Abstract
The aim of this informal deliverable is to describe some of the results obtained as an outcome of the process of the development of the Legal Case Study first prototype, and what are the next steps regarding with the development its.

Keyword list: legal case study, development

WP10 Case Study: Intelligent integrated decision support for legal professionals.

<table>
<thead>
<tr>
<th>Prototype</th>
<th>CO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contractual date of delivery</td>
<td>Actual date of delivery</td>
</tr>
<tr>
<td>31/12/2005</td>
<td>31/12/2005</td>
</tr>
</tbody>
</table>
SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IPS 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Empolis GmbH
Europapallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540
Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592
Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891
Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@cs.shef.ac.uk

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475
Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006
Madrid
Spain
Tel: +34 913 349 797
Fax: +34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00
Fax: +41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912
Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Sirma AI EAD, Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vall` es)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es
Executive Summary

The aim of this informal deliverable is to describe the current status of the Legal Case Study Prototype. This prototype will be delivered on December 2005 as a result of deliverable D.10.3.1. Prototype.

This document describes the current design of this prototype, marking on several diagrams what the evolution of the prototype was, from the preliminary version of the design presented at Deliverable D10.2.1 [1]. Also, we describe some identified improvements, related most of them with the semantic similarity calculation.

The deliverable ends with an enumeration of next steps and conclusions related with the design adopted, and the integration with SEKT technologies.
## Contents

SEKT Consortium ......................................................................................................................... 2  
Executive Summary ....................................................................................................................... 3  
Contents ........................................................................................................................................ 4  
1 Introduction ................................................................................................................................ 5  
2 FAQ System Architecture ............................................................................................................. 6  
   2.1 Main Stages ............................................................................................................................ 7  
      2.1.1 Ontology Domain Detection .......................................................................................... 7  
      2.1.2 Keyword/Synonym detection ....................................................................................... 8  
      2.1.3 Ontology/Semantic Distance ....................................................................................... 8  
3 Scenario Description .................................................................................................................. 9  
   3.1 User Search Scenarios .......................................................................................................... 9  
      3.1.1 User question search scenario .................................................................................. 9  
       3.2 System scenarios .......................................................................................................... 23  
          3.2.1 Search system initialization use case .................................................................... 23  
       3.3 Administrator scenarios ............................................................................................... 24  
          3.3.1 Tuning iFAQ System use case ............................................................................ 24  
4 Architecture diagrams .............................................................................................................. 26  
   4.1 Component diagram ............................................................................................................ 26  
   4.2 Classes diagrams ................................................................................................................ 27  
      4.2.1 Administration Class Diagram ................................................................................... 28  
      4.2.2 FAQ and Domain class diagram .............................................................................. 28  
      4.2.3 Search class diagram ............................................................................................... 29  
      4.2.4 NLP class diagram .................................................................................................... 30  
      4.2.5 Ontology class diagram ............................................................................................ 31  
      4.2.6 Persistence class diagram ........................................................................................ 32  
   4.3 Deployment diagram ......................................................................................................... 32  
   4.4 SEKT Technology integration diagram ............................................................................. 33  
5 Ontology of Professional Judicial Knowledge ............................................................................ 34  
6 Measurements ............................................................................................................................. 35  
7 Next Steps .................................................................................................................................. 36  
8 Conclusions ................................................................................................................................. 36  
9 References .................................................................................................................................... 37  
10 Figure List ................................................................................................................................. 38
1 Introduction.

The aim of this deliverable is to describe the first draft of the legal case study prototype. This prototype will be used next September 2005 at the Case Study Meeting in Barcelona. The aim of this meeting is to work with a set of final users to validate the user needs and the functionalities of the system. This meeting has been organized by WP8 Usability, optimisation of business benefits and benchmarking.

The approach followed is the prototype is to develop an application where the effort will be centred in user interaction and knowledge base developments.

As a result of this meeting we expect to catch more requirements and identify new applications of the SEKT technology.

At this deliverable, we describe the first part of the prototype. This part is related with the development of the FAQ searching system (on the left hand side of the Figure 1.1.). The aim of this system is to find the best match between a question made by a user (usually a novel judge) and the expert knowledge of the judges stored in the form of a set of question-answer pairs. To find what the question-answer that better matches (semantic similarity) with the input question is, different technologies are applied such as:

- Natural language analysis and processing, to link the terms in the sentence to the ontology.
- Calculate the semantic distance between the set of terms of the input sentence and the set of terms of the stored questions. This process is ontology-based.

The document is divided in seven main sections. Section 2 comprises the description of the architecture and the approach followed to design the final system. Section 3
describes the scenarios of this application, and the main classes and the sequence diagrams. At section 4 we describe the current version of the ontology of the professional judicial knowledge. Section 5 comprises the legal case study measurements. Section 6 describes next steps and the integration with the technological workpackages. Finally, we describe some conclusions about the system.

2 FAQ System Architecture.

Our improved FAQ searching system design is based on Spanish NLP, background calculation of ontologies, caching of background calculated data, and a multistage searching approach with progressive delimitation of the FAQ target.

The software design takes into account the interaction of the user with the system. A user types a question and expects the system to find out a FAQ candidate as close as possible to the question. To achieve this goal, several search and score algorithms have been designed based on Natural Language Processing and on Ontology Concepts Matching [3]. Algorithms have been organized around an architecture based on an adaptive multistage search chain, which is based on a variation of the “chain of responsibility” pattern:

- The search process is subdivided into several cooperative stages.
- At each stage, the system applies an algorithm that reduces the incoming target FAQs, producing a narrower version of that set.
- Each search stage is premised on the belief that the FAQ’s question that best fits the question posed by the user belongs to the outcome set of FAQ’s at that stage.
- The multistage search algorithm allows the system to stop when the FAQ target has been reduced considerably (beyond a parameterized threshold). Therefore, the stages are not necessarily exhausted, and the computational cost is reduced and adapted to the search needs.
- Besides, multistage searching allows the system to show the best score results to the user before completing all available search stages. In this way, users can decide whether to accept the result or continue with the remaining search stage.

To reach this target the design of the system is based in some technological considerations that we summarize in the following items:

- Accurate search. This system will be able to find out the best possible matching FAQ question with user question.
- Efficiency. Searching must not take a long time to find a result successfully and must scale well with FAQ repository size.

In order to reach these objectives, the main technologies used in this architecture are:

- Natural Language Processing. NLP is used at several search stages to get additional comprehension from the user’s question. A morphological analysis of the user’s question is performed. The relevant words and grammatical patterns drawn from the question are used by other components in further stages.
• Thesaurus Processing. It is used to match words based on synonym relationships. The system attempts at both exact and synonym matching.
• Ontology Processing. The system uses ontologies to obtain additional understanding of the user’s question. The system tries to find out a track of the user’s question or part of it in the ontology. To do so, it builds a graph path that is compared to each of the stored FAQ graph paths\(^1\).
• Cache proxy. The system produces intermediate results of repetitive calculations that can be saved to avoid the repetition of computations. Many of these calculations can also be recovered from a repository like a RDBMS and saved on cached memory.

In the next subsections we describe each stage.

2.1 Main Stages.

2.1.1 Ontology Domain Detection.

As we describe at deliverable D10.2.1. [1], the main purpose of this stage is to determine the FAQ domain target set, based on user question analysis, for its usage at a later stage. Complete FAQ database is made up over some domain FAQ databases as building blocks. Reducing the FAQ database domain target is the first consideration to take into account, so the first search stage focuses on it. This goal can be made using two different approaches. First one is based on the statistical recount of occurrences of all relevant concepts of the user question among the different domain ontologies and the second one is based on the technology provided by WP1, technology that supplies question classification based on the technology of TextGarden\(^2\).

\(^1\) All the processes made over a question input are also made over all stored question. These results are also stored in a relational database.
\(^2\) [http://kt.ijs.si/Dunja/textgarden](http://kt.ijs.si/Dunja/textgarden)
2.1.2 **Keyword/Synonym detection**

At this stage, the user question is word-like tokenized and system tries to match each token with each FAQ question token. Both “exact” and “synonymy” matching are tried. As a result of this stage, a narrower list of candidate FAQ items will be supplied. This might seem similar to other standard searching systems using keywords or metadata. Here, the difference lies on the use of morphological parsing of the user question that discards non-relevant words and the use of synonymy.

2.1.3 **Ontology/Semantic Distance**

This is the most time consuming stage, and therefore it is left to the final stage, when the FAQ target database has already been very reduced by previous searching stages. At this stage, grammatical patterns will be detected from user question. Then, these patterns are searched in the Ontology to build a graph path or trace. Finally, the system tries to match this user question ontology graph path with a reduced FAQ target subset graph paths previously calculated in the background using the semantic distance algorithms.
3 Scenario Description.

The initial use cases designed for the FAQ searching system are described in detail in [1]. This document describes the final version of these scenarios, including some related details of their implementation (screen captures, partial result of the phrase processing, etc).

The next use case UML diagram (Figure 3.1) depicts the most important cases, those have been implemented. Main introduced changes are related with iFAQ administrator’s tasks (shaded cases are new or updated cases introduced in the development phase).

3.1 User Search Scenarios

3.1.1 User question search scenario

Intention of scenario/setting: This is the most frequent and important use case found in our system. A system user types a question that he/she expects to find at a FAQ repository. The system would have to provide to the user the question stored at the database that has meaning more similar to the question input. The system will use technologies available in SEKT project to try to find out the best scoring FAQ item.

User expectations: User expects to find at the FAQ repository the question/answer that can answer his/her question that is the question stored at the FAQ repository that has a meaning more similar to the question input with a reasonable response time.

Pre-conditions: There are some functional considerations behind this software design:

- Exhaustive search can be very time consuming.
- Multistage searching system lets the system stop when FAQ target set has been reduced considerably. Therefore not all stages are exhausted and the computational cost is reduced and adapted to search features.
• Besides, multistage searching allows configuring the system to show the best score results to the user before completing all available searching stages, so users can decide to accept the result or continue with the remaining search stages.

**Figure 3.1: Legal Case Study Prototype implemented use cases**

**Post-conditions:** The system can process the user’s questions and it can return the best FAQ list related with the question and their scoring (the scoring is related with the matching degree between the question-input and the question stored).

**Alternative scenarios:** Related use cases are Overall search system, Ontology domain detection stage, Keyword matching stage and Ontology concept graph path matching stage. These use cases are described in detail in the next sections.

**Screen captures:** The next figures are application screen captures of the user's search. We can see the interface to ask the iFAQ system (Figure 3.2) and the related results screen (Figure 3.3).
Figure 3.2: User's interface to search FAQs.

Figure 3.3: FAQ search results screen
Overall search system

In Figure 3.4 we depict an overall description of multistage search system. This software architecture uses a Factory pattern to build a `FAQSearchEngine` implementation suitable for the ontology search purpose. In our case ontology based `FAQSearchEngine` is to be created, but other `FAQSearchEngine` could be used, if necessary. `FAQSearchEngine` will determine from configuration or on demand what search engine to use among the available ones.

Next, three stages will be processed in order to reduce our FAQ searching target set, with the compromise that the best score FAQ item related with the user question must be included in this target. This last point is very important because if previous searching stages mistakenly determine a FAQ target set, next searching stages will be unprofitable.

At each searching stage, the input will be a FAQ subset determined at a previous stage, and the output a reduced FAQ subset as an outcome of a narrower search. We postulate two assertions in our design:

- The best user question matching FAQ item must be included in the outcoming FAQ subset.
- Each stage is more restrictive than the previous one, that is, the FAQ result set is narrowing in each stage.

We have designed three searching stages that leverage on ontology and NLP technologies: ontology domain detection, keyword/synonym detection and Ontology concept graph path matching, as we have described at section 2.

In the picture there is also another technology we will use frequently in our design: a cache system for rapid accesses to frequently used data, such as, for example, the FAQ database. Performance is the main motivation for using this cache-based design. This searching system uses intensely certain data that is not necessary to recalculate more than once. But as cache memory is limited, a compromise between memory consumption and efficiency has to be managed.

As a result of a completed staged searching phase, a FAQ list ordered by score will be provided.
Figure 3.4: Overall search system UML sequence diagram
Ontology domain detection stage

The next sequence UML graph (Figure 3.6) depicts in detail the workflow followed at this stage. This is our first searching stage. Its goal is to reduce the FAQ database target of our searching system to improve performance in the other phases. The FAQ database can be very vast. Indeed, this FAQ database is composed of several smaller domain databases. Therefore, as the user question will likely belong to a knowledge domain, it is only necessary to search in that specific domain database, not in the rest. In this sequence diagram, the searching system is built over some important components that will often appear:

- **NLP Engine**: responsible of morphologic and grammatical parsing of questions.
- **Morphological Engine Adapter**: a helper class used by NLP Engine, specialized on morphological analysis. This adapter leverages on specialized external morphological analyzers supplied by other companies.
- **Thesaurus Engine Adapter**: a helper class used by NLP Engine, specialized on synonyms searching. This adapter leverages on specialized external thesaurus engines supplied by other software manufactures.

This design, based on adapters, allows us to replace those specialized helper classes with other alternatives provided that these new helper classes fulfil the adapter interface. So adapters allow us to use plug-ins for special tasks.

We have developed a mixed NLP and Thesaurus Engine, named *MixedNLP Engine*. It dissects the user question and detects all its relevant words, analyzing them morphologically. Also, the mixed engine detects the concepts, attributes or instances of the ontology that they are in this list of relevant words. If the relevant words do not have a correspondence with the ontology, this engine search tries to find the correspondence with the ontology using synonyms.

The next example shows the result generated by **MixedNLP Engine** when process a user phrase:

**USER PHRASE**
Ha **venido** por la **mañana** una **señora** que **quería** una **orden de protección**. Llevamos todo el **día** con el **tema**. Se la **acabo de acordar** y le estoy **notificando** la **orden de protección** y ya me **dice** que quiere **retirar** la **denuncia** y **no quiere** la **orden de protección**. ¿Qué hago?²

**MIXED-NLP PROCESSING RESULT**

<table>
<thead>
<tr>
<th>Word</th>
<th>Synonyms</th>
<th>Morphological</th>
<th>Ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>venido</td>
<td></td>
<td>• venir</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• venirse</td>
<td></td>
</tr>
<tr>
<td>mañana</td>
<td></td>
<td>• mañana</td>
<td></td>
</tr>
<tr>
<td>señora</td>
<td>Mujer</td>
<td>• señor</td>
<td>Class: <a href="http://legal/kb#Mujer">http://legal/kb#Mujer</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Attribute:</td>
</tr>
</tbody>
</table>

² One lady came this morning in order to get “protection warrant”. We have been busy the whole day with this subject. I’m giving her the “protection warrant” but now she tells that wants to retract the “denuncia” but she doesn’t want the “protection warrant” anymore. What shall I do?
For each task mentioned above, the engine leverages in other helper classes, those offer special functions. A morphological engine adapter processes each user question word, detecting the POS, gender, number, etc. of each meaning of the word, and all relevant words are filtered\(^4\). This task is very computational consuming so the most frequently processed words are cached in memory. Later accesses to that data are collected from the cache memory instead of morphologically processing the user question again.

Now, we have got all needed processed data from the user question to find out the best candidate of ontology domain fitted to our user question. Then, the next task is to

\(^4\) The applied filter entails considering as relevant words, verbs, nouns, adjectives, adverbs and pronouns
detect what is the domain of the question input. To make this task we can use two alternatives:

- If we have several domain ontologies, we can statistically detect all those concepts in each ontology domain. Now we are implementing a solution based on the number of occurrences of the some ontology term in the phrase. In the actual development, to access the ontology we develop and use an ontology API adapter (KPOntologyAdapter) now it is based in KPOntology API (iSOCO technology), but in the future we integrate an adapter to KAON 2 API.

  To avoid reiterate searches of the most frequent ontology nodes, these can be recovered from cache if the performance of the ontology searching system is not as expected. The outcome is an ontology domain candidate with the best statistical score.

  Once the ontology domain has been determined, the stage final outcome is built, consisting in the complete FAQ list of the candidate ontology domain.

  Another design aspect considered is that all data calculated at one stage must be kept over the complete multistage cycle, because it is likely to be reused again at the following stages. For example, the relevant words meanings of a user question are kept in cache for later use.

- The second approach is based on the domain detection using the technology of TextGarden. Currently, all stored question-answer are manually classified. This classification serves as an input to this software to learn about how it can make the classification, and also provides a new classification of this input. Besides the result of this process can be used in the domain detection of the input question.

  The next sequence diagram (Figure 3.7) shows an alternative solution based in technology of workpackage 1. We can see how the open architecture allows changing the stage implementation without collateral effects.
Figure 3.6: Ontology domain detection stage UML sequence diagram
The second searching stage (Figure 3.8) receives a FAQ list processed in the previous stage. The responsibility of this stage is to filter this input to reject non-matching FAQ subset, considering matching at keyword level. This phase punctuates each FAQ item with a score. All FAQ items with scores beneath a threshold will be eliminated from the candidate list.

The key concept at this phase, which differentiates it from other standard searching system, is the use of morphological meanings and synonyms in this keyword-matching algorithm. This avoids the use of exact or partial word matching. The use of only relevant words also reduces the emergence of many false candidates.

After processing or caching the relevant words of the user question and its synonyms using an NLPEngine (MixedNLPEngine), the searching stage pursues iterating over each input filtered FAQ item.

For each FAQ item, the following strategy is used to determine a matching score between user question and FAQ question: first, the relevant words of the FAQ question are recovered from cache (otherwise, they will be recovered from persistence RDBMS\(^5\)). Then, two inner loops iterate along all the relevant words of the user’s question and FAQ question. For each couple of both meanings, an exact matching is tried and if it fails, a synonymy matching is processed.

---

\(^{5}\) Relational database management system. Currently we use mysql.
Figure 3.8: Keyword matching stage UML sequence diagram.
Finally, all collected matched hits are processed to assign a final score to that FAQ item. As a result, we have got a list of punctuated FAQ items. Each FAQ item with a score below a threshold will be discarded.

**Ontology concept graph path matching stage**

This last stage is the most time consuming so it is important to start it with a very reduced input FAQ candidate list. The work made at this phase can be summarized roughly as follows:

- Parsing of the user question to detect grammatical patterns\(^6\).
- Identifying these grammatical patterns in the ontology, implying:
  - Searching for the concepts of the grammatical patterns.
  - Finding the paths that connect those concepts.
- Computing the minimum distance between grammatical pattern paths of user question and FAQ question, those imply computing the minimum distance between two nodes of the ontology.

The next sequence UML diagram (Figure 3.9) depicts this stage in detail.

After mixed NLP processing of user question, we have a related list of ontology nodes. Next step consists of searching grammatical patterns of these nodes in the domain ontology\(^7\). This is a complex process consisting of different steps, briefly enumerated in the following list:

- Searching of the ontology nodes that match any grammatical concept of the user’s question.
- Searching of the minimum distance path that connects each pair of consecutive concepts.
- Building a minimum path by connecting some or all the pattern concepts in the correct order.

The first task consists on locating the ontology nodes corresponding to each concept of the grammatical pattern in the correct order. The system can locate these nodes from the cache (most frequent used nodes), or by searching them with the help of an ontology API framework.

Next tasks build a connecting path as follows: from each node the system will find a minimum path to the next node, and so on. By doing so, the path that connects all nodes will be completed. Sometimes, it will not be possible to connect a pair of nodes, so the resulting path will not be fully connected.

---

\(^6\) A typical grammatical pattern is formed by a subject, a verb and some direct or indirect objects. Other patterns may be more or less complex. This syntactical analysis is useful to organize the concepts on the basis of the role they played in the sentence.

\(^7\) Currently these searching of grammatical patterns are based on the detection of the relevant words, being a grammatical pattern the sequence of relevant words. This has a temporally solution to avoid the lack of not to have a minimal Syntactical analyzer, that allows us to identify syntactical patterns like the noun phrase, the verb phrase etc.
Figure 3.9: Ontology concepts graph path-matching stage UML sequence diagram.
Now, we face the challenge of finding a minimum distance path between two nodes. A minimum distance path between two ontology concepts is that with the smallest accumulated weight, considering that between two connected nodes there is an edge with a slot weight assigned. We have opted for a Dijkstra’s algorithm variant. This variant is proposed in the dynamic programming paradigm using binary heaps that reduces the algorithm’s running time from $\Theta(n^2)$ to $\Theta(m + n \log n)$, where $n$ stands for number of nodes, and $m$ for the number of edges.

This algorithm uses some suitable data other representation structures, like adjacency matrix and adjacency lists, which improve its efficiency as well. Adjacency structures are previously calculated during the system initialization phase and are recovered from cache (in other words, we make these data persistent).

Minimum distance path between two ontology nodes is very time consuming. Therefore, for performance reasons it would be desirable to keep as many as possible in the cache, so they are previously calculated during the tuning phase and are recovered from the cache.

As a result of this step we get a list of ontology paths for each grammatical pattern detected on the user question. Also from cache we have got similar ontology paths for all FAQ grammatical patterns. Summarizing, we have acquired all data needed for the next step.

**FAQ Navigation by domain.**

**Intention of scenario/setting:** This scenario describes how a user navigates the FAQ repository by domain. The user selects this option in the menu, after that, the system shows a list of FAQs grouped by domain, and then the user can access to the content of them.

**User expectations:** User expects to navigate the FAQ repository by domain, so the user could access to a set of FAQ grouped by their domain.

**Pre-conditions:** Imported FAQs are classified by domain.

**Post-conditions:** FAQs report by domain.

**Alternative scenarios:** The “shows FAQ detail” scenario is included in current scenario, because when the user selects some FAQ detail, the system must execute “shows FAQ detail” scenario.

**Screen captures:** The next figure (Figure 3.10) is application screen captures of the user's FAQ navigation by domain.
3.2 System scenarios

3.2.1 Search system initialization use case

Intention of scenario/setting: Search system leverages on some preliminary calculations made at start-up or in background during normal working or with a schedule of low system activity.

When system starts up it must executed some initialization tasks. The following sequence's UML diagram (Figure 3.11) summarizes this task.
D10.3.1.1 / Prototype - Informal Deliverable

At start-up phase system uses the *FAQSearchFactory* to build all available *FAQSearchEngine*. After building these engines the system initializes them.

In the particular case of an *NLPOntologyFAQSearchEngine* (our particular engine we use in this prototype) the initialization consists of the following steps:

- Load all related ontologies (their ontology API adapters).
- Cache these objects in memory for later use.

Other needed data will be recovered from a persistence repository and stored in the cache memory when they were requested.

**User expectations:** The computation time associated with user’s search should be minimal.

**Pre-conditions:** Process search and data to cache correct.

**Post-conditions:** Ontologies are cached, so minimum computation time in user’s search.

### 3.3 Administrator scenarios

#### 3.3.1 Tuning iFAQ System use case

**Intention of scenario/setting:** This scenario and its alternative use cases (*New FAQ item acquisition use case Existing FAQ item updating use case and Ontology processing use case*) are considered for maintenance purposes (Figure 3.12). Normally a legal domain expert revises both the FAQ repository and domain ontology. As a consequence the FAQ searching system evolves with time, incorporating new FAQ items or updating some of them. Also, we have to consider the updating of the ontology(its modification or extension), because it implies the updating of the results of the initialization. this is the reason why this scenario need to be incorporated in the system. Then, the administrator uses this scenario to maintain the system updated.

**User expectations:** The system must be maintained updated.

**Pre-conditions:** Legal domain expert revises both the FAQ repository and domain ontology.

**Post-conditions:** The system is maintained updated.

**Alternative use cases:** *New FAQ item acquisition use case Existing FAQ item updating use case and Ontology processing use case.*
Figure 3.12: Tuning iFAQ System UML sequence diagram

New FAQ item acquisition use case

The case when a new FAQ item needs to be added to the repository can be seen in section 5 of deliverable D.10.2.1 [1].

Existing FAQ item updating use case

The case when a FAQ item needs to be updated can be seen in section 5 of deliverable D.10.2.1 [1].

Ontology processing use case

One of the most important considerations in the FAQ system searching design is the efficiency, and computation of the minimum distance path between two ontology nodes is very time consuming. Therefore, for performance reasons it would be desirable to keep as many as possible in the cache, so they are previously calculated during the tuning phase and are recovered from the cache, in the tuning phase (Figure 3.12).

The next UML sequence diagram (Figure 3.13) shows how ontology processing computes and stores all the distances between nodes of the ontology, using Dijkstra algorithm.
Ontology updating use case

Intention of scenario/setting: This scenario includes the previous scenario. In short, when a domain expert has modified the ontology, all FAQ items need to be preprocessed again to update their internal data structures and the ontologies need to be processed, so the system needs a new tuning.

Finally all recalculated data will be updated both in the repository database and cached memory.

User expectations: When a domain expert has modified the ontology, the system needs a new tuning.

Pre-conditions: A domain expert has modified the ontology.

Post-conditions: The system is maintained updated.

Alternative scenarios: Tuning iFAQ System use case

4 Architecture diagrams

This section presents the group of components, built up the whole search system. After that, we present some class diagrams that depict the main classes and their relationships for each one. Finally, the last sub-section includes a deployment diagram, to show how the components are distributed in hardware architecture.

4.1 Component diagram

The main subsystems are (see Figure 4.1):

- NLPSubsystem.
- OntologySubsystem.
- Business Logic, which includes different search engines.

Our search system can offer its services to its clients as long as they send a request. Among them, we enumerate the following clients:
• Webservice: the client web service connects to our system to request some search services.
• Web client: it manages as the other examples.

![Component Diagram](image)

**Figure 4.1: Component Diagram**

### 4.2 Classes diagrams

As another result of our technological design, we have built some class diagrams that depict the main classes and their relationships of components that we can see in previous diagram (Figure 4.1).
4.2.1 Administration Class Diagram

Figure 4.2: Administration class diagram

4.2.2 FAQ and Domain class diagram.

Figure 4.3: FAQ and Domain class diagram
4.2.3 Search class diagram

Figure 4.4: Search class diagram
4.2.4 NLP class diagram

Figure 4.5: NLP class diagram
4.2.5 Ontology class diagram

Figure 4.6: Ontology class diagram
Considerations of scalability, efficiency and memory limitation have let us to consider a distributed architecture with different nodes to process the user searching request:

- One or more nodes for NLP processing.
- One or more nodes for Ontology processing.
- One node to gather user requests as a main dispatcher.

All nodes exhibit their services with web services public interface. All connections between nodes use that architecture. This allows us to improve the system efficiency in terms of global memory and processing to fit itself to increasing demand of services. It is important to note that NLP and especially ontology processing tend to consume a lot of memory.

Next diagram (Figure 4.8) depicts this architecture.
4.4 SEKT Technology integration diagram

Figure 4.9: SEKT Technology integration diagram
5 **Ontology of Professional Judicial Knowledge.**

As we describe on section 2.1.3, to find the best match between the question stored in the repository and the question made by the user, we use in the final stage an ontology to obtain the semantic distance. The ontology that we use to make this process is the ontology described at [1], OPJK “Ontology of Professional Judicial Knowledge”. This ontology has been developed by the legal case study team and it is learnt from scratch out of the competency questions posed by the judges during their interviews. Modelling this professional judicial knowledge demands the description of this knowledge as it is perceived by the judge and the abandonment of dogmatic legal categorizations.

The Ontology of Professional Judicial Knowledge has been extracted from the selection of relevant terms from nearly 200 competency questions and has, currently, nearly 50 concepts, 100 relations and more than 300 instances. This is result of a choice to minimize the concepts at the class level when possible in favour of creating instances and relations.

The ontology used at the prototype described at this document, is an ontology modified with some changes resultant of the integration of the PROTON ontology. At Figure 5.1: OPJK we can see the actual hierarchy of the OPJK concepts.
6 Measurements.

At [2] some measurements to apply to the Legal Case Study were defined. These measurements can be summarized at Table 6.1:

<table>
<thead>
<tr>
<th>Measure</th>
<th>Definition</th>
<th>Target</th>
<th>Reported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search-engine usage</td>
<td>Number of queries per day</td>
<td>Continuing growth</td>
<td>At the end of the project</td>
</tr>
<tr>
<td>Average Response time</td>
<td>Average time to respond to a user’s query</td>
<td>Not target at this stage</td>
<td>At the end of the project</td>
</tr>
<tr>
<td>Effectiveness of search</td>
<td>User selection of ranked answers (precision and recall)</td>
<td>Not applicable</td>
<td>At the end of the project</td>
</tr>
</tbody>
</table>

Table 6.1: Measurements.

The measures applied in the case studies will reflect the usage of the SEKT technology as implemented in the 3 case studies. These measures will be evaluated at the end of the project, i.e. during the last quarter of year 3.
In order to evaluate the current prototype, we plan to validate the improvement of introducing the concept of the semantic distance to find the question-answer that better match with the question-input. This evaluation will consist on the comparison of results obtained using a typical keyword based search engine and the results obtained on the application of keyword search in combination with ontology based search. To facilitate this task, we will use an audit system (integrated as part of the system) and also the multistage searching approach followed to design the case study architecture to switch on the respective stage.

7 Next Steps.

In this section we will describe which the next steps are related with the prototype:

- Evaluation of the results obtained of the comparative between the two kinds of search, keyword search and ontology and keyword search combination.
- Integration of the current ontology in the last version of PROTON. The last changes on PROTON include some necessities related with the ontology of this case study.
- Integration with SIP, first steps are made in this direction. Some WP technologies will be used at this prototype, an example of this is the integration of the domain ontology detection using the technology of the WP1. Also other technologies related with the natural language processing will be integrated. Therefore the major effort in integration of technologies from other workpackages will be done on the development of the second part of the case study, the jurisprudence technology.
- We plan to integrate on December 2005 the Search & Browse technology (WP5) to find what the judgements associated to the FAQ are obtained as a result of the FAQ system. Also we will use some results of wp1 technology related with the crawling of judgments.
- Finally, some improvements have been identified as a result of the computation of the paths between the concepts of the ontology, and the match with the question made by the user. Some of these improvements are aimed to the accuracy and performance. Several algorithms can be used, currently we are studying an alternative based on A* algorithm that can be very appropriate for our purpose and very efficient computationally, but it implies transforming the ontology graph in binary tree.

8 Conclusions.

In this deliverable, we have described the current status of the Legal Case Study prototype. It has the following main features:

- It is designed to be accurate and technologically advanced by using NLP and Semantic Web techniques
- It is designed to be efficient, extensible, customizable and scalable.
- It makes use of incremental searches as a process to narrow the desired FAQ set.
- It uses a variety of pluggable search algorithms.
9 References.
10 Figure List.

Figure 1.1: Architecture of the Legal Case Study System ........................................ 5
Figure 2.1: Multistage adaptive searching system ...................................................... 7
Figure 2.2: Ontology Domain Stage Approach .......................................................... 8
Figure 2.3: Keyword/Synonym Stage Approach .......................................................... 8
Figure 2.4: Ontology/Semantic Distance Stage Approach ........................................... 9
Figure 3.1: Legal Case Study Prototype implemented use cases ................................ 10
Figure 3.2: User's interface to search FAQs .............................................................. 11
Figure 3.3: FAQ search results screen ................................................................. 11
Figure 3.4: Overall search system UML sequence diagram ...................................... 13
Figure 3.5: MixedNLP processing example .............................................................. 15
Figure 3.6: Ontology domain detection stage UML sequence diagram .................. 17
Figure 3.7: Ontology domain detection stage based in WP1 technology, UML sequence diagram ................................................................. 18
Figure 3.8: Keyword matching stage UML sequence diagram .................................. 19
Figure 3.10: User's FAQ navigation by domain, screen capture .................................. 21
Figure 3.11: Search system initialization use case UML sequence diagram .............. 23
Figure 3.12: Tuning iFAQ System UML sequence diagram ..................................... 23
Figure 3.13: Ontology processing UML sequence diagram ...................................... 26
Figure 4.1: Component Diagram .......................................................................... 27
Figure 4.2: Administration class diagram .............................................................. 28
Figure 4.3: FAQ and Domain class diagram ............................................................ 28
Figure 4.4: Search class diagram ............................................................................. 29
Figure 4.5: NLP class diagram ................................................................................. 30
Figure 4.6: Ontology class diagram ........................................................................ 31
Figure 4.7: Persistence class diagram ...................................................................... 32
Figure 4.8: Deployment Diagram .......................................................................... 33
Figure 4.9: SEKT Technology integration diagram .................................................. 33
Figure 5.1: OPJK ........................................................................................................ 35