 4 contains an example for a grouping of shapes. The background shape "Grouped" is

just for a better visualization in the example and neither connected nor grouped with the shapes representing the model and gets therefore not analyzed.

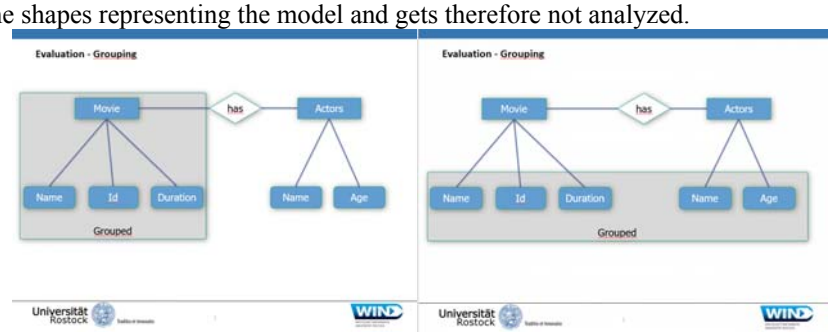

Figure 5: Grouped Shapes – Example 1 (left) and Example 2 (right)

As described in section 3, the retrieving algorithm stores the grouped shapes in a separate data object. As a result, the single slide is represented by two independent data fragments: "Group-Shape" as a representation of the grouped items and "Evaluation-Grouping" containing the rest of the items that are placed directly onto the shape. This example shows the actual problem of grouping: Beside the fact that one relational information is lost (the connection between "Movie" and "has"), the storing of the group is semantically not correct. Even though the group does not contain any semantical information on the visualization perspective, the handling of these kind of shapes is not trivial and can lead to major problems regarding the understanding of diagrams. Figure 5 shows the two examples used in the following.

Diagram Name	ShapeType	ShapeText
Group-Shape	RECT	Movie
Group-Shape	ROUND_RECT	Name
Group-Shape	ROUND_RECT	Id
Group-Shape	ROUND_RECT	Duration
Evaluation – Grouping	RECT	Grouped
Evaluation – Grouping	FLOW_CHART_DECISION	has
Evaluation – Grouping	RECT	Actors
Evaluation – Grouping	ROUND_RECT	Name
Evaluation – Grouping	ROUND_RECT	Age

Table 1: Grouped Shapes – Internal Data Representation – First Example

While in the first example the loss of relational information is still minor, with certain grouping scenarios they can result in a major inability for understanding the diagram. In Table 1, all attributes are grouped. While a semantical meaning could be interpreted, the fact that the grouped shapes are examined separately leads to a misfunction of the algorithm.

As diagrams are just getting stored as part of a model if they have a connection towards another element and attributes are not connected with each other, they are

considered as a separate model. The grouped shapes do not have any valid connection now. As Table 2 elucidates, the attributes were not stored in the internal data representation at all.

Evaluation – Grouping	RECT	Grouped
Evaluation – Grouping	FLOW_CHART_DECISION	has
Evaluation – Grouping	RECT	Actors
Evaluation – Grouping	RECT	Movie

Table 1: Grouped Shapes – Internal Data Representation – Second Example

5 Ambiguities and Heterogeneities in Office Documents

Mukherjee et. al. [23] described in their work 6 types of different ambiguities in Office Documents. They can be mapped into two categories: Structural and semantic. Structural ambiguities are the result of an unclear representation, semantic ambiguities origin in a designing process with no formal description language.

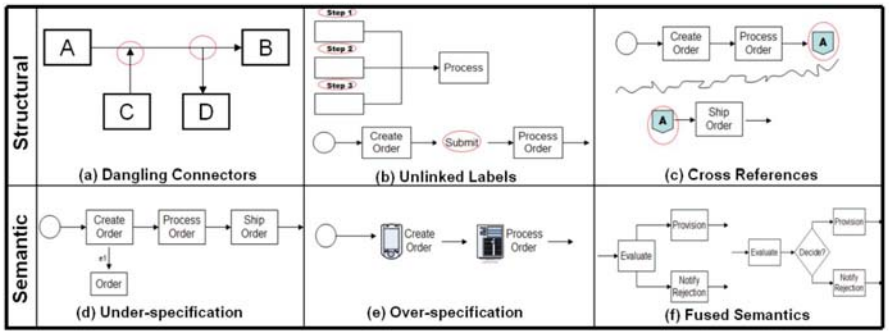

Figure 6: Semantic and Structural Ambiguities (adopted from [23])

All heterogeneities described above in Figure 6 were discovered and considered in this research project. Furthermore, additional "heterogeneities" were discovered:

Grouping: A normal shape has a connector to a different shape. If a representational object is built out of different shapes and grouped, these connectors often relate to the group, not the individual shape. While it is possible to unbox a group and get all different shapes to store them separately (an approach also used in this application), it is unclear how the shapes are connected with the rest of the diagram. Storing a whole group just shifts the problem: The group item is not yet unpacked and the contained information not yet ready for further analysis. Groups might contain important explicit information, but it is also realistic that they just contain implicit information, not related to the formal notation. As PowerPoint is primary a design tool, the groups may not contain any knowledge at all and are just used for shifting sizes and positions of a larger amount of shapes. Also, it is possible

to create a complex representation out of a group of simple shapes. In that case, just the whole group can be seen as one shape.

Multiple Diagrams in one Slide: The retrieval algorithm searches slide by slide and stores them into the diagram data representation. In the case that two different diagrams are drawn in one slide, they both are stored in one diagram data frame. From a programming perspective, the distinction between diagram parts that belong to each other but have no connection and two different diagrams with different meanings is not possible. Even though no shape or relational information would be lost, the storage of these two diagrams in one data frame is semantically not correct.

One Diagram in multiple slides: Opposite to the ambiguity named above, one diagram stored in more than one slide cannot be retrieved properly as well. As the crawler searches slide by slide, this diagram will be stored in multiple diagram data frames, important semantical information will be lost.

Diagram stored as a picture: As PowerPoint is just a presentation tool, the diagrams stored in it often come from visualization tools outside of PowerPoint like Microsoft Visio. While a few might be linked to the origin file and stored as an Object, most of the external visualizations are stored as an image. In the first case, it would be possible to analyze the origin language to retrieve the information. But if the diagram is a picture, it is necessary to use advanced image recognition for retrieving the data out of the picture. The methods proposed in this paper cannot applied.

Nested Shapes: Relations between shapes are not necessarily stored via groups or connector shapes. It is possible that shapes are nested in each other to represent e.g. “has” or “is a” relations. While the order and overlap of shape could contain important information, the shape could also be a simple background with no further meaning. Additionally, these kinds of overlays are mostly not supported by formal visualization languages. For the analysis, it is unclear how to interpret these shapes.

Directed and Undirected Relations: While the Meta Model always contains a direction, the drawing nature of PowerPoint also allows undirected visual representations. On a data perspective though, PowerPoint does store directions – even a line without arrows has a start and endpoint in the underlaying data base.

Microsoft SmartArt: SmartArt and allows the user to build easily complex diagrams like hierarchies, relationships, matrix, pyramids out of predefined visualizations. While the meaning is clear from a visual perspective, the shapes do not use a connector which is easy to analyze. If the shapes are clearly directed and in relation to each other, but apart of the special form used, there is no indication for the relations inside the diagram. Another challenge is the internal data representation of SmartArt. SmartArt is normally not stored like a normal diagram in shapes but uses a special SmartArt-Data representation.

Reading Relation Types: In structured visualization languages, the relation type is stored within a relation object. Yet, Powerpoint does not allow to add text to a connector shape. To describe a relation, a textbox has to be added and placed nearby the connector shape. While it is easily readable for a human interpreter, on a data structure perspective, the textbox and the connector shape are not connected with each other. To bind these two elements, the position of the elements has to be compared to identify if an unconnected textbox is placed near the connector shape.

6 Summary and Conclusions

In this paper, a concept for retrieving models from Powerpoint to the modelling tool ADO.XX was presented. The objective was to investigate possibilities and limits of using office tools, such as Powerpoint, for grass-root EM. We showed how the software prototype retrieves data from slideshows and meta-models from XML representations into an internal data representation. For retrieving PowerPoint data, "Apache POI" was used with additional effort to access information that is not provided by POI. The Meta Model originated of an XML-ADOxx export read by extensive XPATH queries to fetch the overall structure and behavior of the concepts. Two phases of matching were considered, showing an analysis on the level of occurrence of shapes (structural analysis) and with consideration of the relations (semantical analysis). The developed algorithms reach their limits for large meta models. If the notation describes a similar representation for more than one concept, the distinction just on the basis of the appearance is not enough to explicitly identify a model type with the corresponding concepts. Also, the focus of Powerpoint as a drawing tool sets challenges in the occurrence of heterogeneities.

Further research ought to make the software more practically applicable with enabling it to read even inconsistent modelled shapes by solving these heterogeneities. While for the most inconsistencies a way to resolve them is already proposed and requires mainly programming effort, especially the detection of implicit semantic needs more attention in the future and also conceptual research. For further validation of the retrieving algorithm towards different approaches, a comparison between the shown document retrieval methods and the practice of image recognition is recommendable. After the comparison, the possibility of the integration of both methodologies can be assessed. To enable the analysis towards larger Meta Models, further research has to address the limitations of the limited amount of possible optical variables with the identification of further matching criteria. Especially the usage of the (textual) content of a shape could be an additional considerable input.

The possible technical integrations of this kind of algorithms are broad and diverse. On a strategic perspective, the finalized algorithm can not only contribute to a more efficient knowledge management but also help to spread the usage of Enterprise Modelling through the organization without the need for a dedicated training for every modeler. It therefore can be part of a foundation that enables a bottom up grass root modelling.

Acknowledgements. The research was supported partly by projects funded by grants # 18-07-01201 and 18-07-01272 of the Russian Foundation for Basic Research, and by Government of Russian Federation, Grant 08-08.

References

1. Wortmann, J.; Hegge, H. and J. Goossenaerts. 2001. "Understanding enterprise modelling from product modelling." Production Planning & Control Vol. 12 , Iss. 3, 2001.
2. Lillehagen, F. and J. Krogstie. 2008, "Active Knowledge Modelling." Springer, 2008.

3. Wesenberg, H. (2011, November). Enterprise modeling in an agile world. In IFIP Working Conference on The Practice of Enterprise Modeling (pp. 126-130). Springer, Heidelberg.
4. Stirna, J., Persson, A., & Sandkuhl, K. (2007, June). Participative enterprise modeling: experiences and recommendations. In International Conference on Advanced Information Systems Engineering (pp. 546-560). Springer, Berlin, Heidelberg.
5. Sandkuhl, K., Fill, H. G., Hoppenbrouwers, S., Krogstie, J., Leue, A., Matthes, F., ... & Winter, R. (2016, November). Enterprise modelling for the masses–from elitist discipline to common practice. PoEM 2016, pp. 225-240. Springer, Cham.
6. Sandkuhl, K., Fill, H. G., Hoppenbrouwers, S., Krogstie, J., Matthes, F., Opdahl, A., ... & Winter, R. (2018). From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. BISE, 60(1), 69-80.
7. Sandkuhl K., Stirna J., Persson A., Wißotzki M. (2014): Enterprise Modeling – Tackling Business Challenges with the 4EM Method. Springer, ISBN 978-3-662-43724-7S.
8. J. LARKIN and H. SIMON, "Why a Diagram is (Sometimes) Worth Ten Thousand Words," Cognitive Science, vol. 11, no. 1, pp. 65–100, 1987.
9. M. Petre, "Why looking isn't always seeing: Readership skills and graphical programming," Commun. ACM, vol. 38, no. 6, pp. 33–44, 1995.
10. R. E. Mayer and R. Moreno, "Nine Ways to Reduce Cognitive Load in Mul-timedia Learning," Educational Psychologist, vol. 38, no. 1, pp. 43–52, 2003.
11. D. Moody, "The Physics of Notations" IIEEE Trans. Software Eng., vol. 35, no. 6, pp. 756–779, 2009.
12. D. Harel and B. Rumpe, "Meaningful modeling: What's the semantics of "semantics"?," Computer, vol. 37, no. 10, pp. 64–72, 2004.
13. R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval," ACM Comput. Surv., vol. 40, no. 2, pp. 1–60, 2008.
14. Y. Yu, A. Samal, and S. Seth, "A System For Recognizing A Large Class Of Engineering Drawings - Document Analysis and Recognition, 1995., Pro-ceedings of the Third International Conference on," IEEE, vol. 1995.
15. Josep Lladós and Young-Bin Kwon, "LNCS 3088 - Graphics Recognition. Recent Advances and Perspectives,"
16. S. Z. Li and A. K. Jain, Encyclopedia of biometrics. New York: Springer, 2009.
17. U. Ojha, U. Adhikari, and D. K. Singh, "Image annotation using deep learn-ing: A review," pp. 1–5.
18. M. Kozielski, P. Doetsch, M. Hamdani, and H. Ney, "Multilingual Off-Line Handwriting Recognition in Real-World Images," in 2014 11th IAPR Inter-national Workshop on Document Analysis Systems, Tours, France, 2014, pp. 121–125.
19. D. Blostein, "General diagram-recognition methodologies: Methods and applications". University Park, PA, USA, 1995 selected papers," vol. 1072, pp. 106–122, 1996.
20. J. Sas and U. Markowska-Kaczmar, "Logical Structure Recognition of Dia-gram Images," in 2015, pp. 215–224.
21. Aurélie Lemaitre, Harold Mouchère, Jean Camillerapp, and Bertrand Coüasnon, "Interest of Syntactic Knowledge for On-line Flowchart Recogni-tion,"
22. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," pp. 770–778, 2016.
23. D. Mukherjee, P. Dhoolia, S. Sinha, A. Rembert, and M. Gowri Nanda, "From Informal Process Diagrams to Formal Process Models", BPM 2010, vol. 6336, pp. 145–161, 2010.
24. Fill, Hans-Georg, "Bridging Pictorial and Model-based Creation of Legal Visualizations: The PICTMOD Method", IRIS 2015, Jusletter IT 26. February 2015, ISSN 1664-848X
25. Ciriello, Raffaele; Richter, Alexander; Schwabe, Gerhard: "PowerPoint Use and Misuse in Digital Innovation", AIS Electronic Library 2015. http://aisel.aisnet.org/ecis2015_cr/.
26. M. Bjekovic, J. Sottet, J. Favre and H. A. Proper, "A Framework for Natural Enterprise Modelling," 2013 IEEE 15th Conference on Business Informatics, pp. 79-84.