AdaptIE
Using Domain Language concept to enable Domain Experts in Modeling of Information Extraction Plans

Wojciech M. Barczyñski, Felix Foester, Falk Brauer
SAP AG, SAP Research CEC Dresden, Chemnitzer Str. 48e, Dresden, Germany
Wojciech.Barczynski@sap.com, Felix.Foester@sap.com, Falk.Brauer@sap.com

Daniel Schuster
Technische Universitt Dresden, Dresden, Germany
daniel.schuster@tu-dresden.de

Keywords: Information Extraction, Domain Adaption, User Interface, Process Modeling

Abstract: Implementing domain specific Information Extraction (IE) technologies to retrieve structured information from unstructured data is a challenging and complex task. It requires both: IE expertise and domain knowledge, provided by a domain expert who is aware of, e.g., the text corpus specifics and entities of interest. While the IE expert role is addressed by several approaches, less has been done in enabling domain experts in the process of IE development. Our approach targets this issue. We provide a base platform for collaboration of experts' through IE modeling languages. We provide each of the experts with an IE language that is adapted to their respective expertise. IE experts leverage a fine grained view and domain experts can use a coarse grain view on IE plan. We use Model Driven Architecture concept to enable transition among the languages and a model of algebraic IE framework. To prove applicability of our approach we realized *AdaptIE* and demonstrate it in a real world scenario for the SAP Community Network.

1 Introduction

The problem of managing unstructured text and present its information in structured form, commonly known as Information Extraction (IE), is becoming increasingly important. Because the amount of human-generated information (unstructured data) is constantly growing and contains a lot of valuable information, comprising Web 2.0 data as well as enterprise-internal wikis, technical blogs, and community portals. In recent years there has been a considerable amount of interest in developing IE systems, e.g., (Reiss et al., 2008; Shen et al., 2007). Many existing frameworks are general purpose extraction systems - they use algebraic operators or logical programming languages to provide extraction functionality. However, using those kinds of systems imposes severe usability challenges on domain experts. It is important, because (domain) knowledge is a key element in achieving high accuracy of IE (see, e.g., (Sarawagi, 2008)). As domain knowledge we denote both knowledge about semantic and document structure. Contemporary solutions require from domain experts to familiarize themselves with the IE system, to gain technical expertise, and focus on the extraction task itself instead of contributing their domain knowledge to carry out the extraction.

In our work we approach the usability issue of IE languages from a user perspective. In particular we specify this research question as: how to foster the collaboration between IE and domain experts to make them contributing their distinct knowledge into the development of a domain-specific IE system (*DES*)?

Our approach address this question through providing an IE language platform based on an algebraic extraction framework (e.g., (Shen et al., 2008)) and the concept of domain specific languages from Model Driven Architecture (MDA) research (Bosch and Dittrich, 2004). We provide each of the experts with an IE language that is adapted to their respective expertise. An IE expert has a fine grain view and a domain expert leverages a coarse grain view (defined by the IE expert) on a IE plan they both work on. We use MDA concept of model transformation to enable transition among each of languages and algebraic model of IE framework. So created extraction plans are stored in a IE repository for later reuse. Experts' work follows a process that structures actions in creating *DES* from

Generic Extraction System (*GES*).

We realized our approach as an Eclipse-based (http://www.eclipse.org) graphical modeling tool – *AdaptIE*– for both the IE experts as well as the domain experts. To proof benefits of our approach we apply it in a scenario of text extraction of a SDN forum (http://forums.sdn.sap.com). *AdaptIE* was also used to construct IE plans used for entity recognition in a retrieval system presented in (Brauer et al., 2009). Our contributions are: a process model for creating *DES* from *GES* and special-purpose languages for IE extraction task modeling for each of the experts. Our concept, which brings MDA to IE research, follows the call for finding synergies between different research areas to bring new value to database (information systems) research stated in the *claremont report* (Agrawal et al., 2008).

We start our discussion by identifying related work in Section 2. Our work relies on a generic IE framework that we introduce in Section 3. We further generalize and describe in detail a process for creating IE systems for different target domains in Section 4. Section 5 provides insight into *AdaptIE*, which is built on top of an algebraic IE framework. We validate our concepts by considering the problem of extracting products and error messages from the SDN in (Section 6). Section 7 concludes and discuss future directions of our work.

2 Related Work

IE frameworks. *GATE* (Bontcheva et al., 2004) and *UIMA* (Ferruci and Lally, 2004) proposed architectural approach for Information Extraction systems which this time provided a prominent, common infrastructure for IE. Both provide an object oriented framework for black box composition of Information Extraction modules in a pipeline fashion. However, combined modules have to share the same semantics and incorporate information extraction logic and domain knowledge within the code itself.

IE languages. To overcome the drawbacks of architectural approaches, recent work proposes languages for operator composition and separation of domain knowledge from the code. Remarkable approaches in this area are (Reiss et al., 2008) and (DeRose et al., 2007). *Reiss et al.* proposed an *SQL* like language, so called *AQL*, to combine a very small set of well defined operators in an algebraic fashion (Reiss et al., 2008). The aim is to simplify using Information Extraction technology and bring it closer to non IE experts, which are familiar with *SQL*. The drawback of this approach is the limitation for IE experts to plug custom operators, a often required tasks in IE. *DeRose et al.* present a concept of *extraction plans* as an abstract for extraction programs composition. This approach was extended by a *Datalog* alike language for plan creation - *XLOG* (Shen et al., 2007). The authors state, that *XLOG* might help IE engineers to speed up IE plans constructions by relaying on the clear semantic of logic programming language. It also allows the optimization of extraction program composition using power of logical composition. Our approach extends these approaches in three major points. First, we provide a clear separation between extraction logic and domain knowledge without limiting IE experts in extending the extraction logic by introducing new operators. Next, we provide a mechanism and user interfaces for customizing and composing operators into easy understandable extraction programs, pre-compiled via code generation and ready to use by the domain users. Last, we enable domain experts, which are not familiar to IE, to contribute their domain knowledge.

MDA and MDE. We realize our approach using domain specific languages (DSL) and model driven architecture (MDA). The foundations of metamodeling in general, independent from a particular technology are presented in (Favre, 2004a). A metamodel is introduced there and incrementally refined in (Favre, 2004b). *Petrasch et al.* explain in (Petrasch and Meimberg, 2006) in detail the MDA standard defined by the Object Management Group and lays down the fundamentals of model-driven software architecture. *AdaptIE* tool is used to create domain specific languages for each of the users.

3 Concepts and Background

This section presents in details an algebraic extraction framework, which is introduced in (Barczynski et al., 2009). The framework provides a set of generic operators and algebra, which defines how to compose and extend them in order to perform extraction. This clear semantic allows us to make our approach generic for all operators, which are described in the terms of the framework. Moreover we introduce domain independent and domain specific operators. As mentioned in Section 2, there are other algebraic IE frameworks, which could be used as well. Operators work on *annotations*, which are extracted parts (fragments of text) of a document, such as the title of a HTML page or a recognized product. They contain semantic metadata: the entity type (e.g., *SAP Product*), and the extracted entity itself (e.g., *NetWeaver 2004s*) with, if available, a unique id (e.g.,

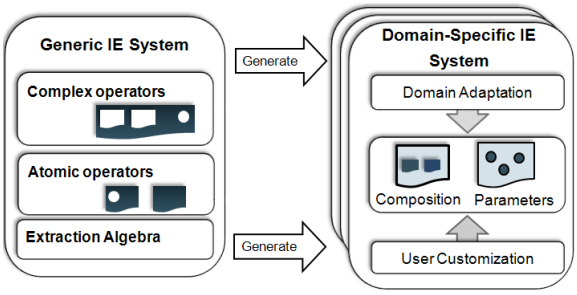

Figure 1: Instantiating a GES to DES.

an OKKAM Id (Bouquet et al., 2008)). An operator takes as input a set of annotations and returns new ones. Every operator has a set of parameters for customization its functionality. Those parameters can be general (e.g., name and id) and domain specific (e.g., regular expression for `ErcRegEx`).

Operators. Each framework operator (see Table 1) is considered atomic if it conducts a single and indivisible task in IE. Tasks include extracting entities, identifying relationships, or combining extracted entities. In our framework we distinguish the following groups of atomic operators: *Basic operators* are used to extract entities from unstructured documents. Grammatical rules or dictionaries can be used for this purpose. *Relation operators* combine extracted annotations to complex entities or map complex entities to structured data. *Set operators*, known in query languages, such as SQL, allow applying group, union and aggregation on extracted entities. *Complex operators* - atomic operators (as well as complex operators) can be chained together to build *complex* extraction operators. Complex operators encapsulate a complete task in the process of IE (e.g., extracting the relation between product and error message).

Domain adaptation. Both types of operators are abstract and only provide a skeleton (structure and functionality), but, without configuration, they lack the domain knowledge necessary for the execution (see Figure 1). By using a generic approach for IE (Generic Extraction System - *GES*) we gain extensibility and generality. But this gain comes along with the need to provide additional information in the framework - domain adaptation, user customization and composition (see Figure 1). Such an customized framework we call Domain Specific Extraction System (*DES*). As such we speak of *domain-independent operators* and, after being provided with domain specific values for parameters, of *domain-specific operators*. The methodology for creating *DES* is provided in Section 4.

Operator Name	Description
`Import`	Imports documents from a given data source.
Basic Operators	
`ErcRegEx`	Extracts text using a regular expression.
`RxR`	Extracts text between two regular expressions.
`SentenceEx`	Detects sentences.
`NounGrpEx`	Creates annotations for nouns groups.
`ErcDict`	Extracts using a dictionary.
Relation Operators	
`ErcRel`	Relates annotations with objects in structured data.
`Relate`	Combines two entities to one complex entity
Set Operators	
`Group`	Groups annotations by document id.

Table 1: Example of operators in the IE framework.

4 User-centric IE

This section provides a classification of user roles in IE, followed by an overview and an explanation of our approach. Specifically, we explain the sequence of steps, called *IE (domain) adaptation process*.

4.1 Users of IE systems

Different users are involved in the process of extracting data and as a consequence, they use IE systems to fulfill different tasks. We classify users by their level of expertise and distinguish two such types. We leave out end users that use a completed *DES*, because this issue does not bring anything new to our discussion.

First, we have *IE experts* (e.g., IE Consultant and IE Engineer) that are familiar with the concepts of IE. They know about NLP, operators, about variability of their settings, and they also know about the right order in which they have to be arranged (e.g., noun grouping before applying other extraction operators). These users are able to create extraction plans by combining and configuring the IE operators appropriately.

Second, there are *domain experts* (e.g., Product Manager). They should concentrate on their specific area and contribute their domain knowledge rather than focus on the extraction task itself. Regarding IE, they do not need to be aware of system internals. They usually do not have IE expertise and they are, thus, not able to deal with the complexity of IE. They need to formulate complex queries against unstructured data. They require a user interface, which let to focus on their task.

4.2 IE adaptation process

Figure 2 shows an overview of our approach. A base is a IE framework (*GES*) - see Figure 2a. As the do-

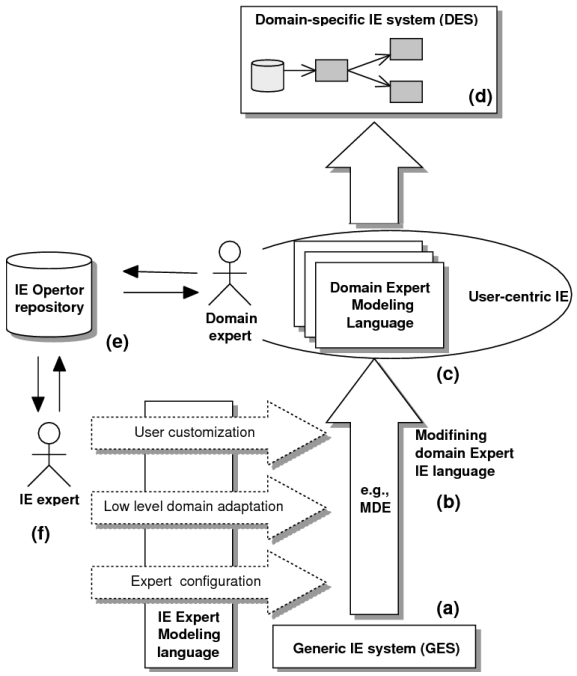

Figure 2: IE adaptation

main expert (Figure 2c) doesn't have the expertise to deal with the complexity of creating a complex extraction program, the IE expert (Figure 2f) uses an *adaptation process* (Figure 2b) to prepare and preconfigure operators for the domain expert. It is the initial step; afterwards whole process consists of iterations over IE plan by both experts. First the IE expert prepares operators with generic configuration to the domain expert. Adaptation consists of combining and adjusting atomic operators appropriately as well as performing a customization tailored to specific domain expert's needs. Additionally – in next iterations after receiving feedback from the domain expert – the IE expert tunes operators based on his IE knowledge, e.g., by setting caching strategies. As such, a set of domain-specific extraction operators can be created and stored in a repository. The adaptation is done by using an expert graphical interface. As a result domain expert is provided with necessary operators in his IE modeling language. The language uses familiar vocabulary (e.g., names of operators) and hides all fine-grained tuning parameters. Using this language, the domain expert contributes his domain knowledge (e.g., knowledge about the structure of forum threads) using an intuitive GUI (see Figure 7), which allows modeling concrete extraction programs. Subsequently, a domain expert can import and export previously defined (complex) operators from repository (Figure 2e). Moreover an IE expert can access adapted operators to specific domain. Final result of several iterations is a *DES*, depicted in Figure 2d.

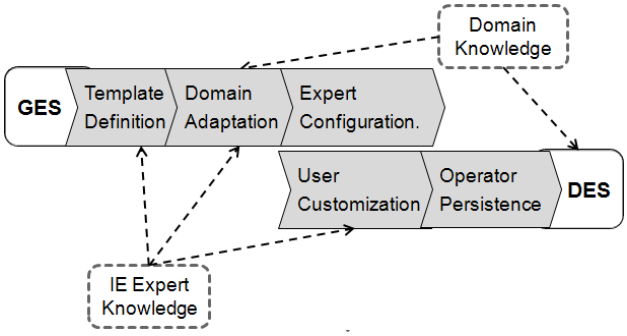

Figure 3: Actions necessary to provide DES.

4.3 Users' Roles and Actions

Several steps have to be performed to incorporate the needed information in the *GES*, and to create a *DES*. Those steps form an adaptation process which is depicted in Figure 3 (see also Figure 2b and Figure 2e). In the following we will discuss the necessary steps and focus on interaction between IE and domain experts.

Template Definition. The IE expert is solely responsible for defining templates. Templates are a mean to *describe* the structure of each operator type. They can be written by defining XML-files by hand or by using the IE expert interface. Parameters, their types, and default values can be defined. Every template consists of parameters common to all operators (e.g. id, name, and version), but also allows the definition of operator-specific parameters. The usage of an operator may be limited to a certain context. For instance, in contrast to other operators, the `Import`-operator must not have predecessor operators, or `ErcDict`, the dictionary-based operator needs to be connected to an external data source providing the dictionary. Templates serve three purposes. First, they decouple IE plan modeling from the concrete implementation in the IE system. Second, they help to automate the process of extending the system with new operators (add to the tool GUI) and third, they constrain the application of an operator.

Domain Adaptation. An IE system needs to be configured appropriately for one application domain, thus, domain-specific information has to be added.

Refining Operators. Refining operators means to provide values for parameters contained in atomic operators. We distinguish *common* parameters (such as name, and version), *performance* parameters (e.g.

caching, memory usage), and *domain* parameters (e.g. location of dictionary). Parameters may be left unspecified by the IE expert, in that case, we speak of *partial refinement*.

Composing Operators. Operators might require (e.g., for achieving higher precision) the presence and execution of other operators before they can be used. This usually includes operators for importing data or for natural language (pre-) processing, such as annotating sentences and noun groups. Thus, it is possible to combine atomic extraction operators in the right order to model complex extraction operators or to combine atomic and complex operators to build a complete extraction program. This is called *operator composition* and can be done before or after operator refinement.

Both, IE expert and domain expert are responsible for performing the domain adaptation. For the expert it means to translate requirements formulated by an domain expert to the world of IE and (partially) preconfigure domain-specific operators. The domain expert, based on his domain knowledge, completes the partial refinement and models the final extraction plan using a *DES* (Figure 2d).

Expert configuration. The IE expert, based on his knowledge or in response to the domain expert request, performs fine-grained tuning. For example, he redefines operators for achieving higher recall.

User Customization. After providing the domain-knowledge, the IE expert performs final adjustments to hide the complexity and the details from the domain expert, thus *customizing* the system to domain expert requirements. We introduce here different categories of customization and how they are performed to provide user-centric IE. We denote those categories *dimensions of customization*. Customization can be applied on the following dimensions:

Operator Parameters. We allow the IE expert to hide parameters, set them read-only and provide default values. This provides a higher level of abstraction for the domain expert.

Operator Composition. Complex operators are created by combining atomic operators. They hide internal structure and complexity of extraction flows. Certain steps of the IE process are not of interest for the domain expert but nevertheless important and needed in overall process.

Operator Documentation. As soon as an operator is configured and provided with parameters, it is ready to be used in a domain-specific context. As such, the semantics of the operator specified. This should be reflected in its description, help, and examples. We use a description-attribute for this purpose.

Debug and Runtime Feedback. If the IE expert changes the IE language for the domain expert it should affect also error and debug messages. The domain expert should see "Product extractor failed..." instead of "`ErcRgx` operator...". This dimension is changed implicitly by changes in the IE language.

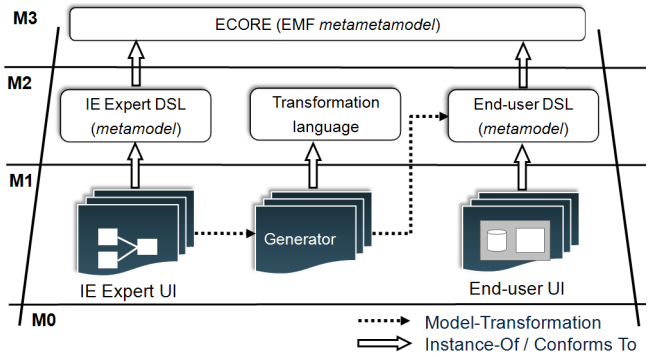

Figure 4: Model transformation is used to implement the user customization.

DES. The domain expert uses the *DES* to build extraction programs. He can model extraction plans using prepared complex operators. Besides, if there is a need, he can contact the IE expert and request the tuning his operators. The IE expert will perform one of the actions shown in Figure 3. We allow the domain expert to perform domain adaptation by operator composition and parameter adjustments.

Operator Persistence. All domain-specific operators are persisted in an operator repository. The major aim of an operator repository is to store operators, for future retrieval and reuse. Domain-specific operators are associated and persisted within the domain they are applied to. Domain experts can choose a domain they want to work with, and load a profile containing all the operators for this specific domain.

5 Implementation

AdaptIE tool is built on top of Eclipse Modeling Framework. We use different metamodels to provide abstract syntax for our extraction languages. Additionally, we use GMF (GMF, 2009) to provide concrete syntax through graphical user interfaces. Figure 4 shows the 4-layer metamodel architecture. Elements on one layer are said to be instances of elements from the above layer and specify elements on the layer below. On M0 are concrete IE plans modeled in UIs in M1. On the very left in Figure 4 (level M1) we have extraction programs or complex operators conforming to the expert metamodel. The M2 is described by

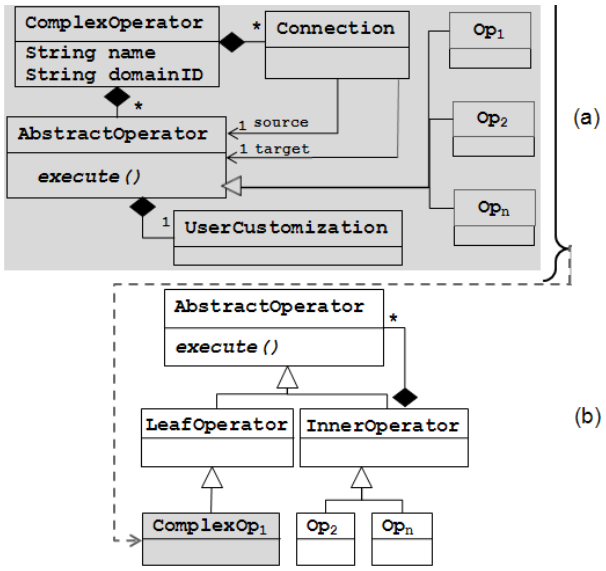

Figure 5: IE expert (a) and domain expert (b) metamodels (simplified).

means of its metamodel - M3 (Ecore (EMF, 2008)). The same applies for composition programs written using the end-user (domain expert) DSL. Those programs conform to the end-user metamodel.

In this work we analyze concrete instances of "expert operators" located on M1. We generate elements for the end-user metamodel, which resides on M2 i.e., we need to cross the layers. The transformation is visualized by the dashed bold arrow in Figure 4. We describe model transformations by defining relations between two models. The mapping between source and target model is described in terms of their metamodels i.e., one level higher than input and output model. In our case we describe rules using Ecore elements.

The transformation itself has to be defined as a model conforming to a metamodel defining model transformation semantics. Java and the ATLAS Transformation Language (ATL) (ATLAS, 2006) are languages that provide means to define and execute a model transformation. In the prototype we used Java, but we consider to use ATL in the next versions of *AdaptIE*. An excerpt of the IE expert metamodel is shown in the UML class diagram in Figure 5a. Intuitively, it allows modeling an instance of `ComplexOperator` by combining `AbstractOperators` with `Connections`. The `UserCustomization` class defines a user-centric adjustment. The domain expert metamodel (Figure 5b) allows to nest `InnerOperators` until a `LeafOperator` (representing an atomic operator) is used, which does not allow further nesting.

We use model transformation for two purposes: First, it allows combining complex operators created in the expert language using the domain expert language. Second, we can use model transformation for implementing the user customization. This takes the form of analyzing concrete instances of the expert metamodel and generating elements for the domain expert metamodel. For example, a parameter marked "invisible" will not appear in the domain expert model.

6 Evaluation

As a method to proof applicability of our methodology, we have selected a scenario based evaluation (Hevner et al., 2004). Because we want to show that our approach can be implemented and used to solve issues in a real world scenario.

Scenario. We consider analysis of relation between *SAP Products* and error messages (*Java* and *ABAP exceptions*) in SDN forum. Results of such analysis can help to understand which kind of problems developers and consultants encounter. In this scenario a product manager is our domain expert. He knows well the forum, conventions used by its members. Moreover he has access to structured data about products, error messages, etc. He starts by selecting the data source for analysis, e.g., RSS feed. The manager configures the `Import`-operator himself. He provides the number of items to be imported and the database connectivity information. The next step is to extract products and error messages. As the domain expert, he is not interested how to extract things, so he uses an operator repository and search extractors for his domain. There are available *Java*, and *ABAP* exceptions extractors, but no operator for *SAP Product*. Therefore he contacts a IE expert, and ask him for such an operator. Moreover he provides to the IE expert a taxonomy about *SAP Products* - SAP Terms database. SAP Terms database contain all terminology used in documentation about a product and list of product's subcomponents.

The IE expert does not need to understand of SDN forum data, so she can focus on her task to provide an operator, which is able to use the taxonomy. Using *AdaptIE*, she composes operators as shown on Figure 6b. The IE expert starts by applying preprocessing operators for a morphological analysis. First, she extracts sentences (`SentenceEx`) followed by noun groups (`NounGrpEx`), thus, rejecting prepositions, conjunctions, and relative pronouns that are not considered important. Next, she applies `ErcDict`, realizing a connection to the SAP Term database. The

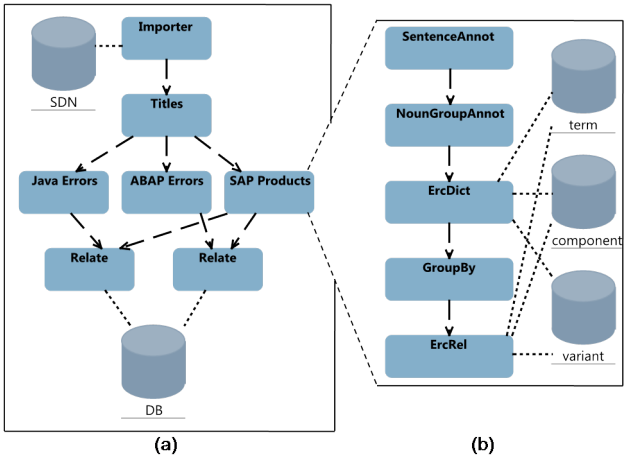

Figure 6: (a) Plan for extracting products and errors (b) Complex operator to identify SAP products (IE Expert's view).

IE expert provides database access information and decides to set the `isCached` property to keep extracted entities in memory thus reducing database access time. Next, she saves the new IE plan as a operator and store it in repository. The IE expert specifies additionally that a domain expert will see *SAP products* as a atomic operator.

Now, the product manager can import the ready-to-use operators and connect them to build the final extraction program (Figure 6 (a)). He knows that most of forum members put in their post about problems *SAP products* and errors' names together in the HTML title. Moreover he knows that his product is used in two scenarios with software components written in Java and Abap. Therefore he selects to extract both *Java* and *ABAP exceptions*. Figure 6 shows the complete extraction program modeled by the product manager, and the complex operator for identifying *SAP Products* as modeled by the IE expert. If operators don't perform as expected, he may consult the expert again. So he does not need to spend a lot of time trying to find a mistake in, e.g., regular expression for *Java Exceptions* extraction: `([a-zA-Z]*\\.)*[a-zA-Z]+(Exception|Error)[a-zA-Z]*`. As such, they can both work on refinement iteratively and use the operator repository (powered by the transformation engine) for exchanging operators.

Tools. The current implementation of *AdaptIE* includes tools for the IE and domain experts. Figure 7a shows the IE expert's tool. It allows to create extraction programs and to model complex operators. The expert has access to all parameters and configuration options. A low-level view on operators allows him to fine-tune the execution of complex IE plans. Furthermore, the tool allows him to perform a user customization, i.e., renaming operators and parameters, setting parameters, marking them mandatory, visible, or read-only as well as providing default values for the end-user. Once operators have been created, they can be associated with a target domain and archived in the repository.

The domain expert tool (Figure 7b) allows for building the final extraction program by combining previously defined domain-specific operators. Access to the repository and import of complex operators is supported. We investigate currently more a declarative UI, which fallows the structure of source documents.

7 Conclusions

To summarize, this paper contribute a IE language platform that can be a base for involving domain experts in IE plan modeling. We presented a process for creating a domain-specific IE system. We showed how our approach can be realized using MDE and a generic IE framework. As a proof of concept we have developed *AdaptIE* and show how it can be successful applied in a real world scenario. As a future work, we want to continue to bring IE to causal users in two directions. The first direction is further investigation on IE languages for non IE expert users. The second direction is work on automatic generation (driven by information available in, e.g., database schema or descriptions of OLAP cubes) of IE plan to simplify performing ad-hock Business analysis over unstructured data.

REFERENCES

Agrawal, R., Ailamaki, A., Bernstein, P. A., Brewer, E. A., Carey, M. J., Chaudhuri, S., Doan, A., Florescu, D., Franklin, M. J., Molina, H. G., Gehrke, J., Gruenwald, L., Haas, L. M., Halevy, A. Y., Hellerstein, J. M., Ioannidis, Y. E., Korth, H. F., Kossmann, D., Madden, S., Magoulas, R., Ooi, B. C., O'Reilly, T., Ramakrishnan, R., Sarawagi, S., Stonebraker, M., Szalay, A. S., and Weikum, G. (2008). The claremont report on database research. *SIGMOD Rec.*, 37(3):9–19.

ATLAS (2006). *Atlas Transformation Language (ATL) User Manual v0.7*. Nantes.

Barczynski, W. M., Brauer, F., Loeser, A., and Mocan, A. (2009). Algebraic information extraction of enterprise data: Methodology and operators. In *IK-KR Workshop at 20th International Joint Conference on Artificial Intelligence 2009 (to be published)*.

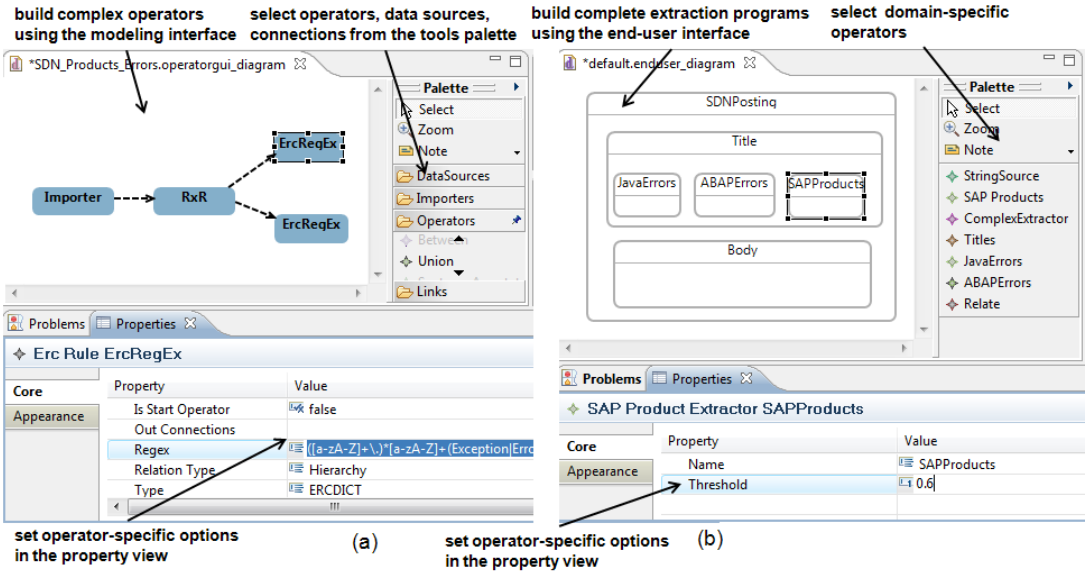

Figure 7: Visual tools for IE expert (a) and end-user (b).

Bézivin, J. and Heckel, R., editors (2005). *Language Engineering for Model-Driven Software Development, 29. February - 5. March 2004*, volume 04101 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Bontcheva, K., Tablan, V., Maynard, D., and Cunningham, H. (2004). Evolving gate to meet new challenges in language engineering. *Natural Language Engineering*, 10(3-4):349–373.

Bosch, J. and Dittrich, Y. (2004). Domain-Specific Languages for a Changing World.

Bouquet, P., Stoermer, H., Niederee, C., and Mana, A. (2008). Entity Name System: The Backbone of an Open and Scalable Web of Data. In *ICSC 2008*, number CSS-ICSC 2008-4-28-25 in CSS-ICSC, pages 554–561. IEEE Computer Society.

Brauer, F., Barczynski, W., Hackenbroich, G., Schramm, M., Mocan, A., and Foerster, F. (2009). Rankie: Document retrieval on ranked entity graphs (demo). In *35th conference International Conference on Very Large Data Bases (VLDB) 2009*.

DeRose, P., Shen, W., 0002, F. C., Doan, A., and Ramakrishnan, R. (2007). Building structured web community portals: A top-down, compositional, and incremental approach. In (Koch et al., 2007), pages 399–410.

EMF (2008). Eclipse Modeling Framework. Documentation available at http://www.eclipse.org/modeling/emf/.

Favre, J.-M. (2004a). Foundations of meta-pyramids: Languages vs. metamodels - episode ii: Story of thotus the baboon1. In (Bézivin and Heckel, 2005).

Favre, J.-M. (2004b). Foundations of model (driven) (reverse) engineering : Models - episode i: Stories of the fidus papyrus and of the solarus. In (Bézivin and Heckel, 2005).

Ferruci, D. and Lally, A. (2004). Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348.

GMF (2009). http://gmf.eclipse.org.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1).

Koch, C., Gehrke, J., Garofalakis, M. N., Srivastava, D., Aberer, K., Deshpande, A., Florescu, D., Chan, C. Y., Ganti, V., Kanne, C.-C., Klas, W., and Neuhold, E. J., editors (2007). *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*. ACM.

Petrasch, R. and Meimberg, O. (2006). *Model Driven Architecture: eine praxisorientierte Einführung in die MDA*. dpunkt-Verl.

Reiss, F., Raghavan, S., Krishnamurthy, R., Zhu, H., and Vaithyanathan, S. (2008). An algebraic approach to rule-based information extraction. In *ICDE*, pages 933–942. IEEE.

Sarawagi, S. (2008). Information extraction. *Foundations and Trends in Databases*, 1(3):261–377.

Shen, W., DeRose, P., McCann, R., Doan, A., and Ramakrishnan, R. (2008). Toward best-effort information extraction. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1031–1042, New York, NY, USA. ACM.

Shen, W., Doan, A., Naughton, J. F., and Ramakrishnan, R. (2007). Declarative information extraction using datalog with embedded extraction predicates. In (Koch et al., 2007), pages 1033–1044.