From Model Extraction to Model-based Reuse of Enterprise Documents

Biplav Srivastava, Debdoot Mukherjee, Rema Ananthanarayanan, Vibha Sinha

IBM Research - India
New Delhi, India
{sbiplav,debdomuk,arema,vibha.sinha}@in.ibm.com

Abstract

The challenge in building software applications in services is that they should be readily adaptable to customers' needs. The Model-driven Architecture and Design approach promises to address this if the *model*, i.e., a formalization of domain concepts and their interrelationships, is given. However, building a domain model from scratch is time-consuming and cumbersome. An appealing approach is to bootstrap model acquisition by learning a probable model from secondary data sources like design documentation or web. In this paper, we propose a semi-automated approach to discover an information model, comprising of entities and relationships, by mining documentation about a particular domain as captured in Word documents. The approach was validated by both comparing it with manually created models in one controlled setting and by verifying with experts in another setting and was found to be accurate, efficient and useful for building applications for SAP engagements. The approach is especially invaluable in large service engagements situations where no single expert knows the full domain to give the model or the model changes with customers and time.

1 Introduction

To build high-quality information systems, the trend today is to use Model-driven Architecture and Design (MDA/MDD) where-in concepts of the domain are captured as models and used at every stage of development to keep the system implementation on track. In services, for software applications that need to be built during engagements to meet customer requirements, the domain includes both customer dependent and customer-independent concepts (e.g., industry-specific). Furthermore, apart from model changing with each customer, no single expert may know the full domain to specify it upfront.

International Conference on Management of Data
COMAD 2010, Nagpur, India, December 8–10, 2010

However it is well known that building the models from scratch is time-consuming and cumbersome[9, 19]. The bottleneck issue is the availability of qualified domain experts. In services settings, it is quite common to have large development projects spread across multiple locations and spanning months and many teams. Different teams are engaged in complementary activities for the project and create different types of documentation as their output. Over time, the participants become experts in the domain concepts corresponding to the activities they were engaged in. But for a complicated domain, no single person may be the authoritative expert.

An example of such a domain is the the multi-year business transformation projects using SAP's packaged applications (called SAP project, for short). These projects involve hundreds of consultants and at least 12 types of documents are created during design dealing all the way from process description to data conversions and testing[1]. No single expert knows the global model of SAP projects while its sub-models are evidenced in the different document types. The authors worked with SAP business consultants (domain experts) to create a sub-model for SAP customization domain that reflected information captured in two document types. The sub-model took 2 weeks to build by interviewing subject matter experts, browsing through project documents and having different experts negotiate to build a consensus; and then many months to stabilize. Figure 1 shows a fragment of the model provided by the experts. The prohibitive effort in extending this partial model to information in other document types was the main motivation for this work.

An appealing approach to bootstrap the model acquisition gap is to learn the model from secondary data sources like web[5] or design diagrams[8]. However, the authors are not aware of any approach to learn

[1] Document types used in paper (total 12): Process Description Documents (PD), Business Process Procedure (BP), Test Conversions (TC), Interface Definitions (ID), Technical Design (TD), Enhancement Design (ED), Data Conversions (DC), Function Specification (FS), Create (CR), Define (DE), Execute (EX), Contracts (CO)

models from enterprise documents that are created by word processors and where the domain model is fragmented into sub-models as reflected in the different document types. We fill this gap with a method that automatically harvests documents to separate content from presentation, identifies candidate sub-model elements within a document type and relationships across document types, consolidates and learns the aggregated (global) model of the domain; and has well-defined manual review steps to make domain-specific decisions. The approach was validated by both comparing it with manually created models in one controlled setting and by verifying with experts in another setting. It was evaluated on different types of documentation created in SAP projects and it was found to be accurate (up to 77% output concepts matched that by experts, rest are new concepts) and efficient (completed in less than 3 hours for under 100 documents and up to 10 document types). The approach is especially invaluable in situations where no single expert knows the full domain to give the model or the model changes with time. Finally, we also show that the created models realize their potential usage.

Our contributions in the paper are that we: (1) propose unstructured formatted enterprise documents as a source of model, (2) provide method for learning sub-models from document types and then aggregating sub-models across document types, (3) demonstrate the accuracy and efficiency of the approach, and (4) demonstrate the usage of learned model. In the remainder of the paper, we give preliminaries describing the terminology used, motivation and related work; then present the problem and the solution, followed by experiments demonstrating the benefits of the approach. Next we show the usage of the learned models and conclude with pointers to future work.

2 Preliminaries & Related Work

Document: A document captures the output of a specific activity of interest in a domain. At its simplest, a document is a collection of plain text with some formatting information for the text. In addition, the document can contain objects such as diagrams (e.g., entity-relation) and multi-media objects.

Semi-structured document: A semi-structured document is a type of document whose content is optionally structured with formatting constructs (like document title, section headings and appendix). Word processors allow complicated visual formatting as well[18]. A semi-structured document can be represented in XML format. All documents we consider in the paper are semi-structured documents.

Document Type (Category): A document type documents the output of a type of sub-activity of interest in the domain.

In enterprises, team-based document creation is wide-spread. Such documents typically start from

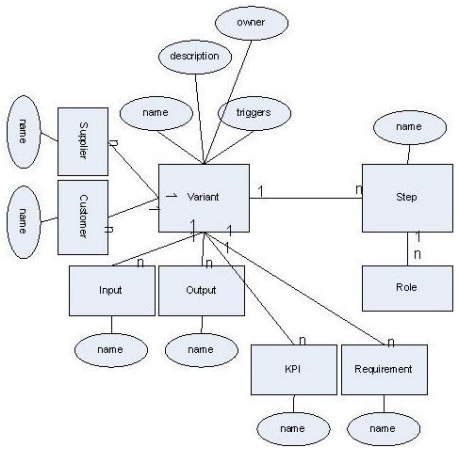

Figure 1: Fragment of a model specified by an expert. We refer to this later as M^*_{PD-BP}.

mandated templates but end up with rich variance. Melnik[12] defines two types of metadata: *control* metadata which is used to structure the sub-parts of documents regardless of their content and *guide* metadata which are used to structure the content recorded in the document. A project template is made up of *control* metadata like document title, approver list, table-of-content and appendix.

Example: Consider the domain of telecommunications. An enterprise may be engaged in software development for this domain and generates different documents. During software development for telecommunications, acceptance test specifications is a specific type of document that specifies the checks needed to prove that the software meets the stated requirements. It is made up of *guide* metadata like preconditions, test-path, success-criteria and coverage; and this type of metadata will vary from one document type to another. The set of documents during software development for telecommunication is recorded by a set of documents, D, made up of documents, d_i. Each document is of a particular type, t_j, corresponding to some sub-activity in the domain (e.g., acceptance testing). Hence, the set D is also $\{\prec D_j, t_j \succ\}$ where $D_j = \{ d_i \text{ s.t. } \text{type}(d_i) = t_j\}$. In the domain of SAP projects, there at at least 12 document types (t_j).

Related Work: Our effort to fill the model acquisition gap has overlap with the areas of wrapper induction, ontology creation and integration. Automated wrapper induction is a related problem that has also been well studied [4],[17] to identify template patterns and extract information from the web. The prior work does not consider enterprise documents as a source with their complexities of unrestricted formatting, object embedding and unstructured text.

Ontology techniques have been considered for extracting and integrating schemas. In [6], the authors

describe a conceptual modeling approach for data extraction from the web where ontology is used to describe the data of interest, including relationships and lexical appearance. Output from parsing the ontology is used to recognize and extract data from unstructured text. In our context, the output models can be considered as an ontology while the input is from enterprise documents.

The direction that we are looking at is best captured in literature under the head of *Information Extraction from Semi-structured data (SSD)*. A good survey for information extraction techniques is [16]. Our work is unique in the type of input (formatted structured documents created by word processors) and the variety of document types considered.

3 Problem

We are given a document collection D and each of its document d_i belongs to a document type t_j. A document type has associated with it, explicitly or implicitly, a set of concepts C.

Concept: A concept is a term or phrase that denotes some meaningful information in the domain of interest. Concepts are linked to each other through relationships.

Relationship: A relationship denotes how concepts are related to one another.

Model: A model is a representation of a collection of concepts and relationships between them. The concepts are formally represented as *models* (M), a graphical representation, where nodes (M_n) represent concepts and edges M_e represent relationships between nodes.

Now, each document type t_i records a fragment of the global model M, represented M_{t_i}. Further, $M \supseteq \bigcup_i M_{t_i}$, since all available document types may still not record the global model. We loosely call the aggregate of all the sub-models as the global model since the only evidence of the domain is from the available document types. Figure 1 shows an example of a model provided by an expert. Our aim is to learn such models from documents.

Putting precisely, given a set of documents from some domain, our goals are: (1) find the sub-models M_{t_j} corresponding to each document type t_j, (2) find the aggregate model M.

4 Solution

Our approach for solving the problem needs to handle the noise and variability of the individual documents, the fragmentation of domain model as evidenced in different document types, and the need to have a user review, enhance and work productively with the resulting model. The proposed approach is summarized in Figure 2 and explained further in the section.

In steps 1-3, we first identify model elements that may have been used to structure the documents (i.e., control metadata or *template* concepts). Such elements are organization or project specific and do not reflect any domain concept. Next, in steps 4-6, domain model elements (domain concepts) are discovered on the complete document pool. This has the potential to overcome the noise of individual documents and focus on *candidate concepts* from the global model as well as sub-models of individual document types. The candidate concepts are characterized by high support in the corresponding document collection. In steps 7-8, now detailed relationships (links) are learned from individual documents of each document type, *but only for candidate concepts*. In Step 9, the sub-models are merged and finally, in step 10, it is output in any suitable representation.

4.1 Learning Concepts

Figure 3 gives the method we use to find the potential model elements in a pool of documents of potentially different document types. The method first parses the documents to identify demarcating text fragments (pre-processing, lines 1-7) and then looks at their statistical significance to determine if a fragment is a potential model element (main steps, lines 1-3). The method builds on the document harvesting techniques of [18].

Algorithm: **LearnModel**
Inputs: A set of documents D with subsets D_i of type t_i: $\{\prec D_i, t_i \succ\}$
Output: A model

Main Steps:
1. Run `Concept-Learner` on D, with *raised* thresholds, to find template concepts
2. Manually review output
3. Remove template concepts (add in stop list)
4. Re-run `Concept-Learner` on D, with *reduced* thresholds, to find likely concepts
5. Manually review output
6. Run `Concept-Doctype-Refiner`, to find likely concepts per document type
7. Run `Linkage-Learner`, to find linkage/ relationships across document types
8. Consolidate concepts and relationship links
9. Manually review learnt links
10. Output learnt model in requested format

Figure 2: Pseudo-code of Model Learning.

Algorithm: **Concept-Learner**
Inputs: 1. A set of documents D
2. *LOW* and *HIGH* thresholds
Output: A set of concepts, M

Processing for each document d_i in D:
1. (MS Word specific) Convert d_i to XML representation
2. Parse d_i
3. Group characters along word boundaries
4. Group words along paragraph (formatting) boundaries
5. Record paragraphs
6. Remove non-textual and non-formatting content
7. Identify possible concepts at paragraph boundaries

Main Steps on document pool, D
1. Collect the list of concepts in the pool and their frequency
2. Establish thresholds to establish segments that are neither rare (*LOW*) nor overly abundant (*HIGH*)
3. Filter and return concepts that are within the thresholds

Figure 3: Pseudo-code of `Concept Learner`

Algorithm *Concept-Learner* works on the complete document pool and produces overall candidate con-

cepts for the domain. Depending on the thresholds, it can be used to find concepts at a particular level of support in the document pool, and we run it separately to find control and guide metadata (concepts).

In Algorithm *Concept-DocType-Refiner* (Figure 4), the candidate concepts are used to find sub-models by document types (Lines 1-3) and their co-occurrence in each document type (Lines 4-7). The latter is only information about direct links between model elements.

Algorithm: **Concept-DocType-Refiner**
Inputs: 1. A set of concepts C,
2. $D = \{\prec D_i, t_i \succ\}$ (Document subsets and doc types)
Output: 1. $C = \{\prec C_i, t_i \succ\}$ (Set of concepts per doc type)
2. $L_c = \{\prec c_i \overset{T_k}{\rightarrow} c_j \succ\}$ (Set of co-occuring concepts and associated document types)

1. For each t_i
2. $\quad C_i = \bigcup_j c_{d_j}$, d_j in D_i
3. $C = \bigcup_j C_i$
4. For each d_i
5. $\quad$ For each co-occurring concept pair, c_m and c_n in c_{d_i}
6. $\quad L_c \bigcup= \{\prec c_m \overset{\{t_{d_i}\}}{\rightarrow} c_n \succ\}$
7. Consolidate L_c for elements with same concept pairs

Figure 4: Pseudo-code of `Concept DocType Refiner`

4.2 Learning Relationships

In order to find deeper links between model elements, we turn to algorithm *Link-Learner* in Figure 5. Here, in Steps 1-3, processing happens iteratively on each document of a document type to extract its hierarchical structure made up of content containers (e.g., paragraphs, tables) and their tree-based ordering. Then, the content of all the documents are *merged* around the trees' hierarchical structure. Now in Steps 4-5, the tree is searched for all adjacent container pairs (4a) and they are used to infer directed relationships between model elements. If the links are between candidate concepts, they are considered (4b), otherwise ignored. Finally in Step 6, all transitive relationships among included links are deduced.

Algorithm: **Link-Learner**
Inputs: 1. A set of documents D_i of type t_i
2. Candidate concepts, C
Output: $L = \{\prec c_s, c_d \succ$ (Set of pair of links from source to destination concepts)
1. For each document d_i
2. $\quad p_i$ = Parse d_i's hierarchical structure
3. P = Merge($\bigcup p_i$)
4. For each *container pairs* in P, c_m and c_n, where (a) c_n is a child of c_m and (b) $c_m, c_n \in C$
5. $\quad L \bigcup= \{\prec c_m \rightarrow c_n \succ\}$
6. Transitively, find all relationships in L

Figure 5: Pseudo-code of `Link Learner`

In addition to learning the links, we can also distinguish their different types:
Containment: The *Link Learner* algorithm as shown in Figure 5 learns only containment relationships between concepts, $c_i \in C$. Containment stems from two types of formatting elements: Section and Table. A section may contain lists, paragraphs, tables and other sections. A concept captured in a section heading is deemed to *contain* concepts, which are encapsulated by formatting units contained in the section. Similarly, a concept represented at the level of a table row is said to *contain* concepts listed at each of the cells.
Co-occurrence: The links output by *Concept DocType Refiner* capture co-occurrence among concepts. Furthermore, in the graph produced by *Link Learner*, concepts that are siblings are said to *co-occur*. For example, concepts in different paragraphs of the same section co-occur with each other; concepts in the various columns of a table co-occur, etc.
Similarity: If two nodes in the learnt graph have identical (or nearly identical) sub-trees rooted at them but the concept names discovered for them differ, then the concepts are marked as *similar*. Later, users may manually collapse similar nodes in the model graph if they indeed refer to the same concept.

The set of concepts M_n and qualified linkages M_e can help us to meaningfully index the content to drive applications such as semantic search. However, in order to create an E-R diagram and bootstrap a relational database, we need to do the following:
Differentiate between entities and attributes: As a simple rule, we may treat all concept nodes with only one incoming link to be *attributes* and other concepts to be *entities*. However, manual review and correction is necessary to such a distinction precisely.
Define cardinalities on relationships: Formatting cues can help us infer cardinalities to a large extent. For example, if a concept B is represented as a bulleted or numbered list, which is present in a section hosting concept A, then the link from A to B is said to have a cardinality of $1:n$ because many items of B may be contained in a single A. On the other hand, if B is captured as a paragraph inside section A, then $Cardinality(A \rightarrow B)$ will be $1:n$, unless we find multiple instances of such paragraphs inside the section. Similarly, a concept underlying a section has a cardinality 1:n with a concept emergent in a table contained in the section. Cardinalities of the nature n:n are only be defined when we aggregate across sub-models.

4.3 Aggregating Sub-Models

As a result of *Concept Learner*, one can learn the concepts in the aggregated (global) model across all the document types. However, since *Link Learner* runs separately on each document type, we learn sub-models corresponding to each document type. In order to learn the overall model, we need to aggregate links across the sub-models.

We make the assumption that concept names are unique and unambiguous in the domain of discourse. Hence, links across all the document types can be simply collected together to get the full inter-relationships for domain concepts. The global model can be cre-

ated by selecting concepts and links from different sub-routines, *Concept Learner, Link Learner, Concept-DocType-Refiner* or a combination, following different strategies depending on the level of human review available on the output. In experiment section (Figure 8), we show the relative statistics for learnt concepts and links. Some examples of the strategy are below. We used the conservative strategy in the results wherein only concepts that were common in the output of all sub-routines is output, and the links are only for these concepts as returned by *Link Learner*. In the moderate strategy, no concept from *Concept Learner* is dropped while in aggressive strategy, all links found *Concept Learner* are also returned.

Conservative Strategy:

$$M_{global} = \begin{array}{c}(M_n^{ConceptLearner} \bigcap (\bigcup_{t_i} M_n^{LinkLearner}), \\ \bigcup_{t_i} M_e^{LinkLearner})\end{array} \quad (1)$$

Moderate Strategy:

$$M_{global} = (M_n^{ConceptLearner}, \bigcup_{t_i} M_e^{LinkLearner}) \quad (2)$$

Aggressive Strategy:

$$M_{global} = \begin{array}{c}(M_n^{ConceptLearner}, \\ (M_e^{ConceptDocTypeRefiner} \bigcup (\bigcup_{t_i} M_e^{LinkLearner})))\end{array} \quad (3)$$

4.4 Nature of Manual Review

Algorithm *Learn-Model* has built in steps for manual review. We now comment on the type of judgment done at each step.

Step 2: One validates *control* metadata that is used to structure content regardless of content. They can be considered as document template concepts.

Step 5: One identifies concepts that need to be suppressed because either they are sensitive (e.g., for client documents) or frivolous (e.g., text used to format document).

Step 9: One decides whether to allow concepts to have self-loops or not; characterizes the type of links; resolves incorrect links which are possible if unique name assumption does not hold.

In addition, the *raised* and *reduced* threshold used in *Concept Learner* needs human judgment but it did not seem to have an impact across the datasets considered. Finally, the aggregation strategy is decided for step 8 but the consideration is the time experts can spend to review the output.

4.5 Representation of Output

As noted in Section 3, the output models M is a graph with nodes (M_n) representing concepts and edges (M_e) representing relationships between nodes. The edges can be undirected (e.g., in the case of co-occurrence) or directed (e.g., output of *Link-Learner*). We present M as a graph but can convert it into any representation depending on usage. In Section 6, a software modeling representation (EMF[2]) will be discussed.

5 Experiment

We conducted experiments to evaluate the quality of learned models, the practicality of the approach and application scenarios where the techniques could be relevant. In the following sub-sections, we discuss the data sets used, then report on the accuracy of the learn models, the practicality of the approach, and its suitability for scenarios like new model discovery and model evolution.

5.1 Data Sets and Experimental Settings

We had a rich data-set of documents created during blue-printing stage of SAP projects from multiple customers. Recall that there are at least 12 document types in this domain. The data-sets in the experiments are summarized below:

DS1: A collection of 96 Word documents made up of 55 PD and 41 BP documents. The files came from different client projects and had been manually cleansed to remove client- and project-specific, non-reusable, details.

DS2: A collection of 10 Word documents of type PD derived from a client project.

DS3: A collection of 98 Word documents randomly made up of types - TC(36), ID(15), TD(13), ED(2), DC(16), FS(1), CR(4), DE(4), EX(6), CO(1).

In addition, we had access to a SAP sub-model created by experts corresponding to the information captured in PD and BP document types. We will refer to it by M^*_{PD-BP} and a fragment of it is shown in Figure 1. Now, we can perform experiments in broadly two settings: one where model from expert(s), M^*_{PD-BP}, is already available and hence, the learned model can be compared with them to establish the former's quality, and the second where no model is available. In the second case, the learned models can only be reviewed by experts.

5.2 Accuracy of Learned Models

5.2.1 Comparing Learned Model with Known Model

To determine the accuracy of the learned models, we ran algorithm *Learn-Model* through *DS1* and *DS2*, and compared the output with M^*_{PD-BP}. Figures 6 and 7 show the learned models on *DS1* and *DS2*, respectively. We observe that the output models are a collection of sub-graphs containing directed edges and sometimes cycles.

In Figure 8 the statistics of key sub-routines of *Learn-Model* are shown. In the results for *Concept-Learner*, the links found by *Concept-Doctype-Refiner*

[2]http://www.eclipse.org/emf/

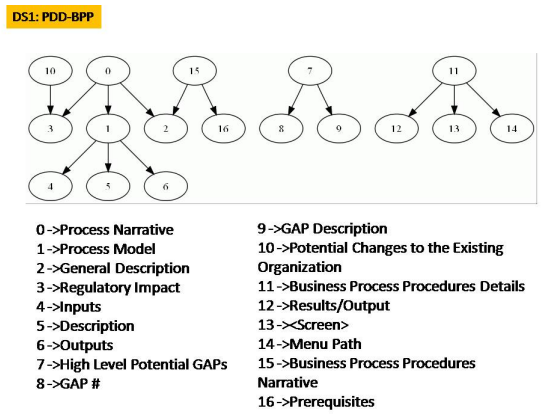

Figure 6: The model M_{DS1} learned for *DS1*.

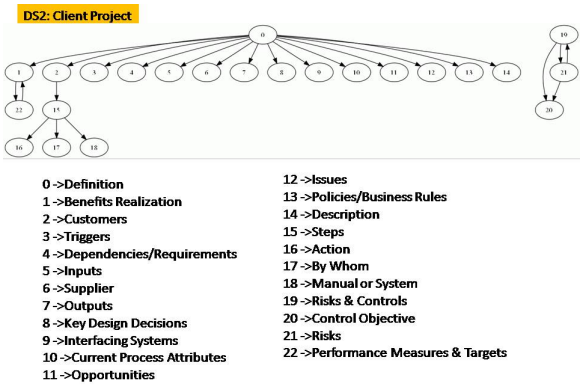

Figure 7: The model M_{DS2} learned for *DS2*.

are included; in *Link-Learner* results, the learned concepts are not required to be common with the candidate concepts, C (hence condition 4(b) is disabled); in *CL+LL* results, the candidate concepts are required to be in C (hence condition 4(b) of Figure 5 is enabled). We observe that the nodes and edges in the model learned by *CL+LL* are conservative and that is what we produce as the output of *Learn-Model*[3].

In Table 1, the learned models are compared with M^*_{PD-BP}. We find that 77% of learned concepts from *DS1* and 63% from *DS2* match those in M^*_{PD-BP}. The remainder of 23% concepts in *DS1* and 37% from *DS2* are new concepts that were not provided my experts in M^*_{PD-BP}, and should be included. Due to manual review steps in *Learn-Model*, no spurious concepts were present in the output. The output links (for matching concepts) were in agreement with those by experts.

5.2.2 Learning Unknown Model

We now consider the case of *DS3* where no reference model was available for comparison. Figure 9 shows the output of *Learn-Model* on *DS3*. Here, the output model is built by processing on 10 different document types (sub-models) and hence, the resulting graph gives deep insights about the information captured in the corresponding documents.

This model was shown to a subject matter expert to evaluate. It was found that the concepts are all meaningful while the relationships are valid.

[3]Another possibility is that along with output of *CL+LL*, concepts returned by *Concept-Learner* are returned so that the expert can use it as a reference (Model Strategy). These concepts had high support across the data-set but relationships were not found.

	Concept Learner		Link Learner		CL + LL	
	Concepts	Co-occurrence Links	Concepts	Links	Concepts	Links
DS1	30	870	88	102	17	14
DS2	43	1892	31	52	23	24
DS3	125	15494	381	555	42	39

Figure 8: Comparing Accuracy across Data-Sets. *CL+LL* represents the conservative strategy.

#Concepts	M^*_{PD-BP}	M_{DS1}	M_{DS2}
	46	30	43
#Matches (New Learned)	(M_{DS1}, M^*_{PD-BP})	(M_{DS2}, M^*_{PD-BP})	(M_{D21}, M_{DS1})
	23 (7)	27 (16)	7 (36)
%New	23%	37%	84%

Table 1: Comparing Different Models.

5.3 Practicality of Learning Models

We now check if the presented work can help users build model for a new domain efficiently when only documents are available. For reference, we knew that M^*_{PD-BP} took more than 2 weeks to build by interviewing subject matter experts, browsing through project documents and having different experts negotiate to build a consensus, and then many months to stabilize.

Table 2 reports the time taken to build the models with *Learn-Model*. We see that although there is wide diversity in the characteristics of the datasets (# documents types, # documents), the overall process is reasonably fast. The maximum time it took was less than 3 hours with 10 different document types (sub-models) and under 100 documents.

We also experimented with the number of documents needed to learn a good sub-model corresponding to a document type. We found that the number varies with dataset depending on how consistent the data in the documents are, but 5 is usually a good choice[4].

5.4 Discussion

Based on the experiments above, we found that the approach leads to learning of accurate models in reasonable time. This can be helpful in scenarios like *DS3* where there was no model in the beginning. The output, M_{DS3} can be presented to experts who can enhance it rather than start from scratch.

Another scenario where the approach will be helpful is in detecting the evolution of models. Over time,

[4]However, sometimes even 5 documents of a type may not be available as in *DS3*.

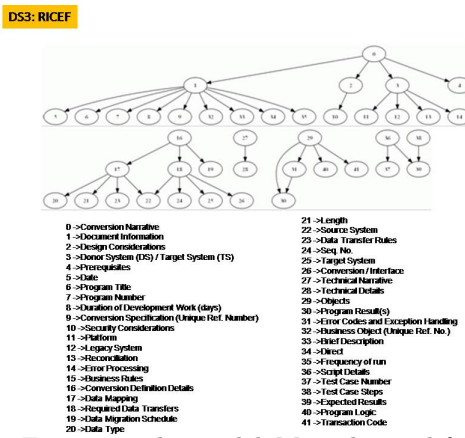

Figure 9: The model M_{DS3} learned for *DS3*.

Approach Step	DS1 Docs: 96	DS2 Docs: 10	DS3 Docs: 98
Step 1	48	6	49
Step 2	2	2	4
Step 3	1	1	2
Step 4	48	6	49
Step 5	2	1	10
Step 6	1	1	1
Step 7	24	3	25
Step 8	1	1	1
Step 9	2	1	10
Step 10	1	1	1
Total	130	23	152

Table 2: Time (in mins) Spent in Different Steps. Steps 1 and 4 take 30secs/doc while Step 8 takes 15 secs/ doc.

all models change. In the context of SAP projects, the information captured on a project changes with every new project. Service providers, who do numerous projects over time, want to detect the changes so that they can know changing industry trends and client preferences. In our setting, *DS2* is a client project which may capture different information (leading to new model) that what the service provider prescribes in *DS1*. The third column in Table 1 shows this. We find that in *DS2*, 84% more concepts are created over *DS1*. Any service provider, who only had M_{DS1}, would want to incorporate these changes.

6 Usage of Learned Model

We now show how the learned model can be used to generate software applications using MDA principles in services setting and the benefits. Often, information in enterprises get siloed in documents with unstructured formats such as MS Word and Excel. This leads to the disadvantage that once individual meaningful categories of information are collated into a single document, it is difficult to distinguish them individually at a later time. For example, if we consider a PD document, then it is difficult to identify automatically that it actually contains information on process description and process steps.

Our presented method helps extract the information model from a collection of enterprise documents. The information model identifies the key information concepts of a domain as inferred from the documents

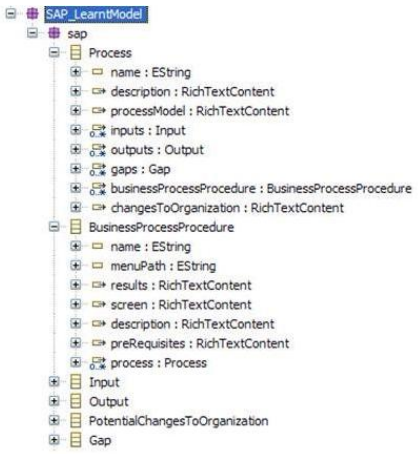

Figure 10: Learned model expressed in EMF.

and the relationship between different concepts. Existence of this information model is a pre-requisite to be able to capture and store structured data. We store the inferred information model as a EMF model, a popular modeling notation. Figure 10 shows an extract of the inferred model definition in EMF.

Using the EMF model, we automatically generate a user interface using which the users can enter information. For each concept, we generate an editor. Each editor has a list of sections corresponding to each concept it can be linked to. There is a single section for all simple properties of the concept. Rich text sections are used for those properties of the concept that allow for formatted text. Using the editor user can create a new concept. Using the individual sections, the concept instance can be linked to other concepts. Figure 11 shows an example of generated editor for BP. There is an "Attributes" section where information on *BP name* and *menuPath* can be entered. There are rich text sections for "Results/Outputs" and "Screen" as these are properties of BP (see inferred model in Figure 10) that allow formatted content. There is a section called "Process" that allows users to link instances to BP to *Process*. Once users have entered information in these editors, the information is already captured in a structured format. Additionally, we provide a utility that can pull content from this structured model and generate a corresponding document version for the same. Figure 11 shows the generated word-based BP document.

7 Conclusion and Future Work

We have presented an approach on learning submodels and the global model from different semi-structured document types in the enterprise domain. Our system performs well in terms of accuracy of extracting concepts, discovering new concepts not explicitly mentioned in the initial templates and in terms of performance time. While our tests are on SAP project

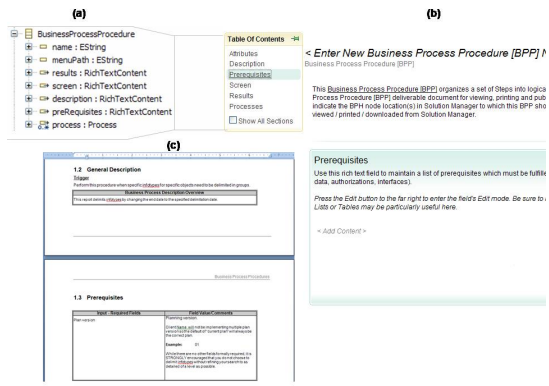

Figure 11: Editor generated from learned model in EMF.

documentation, all the steps are generic and would fit in the scope for any enterprise project documentation, where the documents are of a semi-structured nature and at least implicitly have an underlying model. We believe this is one of the first works in this area to tackle this problem with high potential impact.

One can extend the work along many lines. First, we made the unique name assumption for concept names and this needs to be relaxed. Second, a detailed study is needed on the relative trade-off between different sub-model aggregation strategies. Third, further investigation is needed on what type of links can be learned or distinguished. Fourth, the approach needs to be tested on datasets from other domains. Finally, it may be possible to reduce the manual review steps further.

References

[1] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa. Efficient substructure discovery from large semi-structured data. In R. L. Grossman, J. Han, V. Kumar, H. Mannila, R. Motwani, R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, editors, *SDM*. SIAM, 2002.

[2] J. Biskup and D. W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. In *Information Systems*, volume 28, pages 169–212, 2003.

[3] G. Cong, L. Yi, B. Liu, and K. Wang. Discovering frequent substructures from hierarchical semi-structured data. In *In Proc. of the 2nd SIAM International Conference on Data Mining (SDM)*, pages 175–192, 2002.

[4] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, 2001.

[5] H. Davalcu, S. Vadrevu, S.Nagarajan, and I. V. Ramakrishnan. Ontominer: bootstrapping and populating ontologies from domain-specific web sites. In *IEEE Intell. Sys., Vol: 18, Iss: 5, 24-33*, 2003.

[6] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, Y.-K. Ng, D. Quass, and R. D. Smith. A conceptual modeling approach to extracting data from the web. In *Proc. 17th Concept. Model.*, 1998.

[7] E. Hovy. Using an ontology to simplify data access. In *Communications of the ACM*, volume 46, pages 47–49, 2003.

[8] K. Kannan and B. Srivastava. Promoting reuse via extraction of domain concepts and service abstractions from design diagrams. In *Proc. SCC*, 2008.

[9] V. Kashyap and A. Sheth. Semantic heterogeneity in global information systems: The role of metadata, context and ontologies. In *Cooperative Information Systems: Current Trends and Directions*, 1996.

[10] J. Madhavan, P. Bernstein, K. Chen, A. Halevy, and P. Shenoy. Corpus-based schema matching. In *Workshop on Information Integration on the Web*, 2003.

[11] I. R. Mansuri and S. Sarawagi. Integrating unstructured data into relational databases. In *Proceedings of ICDE*, 2006.

[12] S. Melnik. Generic model management: Concepts and algorithms. In *Springer, ISBN: 3540219803*, 2004.

[13] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tag tree patterns in semistructured web documents. In *PAKDD '02: Proceedings of the 6th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 341–355, London, UK, 2002. Springer-Verlag.

[14] S. Nijssen and J. N. Kok. Efficient discovery of frequent unordered trees. In *Proceedings of the First International Workshop on Mining Graphs, Trees and Sequences (MGTS)*, 2003.

[15] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. In *The VLDB Journal*, 2001.

[16] S. Sarawagi. Information extaction. In *Foundations and Trends in Databases, Vol. 1, No. 3, Pg. 261 to 377*, 2007.

[17] S. Soderland. Learning information extraction rules for semi-structured and free text. In *Machine Learning Journal*, 1999.

[18] B. Srivastava and Y. Chang. Business insight from collection of unstructured formatted documents with ibm content harvester. In *ACM Intl. Conf. Mgmt. of Data (COMAD), Mysore, India*, 2009.

[19] V. C. Storey, R. Chiang, and G. L. Chen. Ontology creation: Extraction of domain knowledge from web documents. In *Conceptual Modeling - ER*, 2005.

[20] H. Vache, T. Vogele, , U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hubner. Ontology-based integration of information - a survey of existing approaches. 2001.

[21] K. Wang and H. Liu. Discovering structural association of semistructured data. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):353–371, 2000.

[22] M. J. Zaki. Efficiently mining frequent trees in a forest: Algorithms and applications. In *IEEE Transaction on Knowledge and Data Engineering*, pages 1021–1035, 2005.

[23] Y. Zhai and B. Liu. Structured data extraction from the web based on partial tree alignment. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1614–1628, 2006.