An Intelligent Contract Editor

Shaun Azzopardi
Department of Computer Science
University of Malta
Email: shaun.azzopardi.10@um.edu.mt

Gordon Pace
Department of Computer Science
University of Malta
Email: gordon.pace@um.edu.mt

Albert Gatt
Institute of Linguistics
University of Malta
Email: albert.gatt@um.edu.mt

***Abstract*—Architects have computer tools that complement their specialist knowledge, as do engineers. Notaries, however, do not. This is in spite of the work that has been done in formally modelling contracts, the basic work of notaries being contract drafting.**

We are currently working on a solution aimed at aiding and complementing notaries' expertise. This will take the form of add-in in to MS Word, the document processing program of choice of local notaries.

Some of the functionality envisioned includes cross-referencing with the laws of Malta, automatically identifying parties involved in a clause, tracking contract changes and conflict detection.

I. Introduction

The main place of IT in any professional's life has been to provide solutions that aid them in their day-to-day work. For example, architects and engineers both use intelligent design solutions the complement their technical knowledge.

Other professionals, specifically notaries, do not have such intelligent solutions. Notaries engage in contract-drafting, with the only computer aids available to them being simple word processing solutions and spell-checking. This is surprising given that contracts have been studied extensively by both philosophers and computer scientists.

Contracts are agreements between two or more parties consisting mainly of obligation, permission and prohibition clauses that restrict the involved parties' behaviour. These concepts have been studied formally in the field of deontic logic, allowing formal modelling of contracts.

Work has also been done in the translation from a specific subset of English to formal languages [1]. This formal modelling allows us to formally analyse contracts, for example certain formalisms allow us to detect conflicts between clauses automatically [2].

Our project will focus on providing a working solution for notaries while also building tools that allow us to formally analyse natural language contracts. Some of the features already implemented include cross-referencing with the laws of Malta, identifying the named entities mentioned and coming up with a set of keywords for the contract. The more theoretical aspect of our project will focus on using linguistic tools to extract semantic information from contracts and thus make it amenable to formal analysis. Other features we are working to implement are conflict detection, party identification and changes tracking.

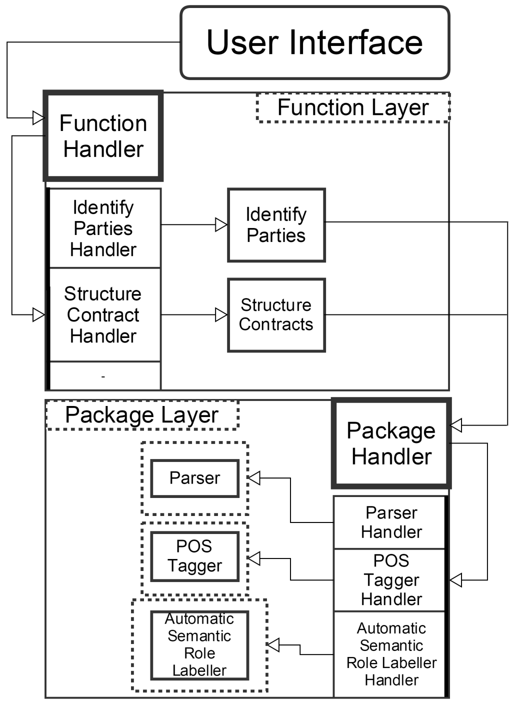


Fig. 1. Solution pluggability architecture.

We have also developed an appropriate architecture for our contract editor, using a dependency injection pattern. This ensures that specific features are implemented into specific modules, that can be replaced at runtime without the need for re-compilation. This ensures loose coupling and easy updating of different features without affecting the whole solution.

The current prototype, that uses this architecture, has been developed using C#.NET. It's user interface is as a Microsoft Word [3] add-in, using Add-In Express [4] that allows the add-in to work on every full version of Microsoft Office on Windows.

The next section will detail this architecture. Then we look at the user experience by looking at some of the features we have developed. Finally we conclude with a summary of the work done and future work.

II. An Architecture for Contract Writing Support

As noted, our solution involves the use of linguistic tools, such as a dependency parser and a part-of-speech parser. There are multiple implementations of these, and moreover most are not developed in C#, our language of choice. To overcome this problem our architecture uses loosely coupled modules and C# wrappers for packages from other languages.

Figure 1 details this architecture. It consists of three layers: the user interface, the function layer and the package layer. Communication between these layers in our implementation occurs via the JSON protocol that specifies the module and function to be called. This ensures that each layer is also replaceable with something with the same interface. The modules developed can also be used in other solutions without any need for modification.

As can be seen from the figure, the function and package layers have a similar structure. They both consist of handlers and of modules, in the case of the function layer these consist of specific functionality tied to contract-drafting, while for the package layer these consist of needed general packages. The latter modules are also wrapped with a C# class, given that most linguistic tools tend not to have the same interface and not be developed in C#.

The bottom two layers also share a handler structure, which consists of a main handler (Function Handler and Package Handler) and of a sub-handler for each module. The purpose of the former is to serve as the only entry point for the layer. These then pass on any request to the sub-handler associated with the needed module. These sub-handlers handle any conversion and packaging of any parameters and output of the package, which is passed back to the main handler.

Each sub-handler knows which module to call through the use of a configuration file. This file can be edited to change the module that is called. For example, one of the linguistic modules is a Named Entity Recognition (NER). NE recognisers use a statistical model to identify human and non-human named entities. Consider now a situation in which an existing NER module, the Stanford NER [6], has been incorporated into the system, but now needs to be replaced by a new one, say, the LingPipe NER module [5]. In this case, the transition can be achieved seamlessly by simply editing the file and re-running the solution without the need for re-compilation of the solution.

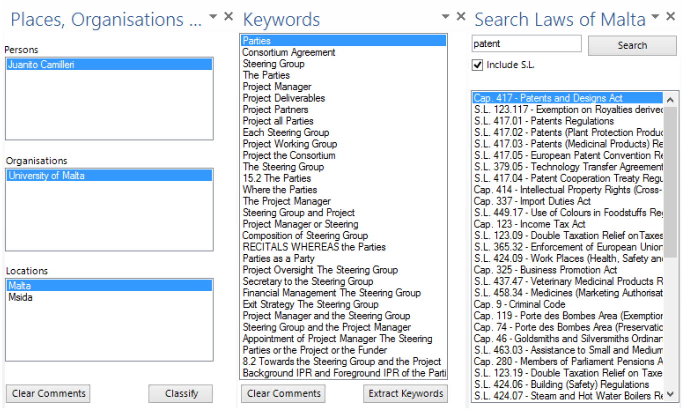
Our architecture thus allows different modules to be developed separately and independently of our system, and as long as they expose the same interface then they can be used in the system easily.

III. UX Considerations

Each of our solution's current working features have been finalized as a task pane in Word.

In Figure 2 one can see three different task panes, the first identifying any entities mentioned in the contract through named entity recognition, the second identifying keywords and the third allowing cross-referencing terms appearing in the contract with the laws of Malta.

The first two features also tag the contract with their results, i.e. clauses are marked with entities mentioned in them and keywords, if any. They also allow highlighting of the selected terms in the contract, for easy retrieval of clauses mentioning them.



Persons
Juanito Camilleri

Organisations
University of Malta

Locations
Malta
Msida

Parties
Consortium Agreement
Steering Group
The Parties
Project Manager
Project Deliverables
Project Partners
Project all Parties
Each Steering Group
Project Working Group
Project the Consortium
The Steering Group
15.2 The Parties
Where the Parties
The Project Manager
Steering Group and Project
Project Manager or Steering
Composition of Steering Group
RECITALS WHEREAS the Parties
Parties as a Party
Project Oversight The Steering Group
Secretary to the Steering Group
Financial Management The Steering Group
Exit Strategy The Steering Group
Project Manager and the Steering Group
Steering Group and the Project Manager
Appointment of Project Manager The Steering
Parties or the Project or the Funder
8.2 Towards the Steering Group and the Project
Background IPR and Foreground IPR of the Parti

patent
[✓] Include S.L.

Cap. 417 - Patents and Designs Act
S.L. 123.117 - Exemption on Royalties derived
S.L. 417.01 - Patents Regulations
S.L. 417.02 - Patents (Plant Protection Produ
S.L. 417.03 - Patents (Medicinal Products) Re
S.L. 417.05 - European Patent Convention R
S.L. 379.05 - Technology Transfer Agreement
S.L. 417.04 - Patent Cooperation Treaty Regu
Cap. 414 - Intellectual Property Rights (Cross-
Cap. 337 - Import Duties Act
S.L. 449.17 - Use of Colours in Foodstuffs Re
Cap. 123 - Income Tax Act
S.L. 123.09 - Double Taxation Relief on Taxes
S.L. 365.32 - Enforcement of European Union
S.L. 424.09 - Work Places (Health, Safety an
Cap. 325 - Business Promotion Act
S.L. 437.47 - Veterinary Medicinal Products R
S.L. 458.34 - Medicines (Marketing Authorisat
Cap. 9 - Criminal Code
Cap. 119 - Porte des Bombes Area (Exemptio
Cap. 74 - Porte des Bombes Area (Preservatio
Cap. 46 - Goldsmiths and Silversmiths Ordinar
S.L. 463.03 - Assistance to Small and Medium
Cap. 280 - Members of Parliament Pensions A
S.L. 123.19 - Double Taxation Relief on Taxe
S.L. 424.06 - Building (Safety) Regulations
S.L. 424.07 - Steam and Hot Water Boilers R

Fig. 2. Different features as exposed to the user.

The search task pane enables the relevant law to be opened simply with a double click on the relevant item, and automatically the word searched for is marked in the document opened.

IV. Conclusion

Our prototype for an intelligent contract editor already has several features already included, with the next step being testing it with notaries to get their response, especially on the user interface.

An interesting aspect of this prototype is the architecture we have developed that allows easy exchanging of modules before run-time, which is useful for testing and also enables easy upgrading of these modules. This architecture also ensures the re-usability of the modules we have packaged and the features we have developed.

Future work includes working on the contract changes tracker, to identify any inconsistencies or conflicts that a change to a contract might have We also aim to use this to formally analyse the different versions of the same contract with respect to each other. More work also is being done on identifying the parties, using a semantic role labeller, which is essential to being able to translate the contract to a formal model.

References

[1] S. M. Montazeri, N. K. S. Roy, and G. Schneider, "From Contracts in Structured English to CL Specifications," in *5th Workshop on Formal Languages and Analysis of Contract-Oriented Software 2011 (FLACOS'11)*, ser. Electronic Proceedings in Theoretical Computer Science, E. Pimentel and V. Valero, Eds., vol. 68, Málaga, Spain, September 2011, pp. 55–69.

[2] S. Fenech, G. J. Pace, and G. Schneider, "Automatic Conflict Detection on Contracts," in *6th International Colloquium on Theoretical Aspects of Computing (ICTAC'09)*, ser. Lecture Notes in Computer Science, vol. 5684. Springer Verlag, 2009.

[3] Microsoft, "Microsoft Word," https://products.office.com/en-us/word/.

[4] Add-in Express Ltd., "Add-In Express," https://www.add-in-express.com/.

[5] Alias-i, "LingPipe 4.1.0. ," http://alias-i.com/lingpipe/, 2008.

[6] The Stanford Natural Language Processing Group, "Named Entity Recognition ," http://nlp.stanford.edu/ner/.