Science of Computer Programming

An industrial experience report on Model-based, AI-enabled Proposal development for an RFP/RFI

--Manuscript Draft--

Manuscript Number:	SCICO-D-23-00080R1
Article Type:	VSI:MDE Success Stories [MGE-Steffen Zschaler]
Keywords:	MDE; meta-model; document to models; model to text; Proposal; RFP; RFI; document parser; knowledge driven; AI; NLP
Manuscript Region of Origin:	INDIA
Abstract:	Large organizations respond to huge volumes of Request for Proposals (RFP)/ Request for Information (RFI) every year. The process of developing a proposal for an RFP/ RFI is completely manual and time-, effort-, and intellect-intensive. While Model Driven Engineering (MDE) approaches have been popular in downstream Software Development Lifecycle (SDLC) phases to transform the design models into code, there has been a gap in leveraging model-based techniques in document-centric phases of proposal development, requirements analysis, etc. This paper presents an automated proposal development approach for a client-supplied RFP/RFI using a combination of model-based and AI-enabled techniques and describes the case study of its successful deployment to hundreds of presales users across multiple geographies. We explain the Proposal system and report on the experience of deploying the system in the industry, bring out its efficacy, and user feedback, and discuss the lessons learnt.

Thanks for the review comments. They have helped in refinement of concepts and articulation.
Listed here are the resolution remarks for the review comments given by all reviewers.

Sr	Reviewer	Review Comment	Resolution remarks
1	Reviewer #1:	Section 1.2 One suggestion here is to use the same naming format for the boxes of the same type. For instance, some rounded squares use verbs (Load, Generate,) while others not (Leaning engine, Solution recommendation).	- Modified the figure 2
2	Reviewer #1:	Section 2 My first observation was that it is was difficult to map the elements in Fig 3 with those in Fig 2, which would be nice	- Modified the figure 2 and 3
3	Reviewer #1:	Section 2 I would suggest to explain what is a reflexive modeling language and which of its properties make it useful for this work. The reference provided Kulkarni et al., 2010; does not clarify this aspect.	- Added “Modeling framework enables the creation of purpose specific metamodel and models.”
4	Reviewer #1:	Section 2 The metamodel defined in Figure 4 is unclear to me. More precisely what is the meaning of the arrows. Sometimes the same type of arrow seems to be used as association, other times as composition or as inheritance.	- Updated Figure 4
5	Reviewer #1:	Section 2 Offering seem to play an important role in the entire approach and manuscript. But in Fig 4 seems that the Domain is the central aspect and it is composed of Concerns. This needs to be clearly explained. Also the word offering is associated in many places in the paper to different other concepts like: domain/offering, verticals/offerings, horizontal/vertical domains and offerings, which does not help with the clarification.	- Updated explanation of Metamodel.
6	Reviewer #1:	In Section 2.3, one important aspect is the patterns, but their exact link to the metamodel is not defined. A reference to the pattern mapping language and a short description is provide in the paragraph below figure 5 but it still does not clarify the mentioned aspect. The authors state "The Pattern Interpreter interprets the extraction syntax specified for classes, properties and associations, extracts model information from the documents, and populates intermediate representation of the model." but how are the extraction rules defined and linked to the metamodel?	- Updated section 2.3 with all the details.

7	Reviewer #1:	In Fig 5 is also not clear if the 5 activities in the Pattern Interpreter are executed sequentially top-down or not and how they are using the patterns. Last but not least how are they using the patterns, or which of them?	- Updated section 2.3 with all the details.
8	Reviewer #1:	BTW, ANTLR not explained.	- Updated section 2.3 with all the details.
9	Reviewer #1:	Page 7, and Fig 6, give a nice example, but it seems that the patterns are defined at model level not at metamodel level as suggested in Fig 5 and they are specific to each model. So this part will be a manual activity? In case I misunderstood maybe it could be better clarified in the text. Maybe the arrows between Extraction patterns and Document are misleading, since the flow for the DocToModel step should be from the Document to Model?	- Pattern has two part. Pattern tree + mapping specification with the documents. Pattern tree defines subset of metamodel and mapping specification is specific with document structure. Updated the paper for this clarify.
10	Reviewer #1:	Section 2.4, is K-Map referring to a well known recommender system algorithm? At least from row 56 on page 7 it seems so. In that case please provide a reference.	- Added reference to our previous paper
11	Reviewer #1:	Also the algorithm in Table 1 is a bit abstract. It would be better to be accompanied by an example.	- Example is already covered is case study section Fig. 10. Hence not covered here.
12	Reviewer #1:	Section 2.6 presents the learning engine. From the approach it is not clear what ML model type is used for training and classification	- Updated section with ML model type.
13	Reviewer #1:	One general observation is that in several cases short acronyms are used but they are not always defined (expanded at the first use). For instance: NL, NLP, AI, ML	- Fixed this. Expanded at the first use.
14	Reviewer #1:	I would use "over 700 BU users" instead of 700+	- Updated as suggested.
15	Reviewer #1:	- instead for small figures like 30+ I would use the exact number, which is 31? - same for 500+ proposals and 30+ offerings	- Updated as suggested.
16	Reviewer #1:	- Fig 8 is difficult to read due to low resolution	- Updated figure with higher resolution
17	Reviewer #1:	Fig 12 and its explanation confuses me. From the paper I understood that a document generated by the approach is called a proposal (which is not modeled in the meta-model). And a proposal contains information extracted from or related to different offerings (domains). From the explanation of fig 12 comes out that for each offering there are	- Proposal is modelled in meta-model (Fig 4) - Added explanation and examples of domain and offerings (section 3.1). - Modified Fig .9 illustrating the domain, offering, and KE in a better manner. - Added explanation in section 1.3 and Fig 10 about the offering and the specific proposal.

		multiple proposals generated. Maybe this can be better clarified.	
a18	Reviewer #1:	The evaluation in Figure 13 is limited and done for 10 users. It is not clear what type of users they are, if more then one belong to the same type... also an analysis and explanation on the large differences in improvements between users 2,3 and 4,6,8 and 1,5,9. The average time for the automated approach seems somewhere at 18 hours?	- Evaluation section has been significantly expanded with details
19	Reviewer #1:	Last but not least, no evaluation of the machine learning approach is provided wrt to its performance. The initial dataset was manually created. How much data was needed before the classifier can do accurate selections?	- Fig.9 shows the volume of manually created knowledge data.
20	Reviewer #1:	Finally, some evaluation results are provided in the conclusions instead of the evaluation section.	- Removed from conclusion.

1	Reviewer #2:	However, I do not feel that in its current for the paper presents a good example application of MDE. In particular, a lot of the paper focuses on aspects of the approach that are not relevant to the field of MDE.	- Updated discussion, updated figures. - MDE is central to the proposal system.
2	Reviewer #2:	Section 2 outlines the approach, and mentions a lot of technologies, but I feel should be much more focused on emphasising the MDE aspects.	- Figures are updated
3	Reviewer #2:	For example by describing what MDE toolsets have been employed - rather than explaining the algorithms for e.g. k-map recommendations. This is similar throughout the reflection section, which reflects on why the approach was successful, but again not on the use of MDE and the benefits and challenges it brought. Also, I feel that currently the paper is ot written as a review and reflection, but more a statement of new work.I feel it could be reworked a little to improve this.	- Added separate paragraph in discussion section. Figures are also modified with show models in a core part of the approach.
4	Reviewer #2:	The description paragraph for Fig 3 contains a lot of technologies that are not illustrated in the picture. I would suggest either describing the elements themselves within the text and/or highlight the technologies in the figure. It appears that many of these technologies are programming libraries rather than MDE frameworks/tools and here I would say it would be good to emphasise the MDE tools further, for example explaining the features that were used.	- Updated figure 3
5	Reviewer #2:	The description paragraphs for Fig 4 seems not to match the figure. In particular a domain has offerings relating to domain in the figure, but the text talks	- Updated figure 4.

		about offerings consisting of a set of concerns. These paragraphs need a bit of a clearer discussion. This seems to be the core element of MDE presented in the paper, but the model and discussion do not align. I am also not convinced by the model itself in its current form. Here why not present a more standardised UML class diagram with specific association types etc and multiplicities.	
6	Reviewer #2:	In section 2.6 is not clear to me, what is the learning used for, it seems to be to improve the knowledge base, but how is this done in relation to the type of rfx? Is it doing some sort of semantic based matching of questions when new data is added? does MDE help with the learning?	- Learning engine is not model based. But helps in augmenting the models.
7	Reviewer #2:	Figures 9 and 12 really do not add much, they are two generic for any insights to be drawn as it is not clear what offerings are. If this is sensitive then I am not sure how this could be fixed, but a discussion of example offerings would at least improve things. Perhaps also a discussion on the nature of the key offerings. Similarly, figure 13 does not offer many insights. Why only present across 10 users if hundreds have been involved? There is also a large disparity in the time taken across the manual efforts needed to develop proposals and hence it is hard to understand the gains. Here I would present the average time for a user perhaps? Finally the details of the user study setup are completely missing.	- Included section 3.1 explaining on domain and offerings. - Modified Fig 8 knowledge model illustrating example offering Service desk. Significantly updated evaluation section.
8	Reviewer #2:	Reflection section discuss challenges, but not necessarily how MDE helped overcome these challenges. It would be nice to see why MDE helped in particular. For example, the discussion on Docx4j is more around producing documents in java, rather than anything MDE related.	- Added paragraph is discussion section
9	Reviewer #2:	I would like to see the related work paragraph on the MDE approaches expanded to compare and contrast similarities to the presented work. At the moment it only states that "other work exists".	- Related work updated
10	Reviewer #2:	The work makes in several places that the approach is generic, for example "A generic proposal system meta-model that is domain and offering agnostic", but from the presentation it is not clear why this is the case, as it appears that a single case study with a single organisation has been undertaken. Here I think the case study is fine, but I would not make such bold claims in general.	- Removed generic from multiple places. The approach is domain-agnostic and can be applied for proposal automation in any domain.
11	Reviewer #2	Finally, I have also attached a PDF with notes on typos, suggested changes etc.	- Fixed typos

1	Reviewer #3:	the paper is written in such a way that it is hard to follow especially for someone who is not	- Added section 3.1 explaining domain and offerings

		familiar with the domain of RFx. The problem is exacerbated by the high-level description of the different components of the system. the evaluation of the work is not systematic.	- Added figure 10 illustrating a sample RFP. - Section on evaluation has been significantly updated with details.
2	Reviewer #3:	Some more information on RFx on how they look like, and how you can extract semantic information from them would have improved the readability of the paper.	- Included Figure 10 that illustrates a sample RFP and questions asked. - Table 2 shows a semantic evaluation of the information in two example questions. -
3	Reviewer #3:	Does the NLP engine interact with the modelling framework or with the documents in natural language? What are the semantics of the different arrows?	- Updated Figure 3
4	Reviewer #3:	The authors mention that they are using ML for the creation of models. However, there is limited information on this aspect.	- Added some text in learning and discussion section
5	Reviewer #3:	An instance of the proposal system metamodel would have been useful to help the reader understand how such models look like.	- Fig 8 shows an instance of MM. Updated this figure for better resolution
6	Reviewer #3:	Since the Doc2Model paper has not been published yet, the authors should have provided more information on this component to help the reader assess their solution.	- Updated the section 2.3 with details - Paper is now published and available
7	Reviewer #3:	- It is not clear what the fuzzy matching rules mentioned in the algorithm (Table 1) are.	- Updated text
8	Reviewer #3:	Figure 8 is not readable	- Updated
9	Reviewer #3:	Why are the proposed evaluation metrics of interest? In general the evaluation is not presented in a systematic manner and it does not provide information on the evaluation set up, research questions, etc.	- Evaluation section has been significantly updated. .
10	Reviewer #3:	In the evaluation a quote from one of the users says "The RFx technology automates approximately 70% - 75% " Which parts of the RFx development are not automated? What is the quality of the generated solutions?	- Evaluation section has been significantly updated and it includes metrics on %automation. -
11	Reviewer #3:	The paper has a number of typos. Below I write some indicative examples of typos but there are many more in the paper:	- Fixed

An industrial experience report on Model-based, AI-enabled Proposal development for an RFP/RFI

Padmalata Nistala
TCS Research
Tata Consultancy Services, India
nistala.padma@tcs.com

Asha Rajbhoj
TCS Research
Tata Consultancy Services, India
asha.rajbhoj@tcs.com

Vinay Kulkarni
TCS Research
Tata Consultancy Services, India
vinay.vkulkarni@tcs.com

Sapphire Noronha
Tata Consultancy Services, India
sapphire.noronha@tcs.com

Ankit Joshi
Tata Consultancy Services, India
ankit.joshi@tcs.com

Abstract

Large organizations respond to huge volumes of Request for Proposals (RFP)/ Request for Information (RFI) every year. The process of developing a proposal for an RFP/ RFI is completely manual and time-, effort-, and intellect-intensive. While Model Driven Engineering (MDE) approaches have been popular in downstream Software Development Lifecycle (SDLC) phases to transform the design models into code, there has been a gap in leveraging model-based techniques in document-centric phases of proposal development, requirements analysis, etc. This paper presents an automated proposal development approach for a client-supplied RFP/RFI using a combination of model-based and AI-enabled techniques and describes the case study of its successful deployment to hundreds of presales users across multiple geographies. We explain the Proposal system and report on the experience of deploying the system in the industry, bring out its efficacy, and user feedback, and discuss the lessons learnt.

Keywords: MDE, meta-model, document to models, model to text, Proposal, RFP, RFI, document parser, knowledge driven, AI, NLP

1. Introduction

1.1. Background

Business organizations and governments (customers) seeking products, solutions, or services publish Request for Proposal (RFP) or Request for Information (RFI). RFP/RFIs are project announcements posted publicly by customers inviting bids from suppliers. An RFP is a "collection of formal documents that includes a description of the desired form of response from a potential supplier, the relevant statement of work for the supplier, and required provisions in the supplier agreement" (ISO/IEC/IEEE 24765: 2017). An RFI typically precedes an RFP and seeks information about a product, service, or supplier capability. In this paper, the term "RFx" refers to RFI/ RFI/ any other form of bid. Each RFx typically comprises a set of questions on various capabilities that need to be answered, and the response submitted is a "proposal document".

In large IT organizations, each business unit (BU) responds to hundreds of customer RFx yearly. An RFx is best viewed as a set of questions in a specific context. Though BUs have a dedicated presales/ bid team responsible for responding to RFP/RFIs; the team is always overloaded. An RFx has approximately 30 to 50 questions, and interpreting all these questions and developing a proposal by referring to various available sources of information is **a time-consuming manual activity**. Fig. 1 depicts a typical view of some of the challenges faced by the bid teams working on proposal development.

There is a huge dependency on various subject matter experts (SMEs) of presales teams to develop quality proposals within the **RFx deadlines**. SMEs analyze the questions to identify the domain/ subject areas of interest, the type of information asked, whether it is related to company facts, a strategy or a methodology adopted, or prior experience in that domain/ offering. Information is scattered across multiple documents; hence **Manual search** is carried out by searching for specific documents in local/ shared repositories, searching for keywords, selecting relevant documents, and marking relevant portions of reusable text. Any information from existing documents/ past proposals is suitably **customized** for the specific RFx being worked. Often, there is **no single source of truth** and a lack of periodic knowledge validation resulting in inconsistent information and facts presented to customers. Most SMEs prefer to use data from their hard disks resulting in outdated/inaccurate information. **Resource constraints** in handling high RFP volumes result in experts being assigned multiple bids. As RFx questions generally seek information on capabilities across domains and the data lies with different groups/ experts, **numerous emails and follow-ups** are usually necessary to obtain the relevant answers for the questions. The complexity of multi-tower bids results in huge bid teams created for each pursuit and mammoth coordination of efforts. Organized best practices/knowledge sharing across offerings is absent. Moreover, the proposal must consider customer-specific contextual parameters such as the geography of the business, product or service offering, and so on. For example, while positioning past experience in the domain, **case studies in the same geography** are usually searched and positioned. Finally, the gathered information for each RFx question has to be composed into a "proposal document" by copy-pasting the relevant information as per a meaningful compositional order, customized, and **manually formatted** as per a standard/ desired document template. Every stage in this process is manual and people-dependent, and MS Office is usually the only software used.

Fig. 1 A Day in the life of Bid Team

1.2. Overall Approach

Companies bid for multiple RFx and a lot of standardized content is reused across the proposals. It is observed that 50- 70% of any proposal has standardized content taken from existing documents. These documents are maintained in local repositories in word/ excel formats. Analyzing the information sources and solution content used in creating proposals by BUs, it is identified that technology capability for structuring knowledge content and linking knowledge to RFx questions can enable automation support to develop a better and faster "Proposal document". A lot of time and effort that is spent in followups/ emails to concerned SMEs, manual activities of search and document formatting can be saved through a Proposal system that can transform existing knowledge into machine-processable models and further leverage those models and harmonize with AI techniques to generate proposal documents automatically.

Modeling has been a proven enabler for productivity in code-centric phases to generate code from design models. AI in SDLC is an active research area (Giudice, 2016; Xie, 2018). AI and knowledge-aware software have remained largely untouched by MDE, and it has the potential to drastically change the way MDE is used and perceived in the software community (Cabot et al., 2017). Here we propose an approach for automating the proposal development by exploring model-based techniques in combination with AI techniques. Fig. 2 depicts the overall approach.

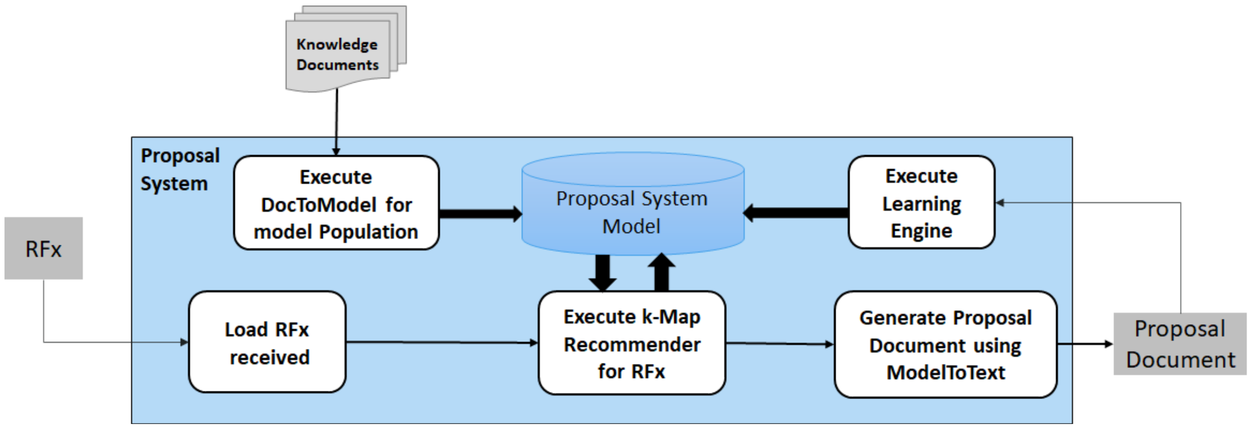


Fig. 2 Overall Approach

This is a model-based AI intervention to generate a proposal in the desired format for a client-provided RFP/RFI. At the heart of the system is the "Proposal System Model" that enables a super-structure to be imposed on the information of interest lying around in a fragmented form as NL -text. This purpose-specific meta-model serves as the lens to extract information from the NL documents and facilitate *document-to-model transformation* in which the NL documents are parsed, and knowledge information is extracted into a machine-processable proposal knowledge model. Once a new client-provided RFP/RFI document is loaded, instantaneously on-click, leveraging the proposal knowledge model populated, *solution recommendation* can be generated for all RFx questions that have some relevant information in the knowledge model. As the complete information is in the model form, using model-to-text transformation, the proposal document can be automatically generated based on choices on recommendations. Learning Engines harvests new content from the proposals and updates the knowledge model. The system thus can become self-learning and continue to evolve and become more efficient. It enables faster turnaround time for proposal generation and quick and easy access to accurate information anytime/anywhere.

The key contributions are:

- A domain-agnostic purpose-specific meta-model for the Proposal system
- Use of a document-to-models transformation for automatic parsing of knowledge documents into machine-processable models
- Model based solution recommender for providing knowledge-map recommendations for RFx questions
- Model-to-text transformations for the automatic generation of Proposal document for a given RFx

The whole approach is model-based and suitably augmented with Natural Language Processing (NLP) and Machine Learning (ML) techniques to handle the problems in the manual, document-centric proposal development. The paper is organized as follows: section 2 describes the proposal system architecture and key components, section 3 discusses the case study results and evaluation, section 4 discusses the lessons learnt at various stages in the development, section 5 briefly discusses the related work, and section 6 concludes the work and way forward.

2. Proposal System: Model-based AI-enabled solution architecture

2.1. System Architecture

The overall approach outlined has been implemented into a digitalized Proposal system by harmonizing multiple realization technologies. Fig. 3 depicts the high-level architecture of the Proposal system. The system takes as input knowledge documents in natural language, customer supplied RFx, and context and automatically generates a proposal document as the response to the RFx.

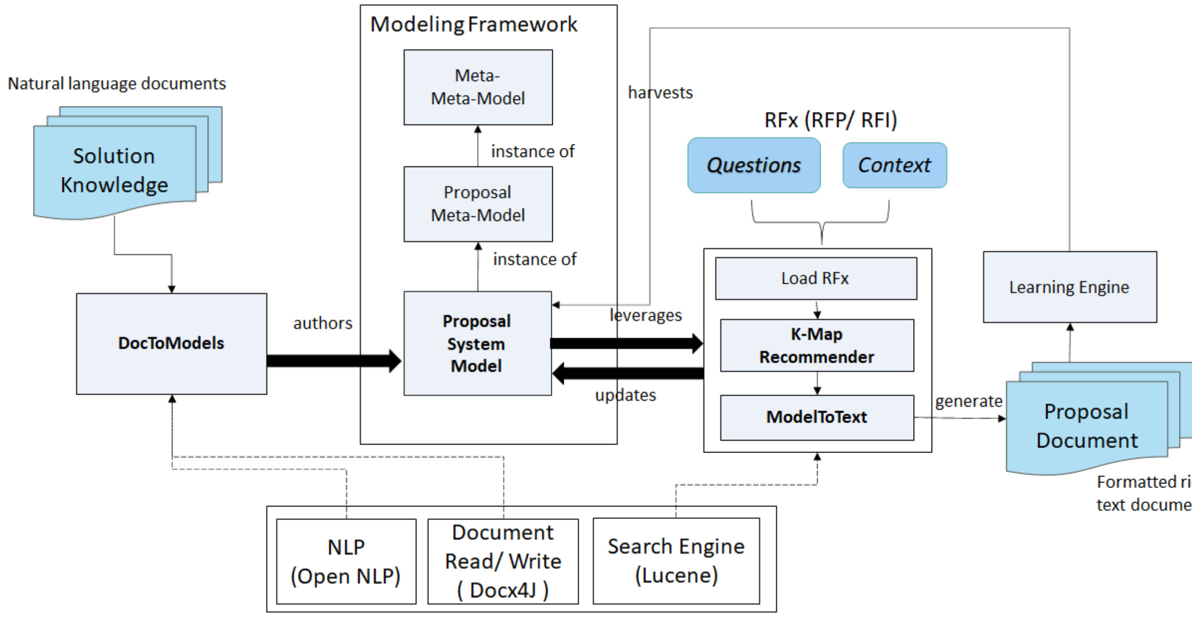


Fig. 3 Proposal System Architecture

The key components of the system are

- Proposal system model: A domain-agnostic meta-model for proposal development for any client-provided RFP/RFI is defined. This meta-model is the heart of the overall system and facilitates knowledge capture and solution generation for any RFP/RFI across domains and offerings.
- DocToModels: DocToModels engine for the transformation of natural language documents into machine-processable model form is proposed and leveraged. DocToModel is meta-model and document-type agnostic and can extract information from any type of document into an instance model of any purpose-specific meta-model. Here, it is used to populate knowledge data into the Proposal model.
- K-Map recommender: For a client-provided RFP/ RFI, the questions can be analyzed, and the instance model can be queried to provide relevant knowledge as *K-Map* recommendations.
- ModelToText: The K-Map is composed into a formatted rich text document using model to text technology to generate a Proposal document for the RFx.

The system architecture is built using a unique harmonization of multiple technologies. Modelling is at the core of this architecture, and every component is built around the models. A modelling framework based on a reflexive modelling language compatible with OMG Meta Object Facility (MOF) is used to define models at various levels (Kulkarni et al., 2010; MOF, 2023). Modeling framework enables the creation of purpose specific metamodels and models. A model at each level is an instance of the model at the previous level. As shown in Fig. 3, A meta-model (Class, Property and Association) is an instance of the meta-meta-model (MObj, MProp, MAssoc). Meta-model consists of a model schema (Object, Slot, Link). The Modelling framework provides a user interface and model APIs to define purpose-specific meta-models and instance models. Information extraction from NL documents usesDocToModel. OpenNLP is used for text processing techniques such as tokenization, lemmatization, stop words removal and matching functionality (OpenNLP, 2023). Docx4j is an open-source Java library for creating and manipulating Microsoft Open XML files (Docx4j, 2023). It is leveraged to compose retrieved solution knowledge into a proposal document. Lucene-based **search engine** facilitates knowledge search (Lucene, 2023). Index-based searching provides faster traversal and retrieval of the model. Indexing and searching using Lucene facilitated a custom query language with a simple query syntax using AND/OR joined Boolean expressions. The Learning engine harvests knowledge from submitted proposals and feeds back into the proposal system model.

2.2. Proposal System Meta-Model

The meta-model in Fig. 4 represents the integrated system meta-model for proposal development leveraging the domain/ offering knowledge. This system meta-model is the central component of the Proposal system and models all the relevant elements and their associations for developing the proposal for a client-provided RFx. This Proposal_MM can be seen as a combination of proposal knowledge and solution model. A reflexive modeling language compatible with OMG MOF has been used to define this meta-model (Kulkarni et al., 2010).

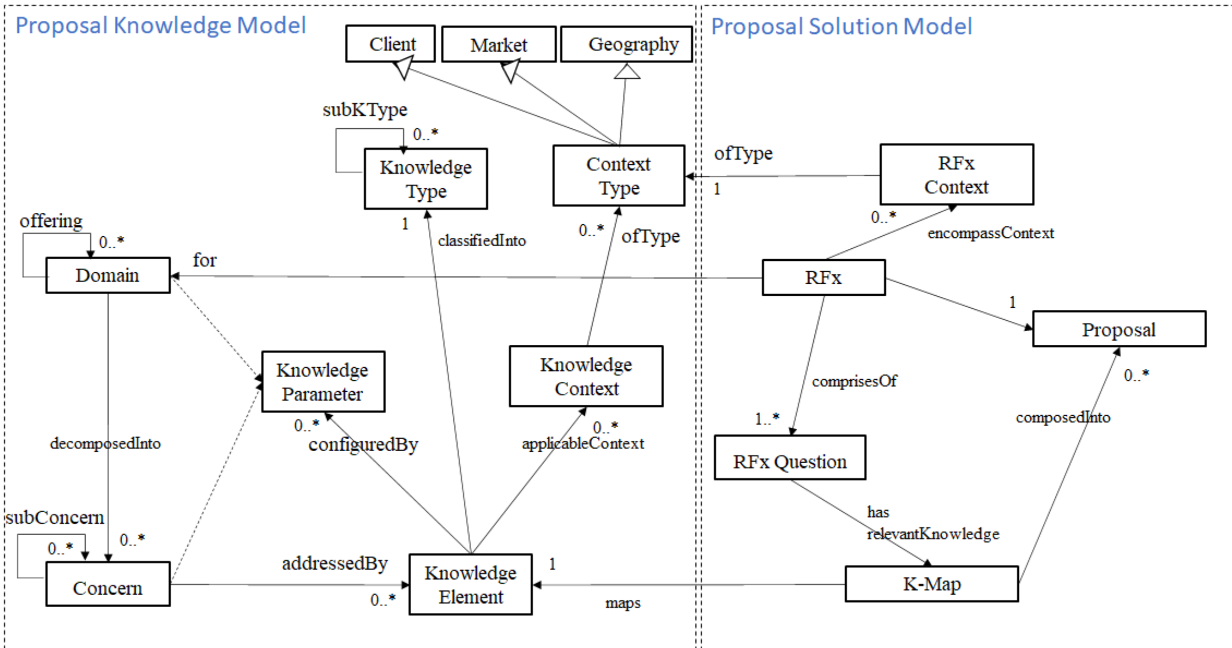


Fig. 4 Proposal System Meta-Model

The **proposal knowledge model** abstracts the solution knowledge relevant to a domain/offering in a model form. A pattern approach has been adopted to structure the knowledge model in terms of problem concern, associated context, and knowledge element. For each RFx received from a client, the proposal solution meta-model is instantiated with data specific to the RFx to be responded to. All Proposal system components populate into/ extract information from this model.

The meta-model elements are described here. Any business operates in specific *Domains*. A business *Domain* can have many domain-specific *offerings*. A Domain-specific *offering* comprises a set of *concerns* that are of interest to the business. In the IT product or services context, each market area for which solution knowledge exists becomes a domain: banking, insurance, financial services, business process services, and so on. In each of these *domains*, *one or more offerings* may be offered. A *concern* uniquely defines a subject area of interest and belongs to a domain/ offering. A *concern* may further be decomposed into *subConcerns*.

KnowledgeElement is the granular piece of solution information belonging to a subject *Domain*, addresses a specific *concern*, and is applicable in a specific *KnowledgeContext*. A *KnowledgeElement* may embed multiple *KnowledgeParameters* data such as domain dictionary, offering parameters, etc. *KnowledgeElement* is classified into *KnowledgeType and subKtype*. A *KnowledgeType* can have multiple *KTypes*. For the proposal process, solution knowledge for an offering can be classified into *KnowledgeTypes* such as *product knowledge*, *process knowledge*, *facts knowledge*, *case knowledge* and so on. Within a *KnowledgeType*, there can be further *subKTypes*. Product knowledge may further be classified into *strategy*, *model*, *technology* etc. Each *KnowledgeElement* is relevant in a specific context that can be expressed in terms of *ContextType*. *Geography*, *Customer*, and Operational *Market* are some important *ContextTypes* to be considered. A lot of information related to specific offerings and context can be parameterized as part of knowledge configuration and embedded into *KnowledgeElements*. *KnowledgeParameters.* A *KnowledgeElement* is configured by a set of *KnowledgeParameters*. These *KnowledgeParameters* can be configured and refined as required for *Domain/Offerings* and *Concern/SubConcerns*. This proposal knowledge meta-model can be instantiated for various domains, offerings, and concerns with corresponding *KnowledgeElements*.

The **proposal solution model** abstracts critical concepts of a client-provided RFP/RFI, i.e., the RFx and the proposal created for an RFx. An RFx is floated for a desired capability in a Domain/Offering and is best viewed as a set of questions housed in a specific context. Each *RFx* is *for* a *Domain/Offering* and comprises a set of *RFxQuestions.* An *RFx encompasses* a certain client context *RFxContext* that is of *ContextType*, such as *Geography*, *Customer*, and so on. Each RFx Question in the specified *RFxContext* can have relevant knowledge in terms of a set of *KnowledgeElements* that can be mapped as *KMap*. The *K-Map* for each distinct *RFxQuestion* in the *RFx* is composed into the *Proposal*. This proposal solution meta-model is instantiated for various client-provided RFP/RFIs and the corresponding *Proposals* prepared and submitted.

2.3. DocToModels

Large volumes of solution documents with proposal-related knowledge are maintained as knowledge repositories. Microsoft Word and Excel are the most common document formats for maintaining these documents. To digitalize the information in the large volumes of NL documents containing solution knowledge into machine-processable models, DocToModels is proposed in our earlier work (Rajbhoj et al., 2023). We leverage that work for the Proposal system. DocToModels proposes an automated approach for generating models from natural language documents. Given a purpose-specific meta-model and a set of NL knowledge documents, the DocToModels can parse the word/excel documents and populate the proposal knowledge model described in section 3.1. It leverages the structural containments and consistency in the documents. The high-level view of the DocToModels is depicted in Fig. 5. The complete pattern mapping language syntax and implementation details of the DocToModels approach are described in our earlier work (Rajbhoj et al., 2023).

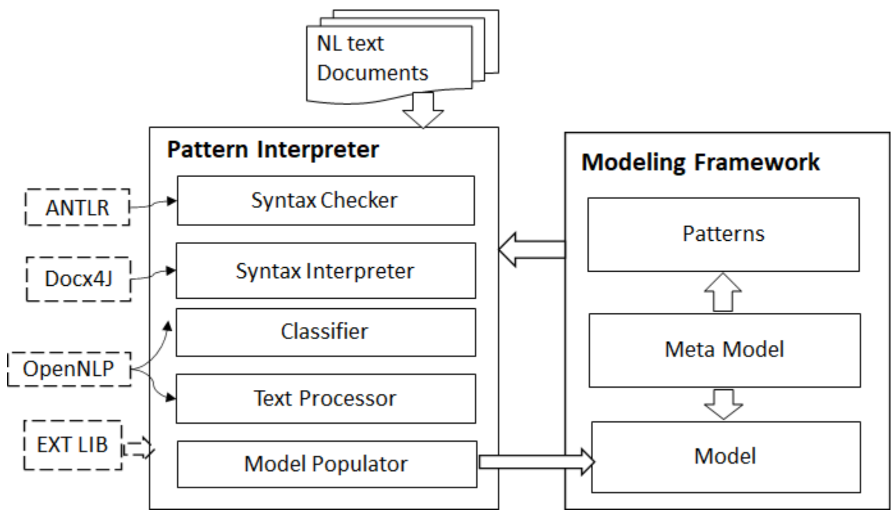


Fig. 5 DocToModels - Overall Approach [26]

DocToModels provides an extraction pattern syntax to specify the mapping of the document content to the purpose-specific metamodel. The **pattern mapping language syntax** comprises statement syntax for various document structural elements: heading statements, style statements, table statements, etc. A statement is related to value extraction from the documents. The extraction pattern syntax provides the essential string processing library functions. An External library with external functions can be used for complex processing logic that cannot be supported using pattern language. OpenNLP parser (Open NLP, 2023) is used for Natural Language Processing (NLP) of text. NLP **is** augmented with a configurable *Domain dictionary* to handle the taxonomical variations of domain terms. Docx4j (Docx4j, 2023) is for extracting styled content from documents as specified in the extraction pattern. *Pattern Interpreter* is implemented using ANTLR (ANTLR, 2023), Docx4J, and OpenNLP. ANTLR (ANother Tool for Language Recognition) is widely used to build languages and tools. Docx4J is used for extracting styled content from documents and OpenNLP is used for the Natural Language Processing of text. Pattern Interpreter has a Syntax Checker, Syntax Interpreter, Classifier, Text Processor, and Model Populator. Syntax Checker validates the mapping syntax specified for all pattern elements of input Patterns for any grammatical errors. Syntax Interpreter interprets the mapping syntax specified for Pattern Model elements. Classifier, Text Processor functions can be used in mapping syntax for processing the extracted text. Classifier uses NLP techniques such as stemming and lemmatization for text preprocessing. It helps in classifying the text into predefined class labels. Multiple text processing functions are provided in Text Processor for string operations and pattern checking for handling unstructured text processing. The ***Pattern Interpreter*** interprets the extraction syntax specified for classes, properties, and associations, extracts model information from the documents, and populates an intermediate representation of the model. On interpreting all input extraction pattern files against all the input documents, IR is navigated, and the purpose-specific model is populated using **Modeling framework** APIs.

Typically, a set of documents belong to a document type and have a standard document structure in terms of organization of headings, styles, formatting, and so on. Hence, the extraction ***Patterns*** that are input to the DocToModel are specific to a document type and are defined as a pattern model in the modeling framework. Patterns helps is specifying subset of Metamodel of interest and mapping of each metamodel element with document structural elements. In pattern mapping, the extraction pattern syntax is specified manually for each meta-model class/property/association. Using the meta-model as a lens, the document structural elements such as headings, paras, bullets, tables, etc. are analysed as per pattern mapping and corresponding model elements are automatically instantiated.

Fig. 6 presents a representative view of the extraction patterns defined for extracting information from solution documents and the mapping of document content to the proposal knowledge meta-model using the pattern syntax. Pattern mapping is prepared based on the document structure and information organization of the document template used for capturing the solution knowledge. It can be seen from the meta-model that there is a self-association *subConcern* for the *Concern* Class. To specify the levels of such associations, class names is post-fixed with annotation. As can be seen in Fig. 6, the two entries are specified as *Concern* and *ConcernSub*. Every Heading1 has information on domain and concern and hence is mapped to the MM classes *Domain* and *Concern*. The heading structure Heading2 indicates *Concern* hierarchy, and the association *subConcern* is mapped. Heading 3 indicates *KnowledgeElement* and mapped. Each Heading3 is a *question* followed by an answer to that question. Answer maps to *summaryArtifact* property of *KnowledgeElement*. This content may have images, styled text, and tables. The complete answer text is extracted and stored as a word document in a binary form in the instance model. *KnowledgeElement* name is also used to classify *KnowledgeType* class and *KnowledgeContext* class. The library function CLASSIFY uses the input domain dictionary data to classify the *KnowledgeType*. Extracted *KnowledgeElement* summary may have *KnowlegeParameter* references. These *KnowlegeParameters* are extracted through an external function getKParameter by passing *KnowledgeElement summary* as a parameter. DocToModel extracts the relevant information from input NL documents using the defined patterns and populates the proposal knowledge model.

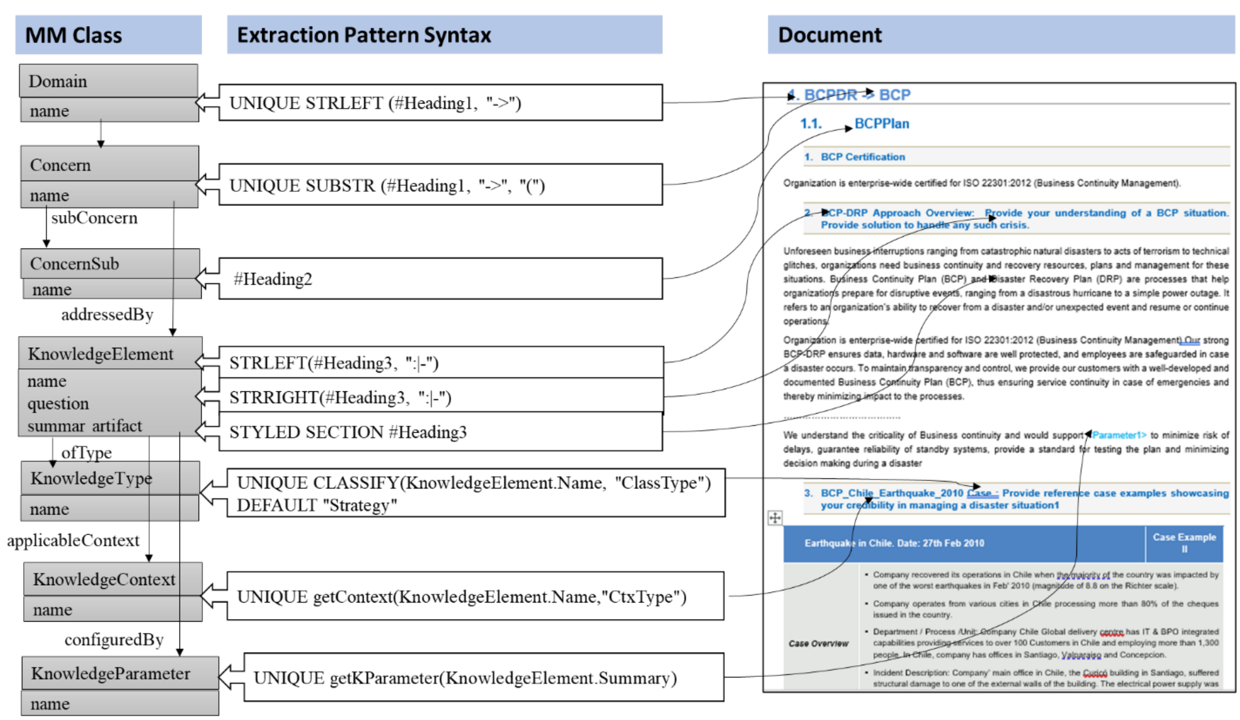


Fig. 6 Extraction Patterns Mapping illustration

2.4. K-Map Recommender

The RFP/ RFI (RFx) document provided by the client is received as input. A typical RFx document contains a list of RFx Questions in natural language text form. The context of the RFx in terms of the client, geography, the offering etc., is captured. Each Question in the RFx is analyzed and transformed into a query to retrieve relevant knowledge elements (KEs) as a K-Map for each question. The K-Map Matching algorithm, as outlined in Table 1, is used to automate this task (Rajbhoj et al., 2019). This algorithm leverages NLP techniques, applies configuration data, and implements fuzzy matching rules. Fuzzy matching considers various types of matching rules i) extract matching, ii) inexact matching ignoring the order of words, iii) inexact matching considering one less word for strings lengths greater than 3.

Question analysis and classification is the first step. Each RFx question text in natural language is pre-processed, and key meta-data is derived - subject area(s) of question, i.e., *Concern*, and type of information requested, i.e., *KnowledgeType*. Further, the Question is classified into Standard/ Offering/ Project. The relevant *KnowledgeElements* are retrieved based on the matched *Concerns* and *KnowledgeTypes.* The *RFx context* is used for filtering the *KnowledgeElements* as applicable. For example, case studies are selected using *geography*. *Offering/ Concern* specific configuration data defined in *KnowledgeParameters* are replaced in the content of *KnowledgeElements.* Further, project-specific solution parameters are captured and used. 5W1H with What, Who, When, Where, What, Why, and How dimensions are commonly used to capture the solution view (Ikeda et al., 1998). The Matching knowledge elements are listed as **K-Map** for each RFx Question. Subject Matter Expert (SME) can view the K-Map recommendation options, select relevant KEs, and update/ change the composition order of the options.

The illustration of K-Map recommendation is shown in the case study section.

Table 1 K-Map Matching Algorithm

KMatching Algorithm: To provide solution recommendations for RFx
Input**:** Client supplied RFx, Context For each RFxQuestion ∈ RFx Begin Step1: Question Analysis to identify applicableConcern and KnowledgeTypes - Parse *RFxQuestion* text using the NLP parser to get POS tags. - Perform stemming to remove stop words and lemmatization to get words in root form. - Retrieve relevant POS-tagged words. (Starting with NN, VB, JJ, RB). - Identify matched *Concerns* using fuzzy matching rules. - Remove tagged words with successful matching - Identify matched *RFxKnowledgeTypes* using fuzzy matching rules Step2: Compose Query - Compose query using identified concerns and types using the following rules. a. If extraction finds only *Concern* and not *RFxKnowledgeType*, then use default *Knowledge Type* for forming query. Default *RFxKnowledgeTypes* are configurable b. If query extraction retrieves *Case* as an *RFxKnowledgeType*, then add *KnowledgeContext* information to the query using AND expression. Step3: Retrieve K-Map for RFx - Retrieve the *KnowledgeElements* matching the composed query - Retrieve the configuration data *KnowledgeParameters* linked to the query - Extract the Knowledge elements matching the query as K-Map for each RFx Question. End

2.5. ModelToText

ModelToText generates a proposal document as per desired document format using the reviewed K-Map. The K-Map recommendations are composed as per compositional rules using model-to-text transformation.

- **Template Selection:** The Proposal needs to be prepared as per a standard document template or a customer-specific document template. It may be MS Word/ Excel/ PPT. The desired document template with suitable styles and formats is selected.
- **Question- Answer Writing:** Questions from the input RFx are selected as per the sequential order in the RFx. Selected K-Map recommendations for each question are composed of a single answer. In the case of multiple *KnowledgeElement* selections, the *KnowledgeElement* text is placed in the answer as per the compositional order of *KnowledgeType*. The Question and the composed Answer are added to the proposal document. For questions without any recommendation, an empty answer placeholder is populated.
- **Parameter replacement:** A *KnowledegElement* text can contain various placeholders such as context parameters, offering parameters, solution parameters, etc. The values of the parameters are replaced in the answer as per the SME selection.

2.6. Learning Engine

The Proposal generation system depends largely on the proposal knowledge model. Initial knowledge documents are manually created by collating information from diverse sources. It is required to perform incremental knowledge updates in a periodic manner. A learning-based approach depicted in Fig. 7 is proposed to mine the relevant knowledge from submitted proposal documents.

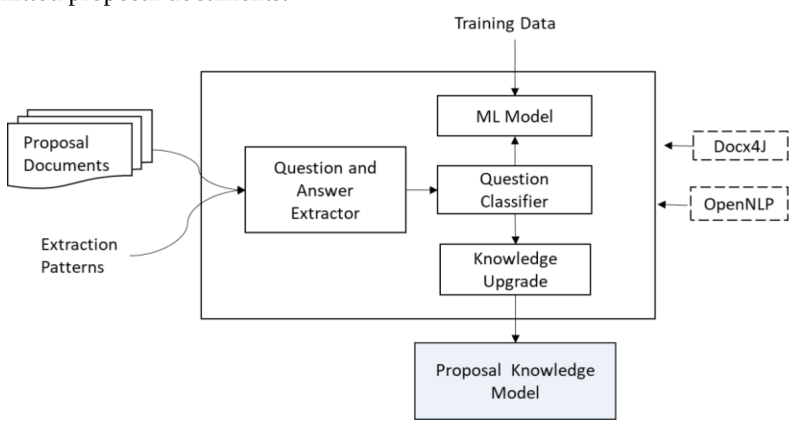


Fig. 7 Learning approach

As proposal documents may have different document structures, a configurable extraction pattern-based approach is used for programmatically extracting *question text* and *answer text*. As per a standard pattern configuration, the *question* is extracted from the document heading and style information. The text after the question until the next question or the end of the document is considered an *answer*. Docx4J is used for reading the document content.ML model is trained using training data. The corpus of training data is prepared by retrieving the questions and corresponding concerns from the model data of the Proposal system. For training maximum entropy (maxent) classifier is used as our features are words and the sequence of words matters. The Maxent algorithm assumes the conditional dependence of the features. The extracted proposal questions are then classified by *Question Classifier* into *concerns*, and concern-specific knowledge data is generated. Automated checks are carried out on whether the question matches the existing *Knowledge Element* question. If so, the answer is checked for similarity, and the answer is marked for update. If it is a new question, then a new *KnowledgeElement* is proposed to be added with association to the appropriate *concern*. Human is in the loop to review/ refine these knowledge upgrade recommendations. This reviewed data is used for incremental *knowledge upgrades* to the *proposal knowledge model*.

3. Case Study

The Proposal system has been deployed as a case study with the sales and pre-sales teams (proposal writing experts) of a large-scale Business Unit (BU) that offers a suite of IT-enabled services to multiple customers across various geographies. Thousands of Requests for Proposal (RFP) / Requests for Information (RFI) floated by customers are responded to every year by that BU.

The team faced multiple challenges in proposal development due to very high people dependency and manual, document-centric process. The solution information with capabilities, tools, technologies for various domains, and solution offerings was maintained in MS Word documents. Since this information is spread across multiple documents, content duplication, and information consistency issues are common. It was a heavily expert-dependent process to select the solution fragments from documents relevant to a specific question in the RFx. Follow-ups are carried out with SMEs for the latest and up-to-date information related to a specific domain/ capability/ offering as required for a question. Manual search, copy-paste of content from individual documents, and formatting for appropriate styles was time-consuming and effort-intensive activity.

3.1. Domain and Offerings

The Proposal system has been deployed and rolled out to over **700 BU users** from sales/ pre-sales teams. There multiple domains in which business operates such as banking, retail, life sciences, IT services and so on. In each of these domains, there are multiple solution offerings. And RFPs are floated by customers for a specific offering or a combination of offerings. Some examples of the domain/ offerings are given here.

- IT Infrastructure Services (IT-IS) domain has various offerings such as IT Service Desk, Workplace Management Services, Data Management services, Application Management Services, and so on.
- Life science domain has various offerings such as Clinical data management, Medical writing and publishing, Regulatory affairs, Pharmacovigilance, Biostatistics and programming and so on.
- Telecom domain has various offerings such as Core telecom back-office processes, Contact center operations, Point solutions for telecom industry, and so on

Customers float RFPs requesting a set of services for one or more offerings in a domain. For example, in IT infrastructure domain, RFPs may request for proposals for IT service desk offering or Work place management services. Proposals are prepared for the specific offerings leveraging the existing solution knowledge for these offering.

3.2. Knowledge Model

The Proposal system has been deployed and rolled out to over **700 BU users** from sales/ pre-sales teams. Solution documents related to the capabilities of **31 offerings** were collated and parsed using the DocToModels. The DocToModels extracted proposal knowledge model information in terms of domain, concerns, knowledge elements (KE), and corresponding knowledge parameters (KParameters) and knowledge context and populated the model.

Knowledge views are developed for navigation of the *Knowledge Model* as depicted in Fig8. This view can be seen at a domain level or a concern/sub-concern level and presents the solution knowledge content as per the knowledge model: *Knowledge-Type* (labelled as Class and K-Type), *Knowledge Element*, *Knowledge-Context* (label K-Context). Fig. 8 illustrates the knowledge model for the domain "IT IS (Infrastructure services)" and offering "Service Desk"

The knowledge model view (Fig. 8) has three windows.

1. **Offering view** (LHS view): *Domain*, *Offering* and *Concern* navigation is on the LHS view. A specific domain/ offering can be selected to view corresponding *KnowledgeElements*. The knowledge files listing for **IT IS domain** and **Service Desk offering** can be seen in the View.
2. **Knowledge Element View** (Middle View): Knowledge elements corresponding to the selected domain/ concerns are displayed along with K-Type and Context in the middle view. A specific *KE* can be clicked to see its content in the Document view. *KnowledgeElements* for "Service Desk" offering are listed.
3. **Document View (RHS):** Provides a view of the document content of the selected KE. A view of the **Service Desk Overview** document is shown.

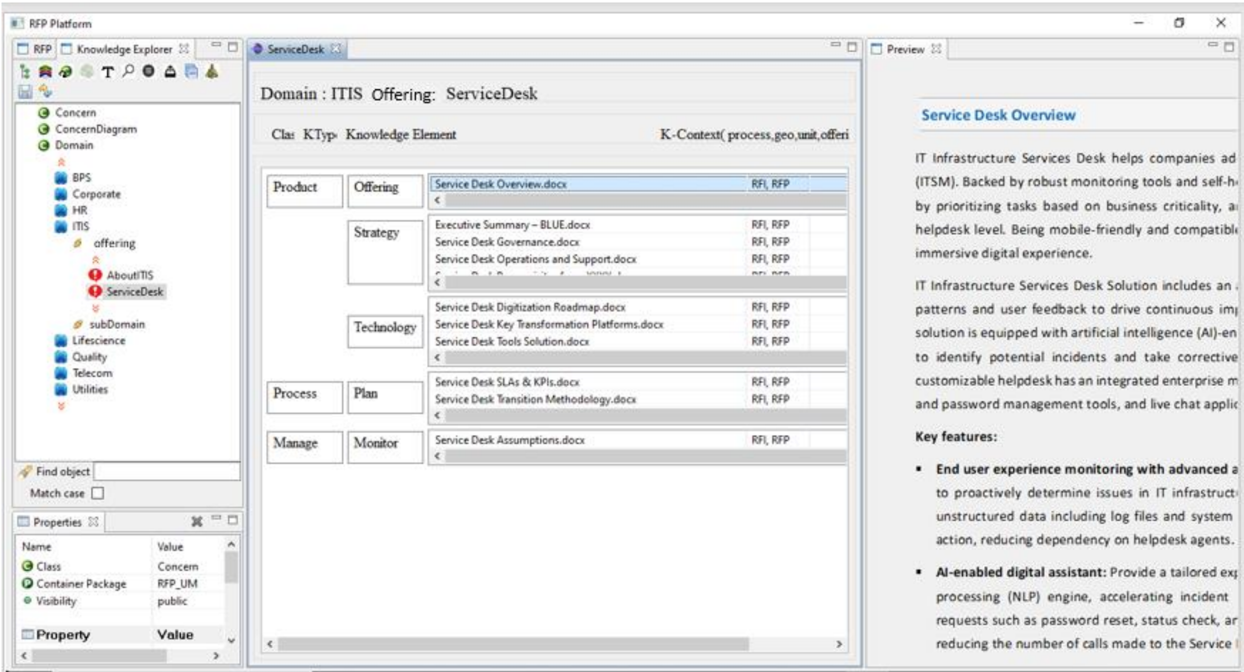


Fig. 8 Knowledge Model View

Fig. 9 presents a view of the knowledge model data – offering-wise count of knowledge elements and parameters. On average, 30-60 KEs and 10-30 parameters are created for an offering. The depth and granularity of knowledge can be seen in the graph in Figure 5.

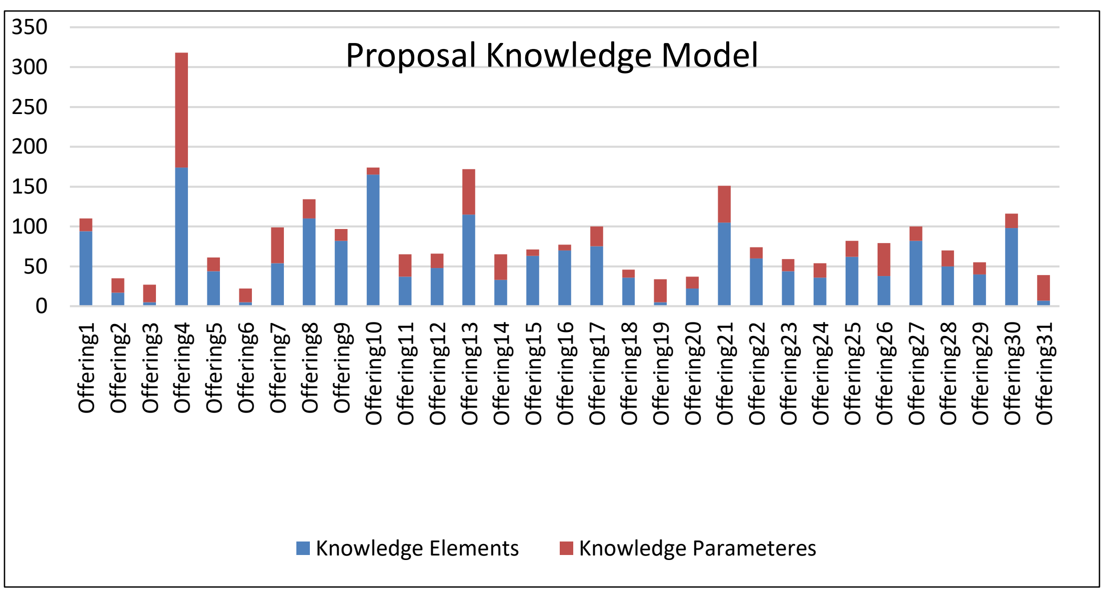


Fig. 9 Case Study - Knowledge Summary

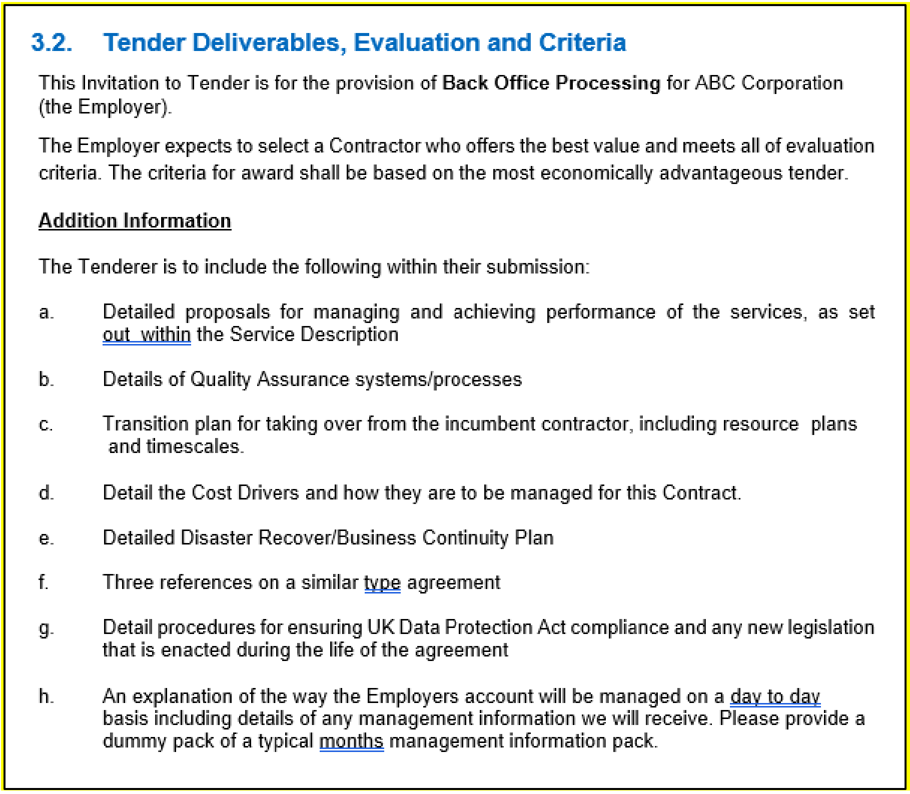
Key metrics *Mean Knowledge Element Density* and Mean *Parameter Density* are computed as follows.

$$Mean\ KnowledgeElement\ Density = \frac{\sum(KE\ Count)}{Offering\ Count} = 60.5$$

$$Mean\ KParameter\ Density\ = \frac{\sum(KParameter\ Count)}{Offering\ Count} = 26.2$$

3.3. RFP

Customers float RFPs requesting a set of services for one or more offerings in a domain. Fig 10 shows a view of the RFP floated by ABC corporation for offering “Back-office processing”. A sample list of questions asked in the tender are given in Fig. 10.



3.2. Tender Deliverables, Evaluation and Criteria

This Invitation to Tender is for the provision of **Back Office Processing** for ABC Corporation (the Employer).

The Employer expects to select a Contractor who offers the best value and meets all of evaluation criteria. The criteria for award shall be based on the most economically advantageous tender.

Addition Information

The Tenderer is to include the following within their submission:

a. Detailed proposals for managing and achieving performance of the services, as set out within the Service Description

b. Details of Quality Assurance systems/processes

c. Transition plan for taking over from the incumbent contractor, including resource plans and timescales.

d. Detail the Cost Drivers and how they are to be managed for this Contract.

e. Detailed Disaster Recover/Business Continuity Plan

f. Three references on a similar type agreement

g. Detail procedures for ensuring UK Data Protection Act compliance and any new legislation that is enacted during the life of the agreement

h. An explanation of the way the Employers account will be managed on a day to day basis including details of any management information we will receive. Please provide a dummy pack of a typical months management information pack.

Fig. 10 A Sample RFP

3.4. Proposal Generation

Fig. 11 depicts a view of the proposal generation for an RFP received from Client ABC Corp. Limited based in UK geography, in the Utilities domain and for the offering Meter to Cash (M2C). Each Question in the RFP is analyzed and transformed into a query to retrieve relevant proposal Knowledge Elements for that question as a K-Map. As shown in Fig. 12, the proposal document is automatically generated using the reviewed K-Map recommendations.

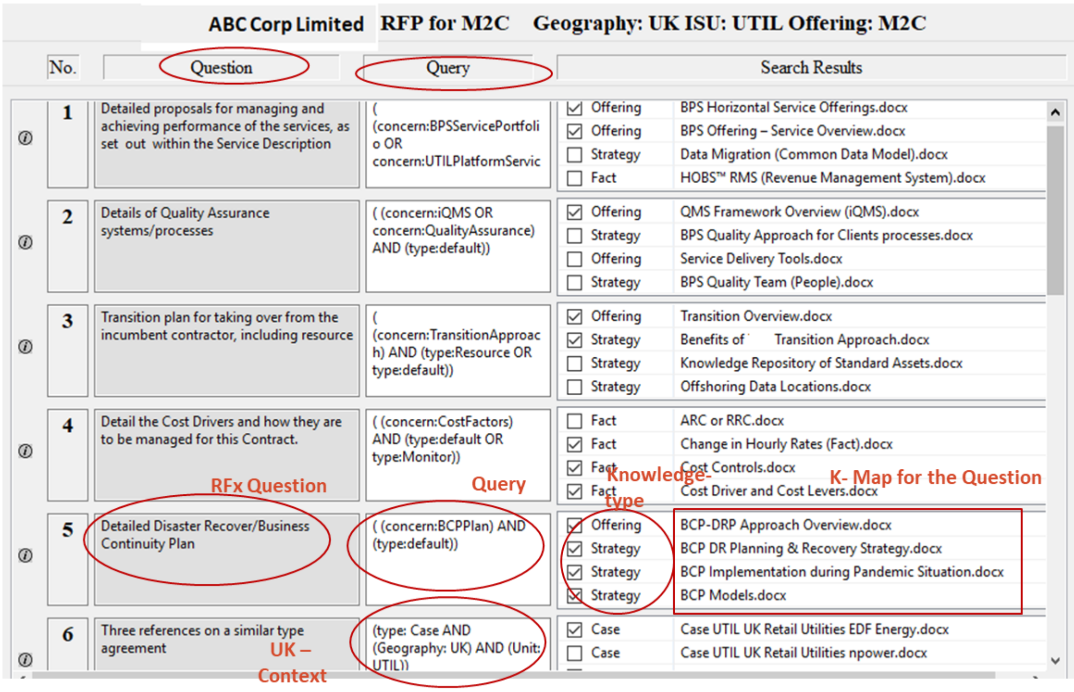


Fig. 11 K-Map Recommendation View

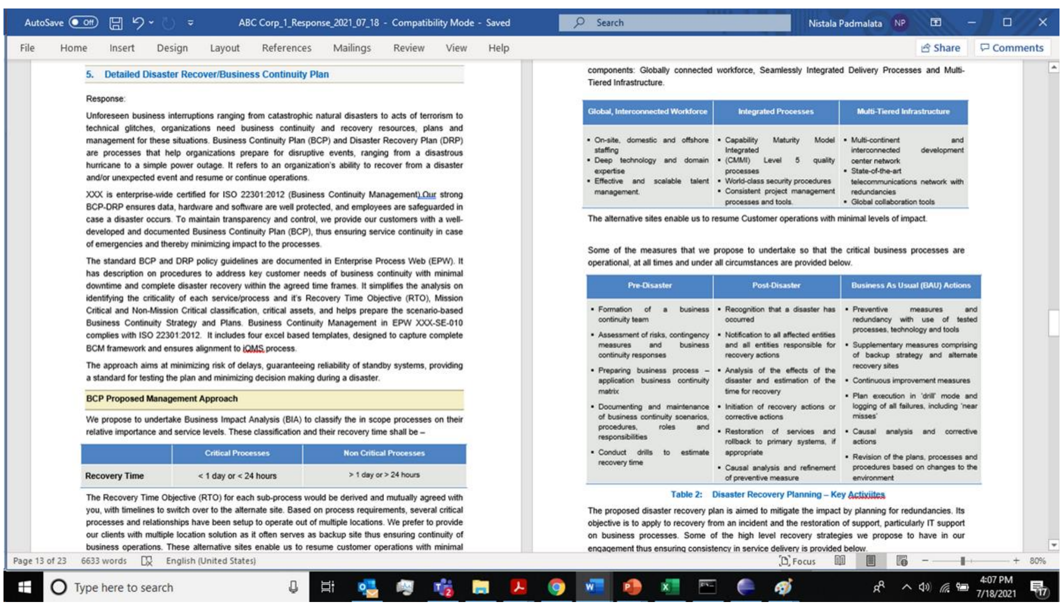


Fig. 12 Generated Proposal Document

3.5. Evaluation

Hundreds of users from pre-sales/sales teams across all geographies have started using the Proposal system. Users belong to various roles of solution architect, bid manager, solution head, etc. Hundreds of proposals have been generated across seven geographies using the Proposal system. For evaluaton purposes, 15 RFP/ RFIs spanning seven business units, six geographies, and nine service offerings. The RFx were selected from the wider set of submitted RFx responses of recent years covering common types of RFx – RFI and RFP, multiple industrial domains - government, hitech, retail, telecom, utilities, life science, manufacturing, multiple geographies – US, Europe, APAC, ANZ, India. Together, these 15 cases represent a wide range of RFx scenarios across domains and geographies.

For each RFx, the client-provided RFP/RFI questions and context parameters are taken into the excel file formats. These input problem and context files are loaded using the technology built. RFx and RFx Context are populated, and on search on any RFx, solution recommendations are generated question-wise. This RFx question-wise response solution recommendation is exported to a data collection excel sheet for evaluation purposes. For each question, the system-generated query is validated and based on a comparison of the submitted RFx response concerns and the generated query, relevant concerns, retrieved concerns, and retrieved and relevant (R&R) concerns are computed.

i. Retrieved concerns: Number of concerns retrieved for the question by the generated query

ii. Relevant concerns: No. of relevant concerns for the question as from submitted RFx Response.

iii. R&R concerns: Retrieved and Relevant concerns.

Two key metrics are computed for evaluation: *RFx_automation* and *RFx_accuracy*

i. RFx Automation: is determined based on the number of RFx questions for which answer is generated. Computed as per the formulae given below:

$$RFx\_automation = \frac{\sum(Query\ R\&R)}{No\ of\ RFx\ Questions}$$

ii. RFx Accuracy: Precision, Recall, and F1 measure are computed as per the formulae given below. F1 measure is taken as the accuracy metric. Precision is the ratio of R&R query concerns to Retrieved concerns, and Recall is the ratio of R&R query concerns to relevant concerns for the question. F1 measure is a standard formula as given below.

$$Query\ Precision = \frac{\sum(Query\ R\&R)}{Retrieved\ Questions} \qquad Query\ Recall = \frac{\sum(Query\ R\&R)}{Relevant\ Questions}$$

$$RFx\_accuracy = \frac{2 * Precision * Recall}{(Precision + Recall)}$$

For RFX in the sample set, question-wise analysis data is summarized, and metrics are computed as per the formulae stated. Table 2, Table 3 and Table 4 summarize the metric computation and values for the selected set of RFx.

Table 2: Question Evaluation – examples

Example	**Example1**
RFx	RFx-1
Question	Provide a detailed Disaster Recover/Business Continuity Plan
Extracted Query	(concern:BCPPlan)
Retrieved Concerns	1
Relevant Concerns	1
R&R Concerns	1
Remarks	KMatching algorithm picked up 1 concern based on question analysis which is highly relevant. Query Precision is 100%, and Recall is 100% for this scenario.

Example	**Example2**
RFx	RFx-1
Question	Are employees required to undergo information security awareness training upon hire?
Extracted Query	(Concern: Training)
Retrieved Concerns	1 (Concern: Training)
Relevant Concerns	1 (Concern: Information Security Awareness)
R&R Concerns	0
Remarks	The question is related to a specific Concern on Information Security Training, and the query retrieved a generic Training Concern. There is a KE gap related to "information security awareness training." Both Query precision and Query recall are zero.

Table 3: Computation of Evaluation Metrics

Metric	**Data Item**	**Metric Value**
RFx		RFx_1
RFx automation	Total questions	17
	Questions for which Query is retrieved	14
	RFx_automation metric	**82.35%**
RFx accuracy	sum Retrieved concerns	27
	sum Relevant concerns	23
	sum R&R concerns	22
	Query Precision	81.5%
	Query Recall	95.7%
	RFx accuracy (F1 measure)	**88.0%**

Table 4: RFx Evaluation Metric Summary

Sr.	RFP / RFI	Unit	Geography	Service Offering	Total Questions	**RFx Automation**	**RFx Accuracy**
RFx-1	RFP	Utilities	UK	Offering1	17	82.4%	88.0%
RFx-2	RFP	Manufacturing	US	Offering2	20	75.0%	77.6%
RFx-3	RFP	Lifescience	Europe	Offering3	34	67.6%	42.9%
RFx-4	RFI	Lifescience	Europe	Offering3	17	94.1%	76.7%
RFx-5	RFI	Lifescience	US	Offering4	30	70.0%	65.6%
RFx-6	RFP	Telecom	ANZ	Offering5	38	65.8%	82.6%
RFx-7	RFP	Hitech	APAC	Offering2	70	77.1%	74.2%
RFx-8	RFI	Retail	ANZ	Offering6	12	91.7%	77.8%
RFx-9	RFI	Lifescience	US	Offering3	22	95.5%	62.2%
RFx-10	RFI	Government	India	Offering7	30	70.0%	83.3%
RFx-11	RFP	Utilities	US	Offering8	60	41.7%	64.6%
RFx-12	RFP	Utilities	UK	Offering5	15	60.0%	77.4%
RFx-13	RFP	Utilities	US	Offering9	31	64.5%	75.5%
RFx-14	RFP	Utilities	US	Offering5	38	55.3%	56.8%
RFx-15	RFI	Retail	US	Offering2	22	95.5%	60.2%
				Mean		***73.7%***	***71.0%***

It can be seen from Table 11 that *RFx_automation* ranges from 41% to 94%, with the mean being 73%. The validation data shows on an average for any RFx received from a client, RFx can automatically generate a response document with standard responses for 73%+ questions. And the accuracy of this generated response is 71%.

Further, feedback has been sought from the users. Quantitative data on time taken for proposal generation before/ after using the Proposal system has been gathered, and the graph is plotted. As seen in Fig. 13, there is a clear and distinct productivity improvement in proposal generation due to the Proposal system.

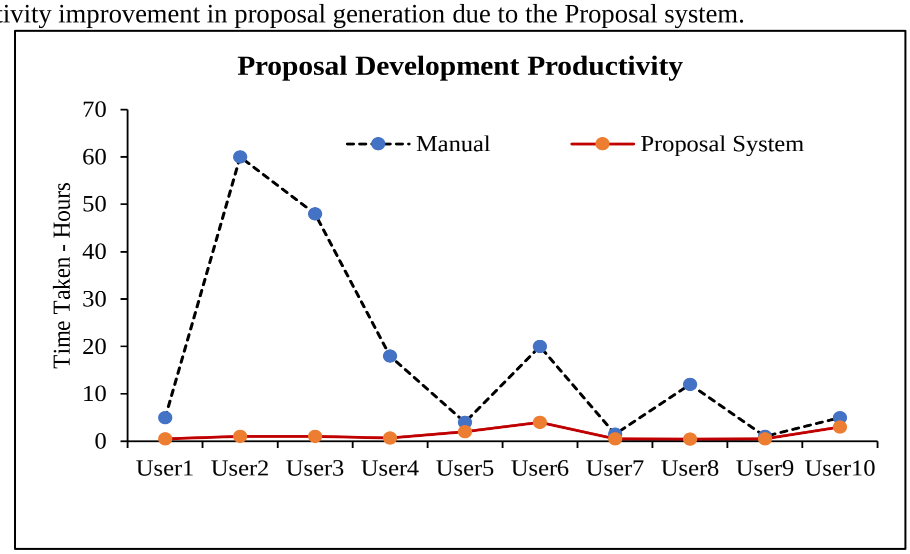


Fig 13 Proposal Development - with/ without Proposal System

The qualitative feedback from the stakeholder on this technology is highly positive, and sample User comments are given here.

"The RFx technology automates approximately 70% - 75% effort in preparation of the RFP/RI responses. Some of the key areas are: 1. Bid authoring: Automates the process of searching for responses across multiple boilerplates. 2. Self-service: Anyone can do it, anytime and anywhere. 3. Reduces human touchpoints: the same person can manage the entire process from upload of the RFx questionnaire to getting a formatted response document. All this allows subject matter experts to focus on higher-value activities. It would ensure a superior quality of response in terms of information and its articulation. This will, in turn, will result in man-hour saves and more business for the company."

"The earlier manual approach required 24-48 hours to turn around an RFP. With the RFx technology, this can be accomplished in 2-3 hours."

"The RFx technology is domain-agnostic and can be extended to other business domains and units working on RFP/RFIs as the process of RFX response is similar across ISUs. Across the organization, thousands of RFX are responded to in a year. There is scope for multi-fold productivity benefits and effort savings."

4. Discussion and Lessons Learnt

Given the current industry practices of heavily manual and expert-driven proposal development for RFP/ RFIs, a model-based and AI-enabled approach is proposed to help the pre-sales teams in automatically generating a standard proposal document and provide a kick-start to the proposal process. A key contribution of the work is a core Proposal system meta-model that abstracts the critical elements from diverse knowledge documents belonging to various domains and offerings and the key constructs of a proposal document. An AI-enabled technology has been implemented around a core modelling framework to automate the proposal development: populating knowledge model from Natural Language (NL) documents, receiving RFP/RFI, question analysis, information search, and finally, composition of a proposal document. This capability to generate a "proposal document" is a big value-add to the presales teams in the industry. There were many technical and operational challenges, and lessons learnt at every stage of the Proposal system development and implementation.

The Model-Driven Engineering (MDE) based approach offered a plethora of advantages for the proposal system, significantly improving its development and maintenance process. By adopting an MDE approach, we were able to abstract the key aspects of the proposal system, including user requirements, proposal data, and the evaluation process. This separation of concerns made the system design more modular, easier to comprehend, and facilitated smoother maintenance. Updating the proposal system to accommodate changes in knowledge and requirements was seamless with MDE. We could easily modify the models to reflect new information. By leveraging existing modeling infrastructure for populating, querying we could rapidly create the proposal system. Non-functional concerns, such as performance, scalability, and multi-user support, well-supported by the underlying modeling infrastructure. This relieved us from having to tackle these concerns explicitly, enabling a more focused development effort.

The initial hurdle was in understanding the diverse knowledge document formats received from presales teams. A series of interactions with the presales teams revealed that though the structure of information in a document varies, the documents contain similar concepts – knowledge content related to a domain/ offering. The understanding led to the definition of the *proposal system meta-model* covering the key concepts – *domain*, *offering*, *concern*, *knowledge element*, and so on. *Knowledge types* are defined to classify the information and knowledge context to link up the specific context in which the knowledge is relevant. This *meta-model* is domain-agnostic and can be used to capture knowledge content related to any domains and offerings. The *meta-model* enables a super-structure to be imposed on the information of interest lying around in fragmented form as NL text and is the central building block in our architecture to which every other component feeds/ retrieves model data.

The next key challenge was handling the diversity in document structures and formats while parsing information from NL documents. The structure of the document provided insights into the information organization and could be mapped to the meta-model concepts. We started with writing a purpose-specific document parser leveraging the document structure such as headings, tables, bullets, and so on and defining extraction patterns for mapping information in the document to the meta-model. This purpose-specific document parser imposes some constraints on pre-processing of knowledge documents. Handling the extraction pattern variations through purpose-specific information extractors is not a scalable approach. Hence, we developed *DocToModels*, a configurable extraction pattern-based model extractor, that can be configured for a specific meta-model and a specific document structure (Rajbhoj et al., 2023). *DocToModels* approach has been implemented to extract information from diverse NL documents and populate the proposal knowledge model. This approach is both document structure agnostic and meta-model agnostic, thus paving the way forward to extend this technology for other business use cases that are manual and document-centric such as requirements. While MDE technology for code has been in place for many years, this research has provided a key MDE enabler for handling documents and bringing documents into the models world.

For *K-Map recommendation*, question analysis is carried out to determine the appropriate *knowledge elements*. Question text is generally written in a free form with taxonomical variations. Standard Natural Language Processing (NLP) wordnet dictionary was insufficient to handle the domain-specific words (Miller, 1995). Augmenting the NLP with the domain dictionary helped in handling this. For question analysis, various matching heuristic such as exact matching, inexact matching, and fuzzy matching was implemented. Conflict resolution was necessary when the NLP analysis of a specific question returns multiple *knowledge types*. It was necessary to define rules on the order of priority for *knowledge type* assignment in question analysis. Further, the specific business context of the RFP in terms of *context types*, such as geography, market, etc., has to be considered while recommending solution options as a *K-Map*. For example, for an RFP from a UK Client, to respond to a question on providing references on past experience, the knowledge elements retrieved need to be past case studies from UK geography. The *K-Map* options are filtered considering the *knowledge context*. There is scope for improving these recommendations through learning engine, which is part of the roadmap.

Reading and writing documents could effectively be performed through Docx4J (Docx4j, 2023). A document can have a variety of content in the form of embedded attachments, tables, images, and all these variations are handled in a suitable manner using Docx4J technology. *ModelToText* transformation to generate a formatted *proposal document* required styled document composition which was a difficult task. Remapping of relationship was carried out for complex objects such as images and attachments. The weaving of answers from knowledge element fragments necessitated the formation of composition rules based on *knowledge types* to harmonize the content. Further, the refinement and addition of the connector text were necessary in a few cases. Dynamic field population was another challenge faced during proposal document composition. In addition to static knowledge element text, the document contains some information to be dynamically populated. A *knowledge element* text embeds *knowledge parameters* relevant to that offering. For example, a data center service offering will have specific parameters on volumetrics, incidents, data center requirements, middleware, and so on. These parameters had to be populated in the proposal document with specific values as applicable to the RFx. There were also instances in which answers demanded further refining of solution content, as only a part of the solution fragment was found to be relevant. Further work on this is required.

There were many operational challenges during the deployment of this technology. Designing simpler *knowledge types* that are understandable and can easily be related by pre-sales teams was a specific deployment learning. A first-time population of knowledge is a necessary prerequisite for proposal generation. Consolidation of all the NL documents containing solution knowledge of various domains/ offerings was an involved exercise. Hence, over a period of 12-18 months, the system was rolled out for 16 domains and 20+ offerings. Initially, for two major offerings of the BU, it was deployed, and subsequently, in a gradual manner for other domains and offerings. A one-time effort was expended towards creating the deployment architecture for the tool and building specific domain/offering boilerplates along with subject matter experts. Technology transfer was carried out, and the system was handed over to a tools team for day-to-day maintenance of the system. The Proposal system functionality is integrated to the existing BU portal which provided the necessary engineering infrastructure for performance, security etc. In a Business As Usual (BAU) scenario, a 1-2 member administrative team continues to work with SMEs to keep the knowledge repository up-to-date with the latest information and manage user access. There was one resource from the technology team to oversee the maintenance/updates of the tool. As of today, the system is deployed to over 600+ users and is successfully being used to generate proposals for over 30 business offerings.

Any large deployment is expected to have evolution-related challenges. As knowledge elements keep growing, challenges in NLP and context handling are anticipated. The domain dictionary will need to be refined to ensure that questions and answers are correctly aligned. Further changes to the meta-model in-terms of additional elements/properties/associations may need to be carried out as required. ML approaches of learning-based classification of the proposal text and further refinement through the human-in-loop approach are being enhanced. As the system is used, labeled data gets generated for training the ML model which can further improve the learning engine. Albeit a roadmap with a pipeline of enhancements, models and model-driven approaches will continue to play a key role in this system architecture, with models being first-class entities in proposal development.

5. Related Work

The area of bid or proposal management has received little attention in the literature, and there is a scarcity of systematic application of models to assist practitioners in bid/ proposal management, and very little has been published on how the supplier produces the solution proposal (Philbin, 2008). Some approaches are discussed using FastText word embedding and soft cosine measure (Liu et al., 2019). Hamid et al. discuss a linguistic-based approach for identifying requirements from RFP and a similarity-based mapping of requirements to IT Service Solutions (Hamid et al., 2016). Field studies are available on call for tender challenges in practice discusses supplier's risks, mitigation strategies, and decision variables (Hochstetter, 2012). Our prior paper discusses the early work related to RFx response generation and the application of NLP techniques (Rajbhoj et al., 2019). In a recent work, an Artificial Intelligence (AI) smart assistant to automate bid proposal document based on closest prior tender match is discussed (Manchanda, 2021).

The commercial products have primarily focused on the customer side of RFx management. A need is being felt now to automate response development and leverage AI in the process. Some commercial products are venturing into this space (Rfpio, 2023) (Loopio,2023). Most of these approaches are tailored for specific domains like security, due diligence, and so on. Significant work is carried out in question-answer systems that provide relevant answers to posed questions in natural language (Bouziane et al., 2015) (Allam et al., 2012). Even though this space partially overlaps with knowledge-based recommendations, it does not adequately address specific challenges discussed in section 1.1.

MDE techniques have been popular in code-centric phases for generating source-code from models (Hutchinson et al., 2011) (Rajbhoj & Kulkarni, 2013). Model-driven engineering (MDE) is integrated with NLP for converting natural language specification to formal models (Keszocze et al., 2013). AI and knowledge-aware software has remained largely untouched by MDE, and it has the potential to drastically change the way MDE is used and perceived in the software community (Cabot et al., 2018). One of the grand challenges of MDE is to extend MDE techniques by incorporating AI techniques so as to make MDE tools more intelligent and self-aware and to be able to link the heterogeneous views belonging to different steps of SDLC in a model form (Bucchiarone et al.,2020).

Model-based approach is applied in combination with NLP techniques to transform requirements into models and further facilitate generation of various requirement artifacts (Nistala et al., 2022). Model-driven grant proposal engineering for research proposals discusses application of MDE techniques for abstract modelling of research proposal information and model-to-text transformations (Kolovos et al., 2014). Many approaches are proposed for model-to-text transformations, i.e., to generate reports, documents, and manual from models in various domains (Nicolás and Toval, 2009) (Delp et al., 2013). These approaches address modelling and output generation aspects of proposal development but have many limitations in handling the complex facets of RFx context, question classification, knowledge recommendation, and final document generation.

6. Conclusion and Way Forward

Responding to a client-provided RFP/ RFI/ RFx with solution proposals is highly critical in the industry for business development. This proposal development is an expertise-driven, heavily manual activity, and is centered around various types of natural language documents prepared, referred, and maintained. This paper presents an industry success story of a Proposal system developed and successfully deployed to hundreds of presales users across the world. Considering the modelling paradigm and leveraging AI techniques, a Proposal System is developed to transform from document-centric, manual proposal process to a model-based and automated approach.

The key contributions are

- A generic proposal system meta-model that is domain and offering agnostic
- Externalization of solution knowledge in NL documents into machine-processable models
- Solution recommendation for any client-provided RFx comprising a set of questions
- Automatic generation of formatted Proposal document with ~70% of questions answered

The whole approach is model-based, and knowledge guided suitably leveraging NLP, and document technologies. There are several advantages due to the purposive meta-model such as ensuring information consistency, identifying gaps/ missing pieces, and facilitating document-to-models and model-to-text transformation etc. DocToModels provides a meta-model and document-structure agnostic approach to parse NL documents and populate instance models. The proposal system addresses the challenges faced by pre-sales team in-terms of not having a single source of truth, delays in searching for right information, contextual positioning of knowledge, document formatting and so on. Automatic generation of a proposal document with 50-70% questions answered is a huge value-add to the team involved preparing proposals.

The Proposal System technology has been deployed by a large-scale BU across seven geographies. The qualitative feedback from the stakeholder on this technology is highly positive. A distinct productivity improvement in response generation has been observed due to Proposal system technology. Even though one large business unit has deployed this system for their proposal development, it can easily be extended to other business units and pre-sales teams working on RFP/ RFIs as the technology is domain-agnostic. We plan to carry out further research to improve intelligence in solution recommendations through learning engines. Further, we started using model-based approaches for use cases in other document-centric SDLC phases such as requirements as much of our technology enablers can be easily repurposed to other types of NL documents and meta-models.

7. References

1. ANTLR open source, https://www.antlr.org/ last accessed 2023/03/10.
2. Allam, A. M. N., & Haggag, M. H.: The question answering systems: A survey. International Journal of Research and Reviews in Information Sciences (IJRRIS), 2(3). (2012).
3. Bouziane, A., Bouchiha, D., Doumi, N., & Malki, M. (2015). Question answering systems: survey and trends. *Procedia Computer Science*, *73*, 366-375.
4. Bucchiarone, A., Cabot, J., Paige, R.& Pierantonio, A. (2020). Grand challenges in model-driven engineering: an analysis of the state of the research. Software and Systems Modeling, 19(1), 5-13.
5. Cabot, J., Clarisó, R., Brambilla, M., & Gérard, S. (2018). Cognifying model-driven software engineering. In *Software Technologies: Applications and Foundations: STAF 2017 Collocated Workshops, Marburg, Germany, July 17-21, 2017, Revised Selected Papers* (pp. 154-160). Springer International Publishing.
6. Delp, C., Lam, D., Fosse, E., & Lee, C. Y. (2013, March). Model based document and report generation for systems engineering. In 2013 IEEE Aerospace Conference (pp. 1-11). IEEE.
7. Docx4J Homepage, https://www.docx4java.org/ last accessed 2023/03/10.
8. Giudice, D. L.: How AI Will Change Software Development and Applications. Technical report (2016).
9. Hamid R. MDizotahari-Nezhad, Juan M. Cappi, Taiga Nakamurra, Mu Qiao, RFPCog: Linguistic-based Identification and Mapping of Service Requirements in Request for Proposals (RFPs) to IT Service Solutions, 49th Hawaii International Conference on System Sciences, IEEE, 2016
10. Hochstetter, J., Cachero, C., Cares, C., & Sepúlveda, S. (2012, April). Call for tenders challenges in practice: A field study. In *Proc. XV Congreso Iberoamericano en Software Engineering, Buenos Aires, Argentina*.
11. Hutchinson, J., Whittle, J., Rouncefield, M., & Kristoffersen, S. (2011). Empirical assessment of MDE in industry. In Proceedings of the 33rd international conference on software engineering (pp. 471-480).
12. Ikeda, T., Okumura, A., & Muraki, K.: Information Classification and Navigation Based on 5WlH of the Target Information. In COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics, (1998)
13. ISO/IEC/IEEE 12207:2017 Systems and software engineering -- Software life cycle processes.
14. Keszocze, O., Soeken, M., Kuksa, E., & Drechsler, R. (2013). Lips: An ide for model driven engineering based on natural language processing. In 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), 31–38.
15. Kolovos, D. S., Matragkas, N., Williams, J. R., & Paige, R. F. (2014). Model driven grant proposal engineering. In *Model-Driven Engineering Languages and Systems: 17th International Conference, MODELS 2014, Valencia, Spain, September 28–October 3, 2014. Proceedings 17* (pp. 420-432). Springer International Publishing.
16. Kulkarni, V., Reddy, S., & Rajbhoj, A. (2010). Scaling up model driven engineering–experience and lessons learnt. In *Model Driven Engineering Languages and Systems: 13th International Conference, MODELS 2010, Oslo, Norway, October 3-8, 2010, Proceedings, Part II 13* (pp. 331-345). Springer Berlin Heidelberg.
17. Liu, N., Feng, C., Wu, S., Chan, A., & Fulton, J.: Automate RFP response generation process using fasttext word embeddings and soft cosine measure. In Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, pp. 12-17. ACM (2019).
18. Loopio website, https://www.loopio.com, last accessed 2022/03/10.
19. Lucene Homepage, http://lucene.apache.org/ last accessed 2023/03/10.
20. Manchanda, S. (2021). Automation of Bid Proposal Preparation Through AI Smart Assistant. In Data Management, Analytics and Innovation: Proceedings of ICDMAI 2021, Volume 1 (pp. 45-57). Springer Singapore.
21. Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.
22. Model Object Facility (MOF), http://www.omg.org/spec/MOF/2.0 last accessed 2023/03/10.

23. Nicolás, J., & Toval, A. (2009). On the generation of requirements specifications from software engineering models: A systematic literature review. Information and Software Technology, 51(9), 1291-1307.
24. Nistala, P. V., Rajbhoj, A., Kulkarni, V., Soni, S., Nori, K. V., & Reddy, R. (2022). Towards digitalization of requirements: generating context-sensitive user stories from diverse specifications. Automated Software Engineering, 29(1), 26.
25. OpenNLP Homepage, https://opennlp.apache.org/ last accessed 2023/03/10.
26. Paech B., Heinrich R., Zorn-Pauli G., Jung A., Tadjiky S.: Answering a Request for Proposal – Challenges and Proposed Solutions. In: REFSQ Requirements Engineering: Foundation for Software Quality, pp16
27. Philbin S.: Bid management: a systems engineering approach. The Journal of High Technology Management Research 19(1), 14–127, Elsevier (2008).
28. Rajbhoj, A., Nistala, P., Kulkarni, V., Soni, S., & Ajim, P. (2023). DocToModel: Automated Authoring of Models from Diverse Requirements Specification Documents. 45th IEEE/ACM International Conference on Software Engineering (ICSE-SEIP 2023), 10.
29. Rajbhoj, A., Nistala, P., Kulkarni, V., & Ganesan, G. (2019). A RFP system for generating response to a request for proposal. In Proceedings of the 12th Innovations on Software Engineering Conference (formerly known as India Software Engineering Conference) (pp. 1-9).
30. Rajbhoj, A., & Kulkarni, V. (2013). Large Scale Model-Driven Engineering for a Multi-site Team--Experience Report. In 2013 20th Asia-Pacific Software Engineering Conference (APSEC) (Vol. 2, pp. 123-128). IEEE.
31. Rfpio website, https://www.rfpio.com/product, last accessed 2023/03/10.
32. Xie, T.: Intelligent Software Engineering: Synergy Between AI and Software Engineering. In International Symposium on Dependable Software Engineering. Theories, Tools, and Applications, Vol. 10998, pp. 3–7. Springer (2018).