FAKULTÄT FÜR !NFORMATIK

Web Scraping
A Tool Evaluation

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering & Internet Computing

ausgeführt von

Andreas Mehlführer, Bakk. techn.
Matrikelnummer 0027418

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao. Univ.-Prof. Mag. Dr. Christian Huemer

Wien, 23.03.2009

_____ (Unterschrift Verfasser)

_____ (Unterschrift Betreuer)

Eidesstattliche Erklärung

Andreas Mehlführer, Liesingbachstr. 214, 1100 Wien

"Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit − einschließlich Tabellen, Karten und Abbildungen −, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe."

Wien, 23.02.2009

Andreas Mehlführer

Acknowledgments

Working on these thesis topics together with bwin has been a great pleasure. I would like to thank Bernd Kretzel for the good co-operation and for giving advice and feedback.
Furthermore I would like to thank the following people for their help and input: Michael Sknura, Harald Göschl, Manuel Berger, Bence Haraszti, Peter Klimo, Doris Kokot, Andreas Nagl and Vassil Nikolov.
I would also like to thank Christian Huemer for supervising my diploma thesis.
Last but not least, I want to thank Margit and my parents for supporting me in my studies.

Abstract

The WWW grows to one of the most important communication and information medium. Companies can use the Internet for various tasks. One possible application area is the information acquisition. As the Internet provides such a huge amount of information it is necessary to distinguish between relevant and irrelevant data. Web scrapers can be used to gather defined content from the Internet. A lot of scrapers and crawlers are available. Hence we decided to accomplish a case study with a company. We analyze available programs which are applicable in this domain.

The company is the leading provider of online gaming entertainment. They offer sports betting, poker, casino games, soft games and skill games. For the sports betting platform they use data about events (fixtures/results) which are partly supplied by extern feed providers. Accordingly, there is a dependency on providers. Another problem is that smaller sports and minor leagues are not covered by the providers. This approach requires cross checks which are manual checks with websites if provider data differs and the definition of a primary source (source which is used in case of different data from providers) have to be done by data input user. Data about fixtures and results should be delivered by a company owned application.

This application should be a domain-specific web crawler. It gathers information from well defined sources. This means bwin will not depend on other providers and they can cover more leagues. The coverage of the data feed integration tool will increase. Furthermore, it eliminates the cross checks between different sources. The first aim of this master thesis is to compose a functional specification for the usage of robots to gather event data via Internet and integrate the gathered information into the existing systems. Eight selected web scrapers will be evaluated and checked based on a benchmark catalogue which is created according to the functional specification. The catalogue and the selection are conceived to be reused for further projects of the company. The evaluation should result in a recommendation which web scraper fits best the requirements of the given domain.

Kurzfassung

Das WWW wurde zu einem der wichtigsten Kommunikations- und Informationsmedium. Firmen können das Internet für die verschiedensten Anwendungen einsetzen. Ein mögliches Einsatzgebiet ist die Informationsbeschaffung. Da das Internet aber eine so große Menge an Information anbietet, ist es notwendig zwischen relevant und irrelevant zu unterscheiden. Web Scraper können verwendet werden um nur bestimmte Inhalte aus dem Internet zu sammeln. Es gibt sehr viele verschiedene Scraper daher haben wir uns entschieden ein Case Study mit einem Unternehmen durchzuführen. Wir haben verschiedene Programme analysiert, die in diesem Bereich einsetzbar wären.

Das Unternehmen ist ein führender Anbieter von online Spielunterhaltung. Sie bieten Sportwetten, Poker, Casinospiele, Soft- und Skillgames an. Für die Sportwettenplattform werden unter anderem Ereignissdaten von externen Feedprovidern verwendet. Daraus resultiert eine starke Abhängigkeit von diesen Providern. Ein weiteres Problem ist, dass weniger populäre Sportarten und Unterligen von diesen Providern größtenteils nicht zur Verfügung gestellt werden. Durch Dateninkonsistenzen der einzelnen Feedprovider sind aufwändige, händische Überprüfungen notwendig. Die Daten sollten idealerweise von einem unternehmensinternen Programm bereitgestellt werden, so das die Abhängigkeit gelöst werden kann.

Eine solche Möglichkeit bietet der Einsatz eines bereichsspezifischen Web Scraper. Dieser durchsucht das Internet automatisiert nach bestimmten Informationen. Für das Unternehmen würde das also bedeuten, dass sie eine größere Abdeckung der Ligen anbieten könnten und nicht mehr von anderen Providern abhängig sind. Das Datenteam soll sich so die Zeit für die manuellen Überprüfungen einsparen. Das erste Ziel der Diplomarbeit ist es, eine funktionelle Spezifikation zu erstellen für die Verwendung eines Web Crawlers. Acht Web Crawler werden dann ausgewählt und weiter bewertet. Die Bewertung erfolgt anhand des auf die funktionale Spezifikation aufbauenden Benchmark Katalogs. Der Katalog ist so konzipiert, dass er auch für weitere Projekte im Unternehmen verwendet werden kann. Die Evaluierung liefert als Ergebnis eine Empfehlung, welcher Web Crawler sich am besten für den Einsatz in diesem Bereich eignet.

Contents

1 Motivation

Today companies sometimes require information which already can be found on web pages. If just some pieces of information are needed, the search can be done manually. But when a lot of information is required, tools like web crawler are needed to provide the information in an automated way.
Currently there are different information extracting tools:

Kobayashi and Takeda defined a web crawler as follows:

> *A web crawler is a program or automated script which browses the World Wide Web in a methodical, automated manner. Other less frequently used names for Web crawlers are ants, automatic indexers, bots and worms* (Kobayashi and Takeda, 2000).

Adams and McCrindle defined a web scraper as follows:

> *A more recent variant of the web crawler is the web scraper, which looks for certain kinds of information - prices of particular goods from various online stores for instance - and then aggregates it into new web pages, often using a database.* (Adams and McCrindle, 2007 [AM07])

Pek defined a wrapper as follows:

> *A wrapper (also referred to as web wrapper) is an information extraction software program. Its goal is to transform data items, extracted from a semi-structured document (for example, an HTML document), into a self described representation (for example, a XML document) that then can be used by a variety of database-oriented applications.* (Pek, 2003 [PLL03])

The main goals of this work are the creation of a technical analysis and gather know how about these tools. Therefore i defined a process shown in figure 1.1.

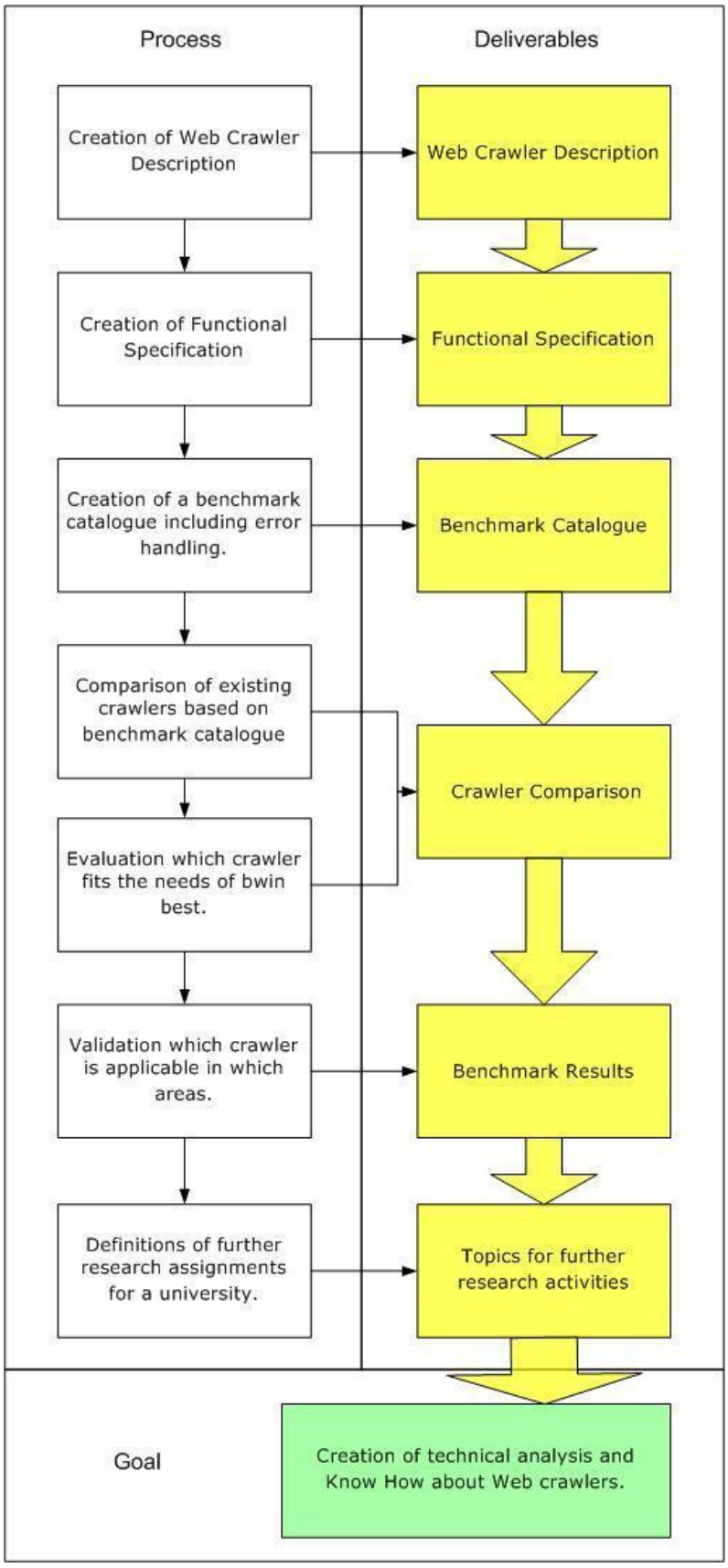


Figure 1.1: Process, deliverables and goal

The deliverables of this diploma thesis are:

- Web crawler description: A general introduction into web crawlers topic.
- Functional specification: A list of desirable functions and features that a web scraper should have to gather information from a site.
- Benchmark catalogue: The benchmark catalogue is based on the functional specification.
- Benchmark results: A web scraper evaluation that analyses how the crawler provides the function and features based on the benchmark catalogue.
- Crawler Comparison: The web scrapers are compared based on the benchmark results.
- Research assignments

Accordingly, the thesis is structured as follows:
Finding information in the World Wide Web (WWW) deals with understanding how it is built up and how retrieving information works. The following section provides a short introduction about the World Wide Web as well as into search engines and web crawler. Finally an introduction into the usage of spiders is given. Chapter 3 contains information about the evaluation process and the criteria. The results are described in chapter 4. In chapter 5 the tools are compared and strength and weaknesses are outlined. Finally, conclusion and further work can be found in chapter 6. The appendix (chapter 7) contains the functional specifications and the detailed rating of each tool.

2 Web Crawler and Web Scraper

2.1 The World Wide Web

The World Wide Web is one of the youngest services of the internet. In 1990 Tim Berners-Lee and his colleges started an initiative to make the internet usable for the exchange of information between scientists. The aim was to enable the upload of formatted documents with graphics. The main idea was to set links to documents even if they are on different servers. This should be realized via hypertext [MN01].

Nowadays, the World Wide Web is still growing faster and faster. Everyday there are thousands of new pages with lots of new information. The question of the availability does not exist anymore. The dynamic and unstructured nature of the available data causes that the users have an insuperable problem to find the relevant information they are looking for. Naisbitt brought this phenomenon with the following statement [Nai82]:

We drown in a flood of information but we hunger for knowledge.

A consequence of this problem of the WWW was the establishment of search engines and web directories. Currently the biggest existing search engines are google (www.google.com) and Yahoo!Search (www.yahoo.com). They use the key-words-method for search queries done by users. Such a user requested search query supplies a relevance ranking list. Search engines allow the users to find information much easier.

As mentioned above the second important technology to structure the web is the web directories. Some of these directories are Open Directory Project (www.dmoz.org) and Yahoo! (www.yahoo.com). They create manual assessments and classifications of websites which assemble a hierarchic structure of topics. This allows users a comfortable search [Bra01].

The WWW can be characterized as a huge amount of documents, which own an explicit defined address called Unified Resource Locator (URL). Most of them are used for websites. A website itself could contain other documents which could be sites again. Other components of a website are links which compose a network of many sites. If it is impossible to use any search tool it just would be possible to visit an initial page which address is know and all linked pages could be opened.

Sites of the WWW may contain various types of information. Mostly they are images, texts, videos, and sounds. The display of these contents is defined by a description language called Hypertext Markup Language (HTML). The WWW can be depicted as a directed graph shown in the following figure. The sites are the nodes and the links are appointed as directed edges. This graph can contain loops, cycles and do not have to be interrelated [Bre98].

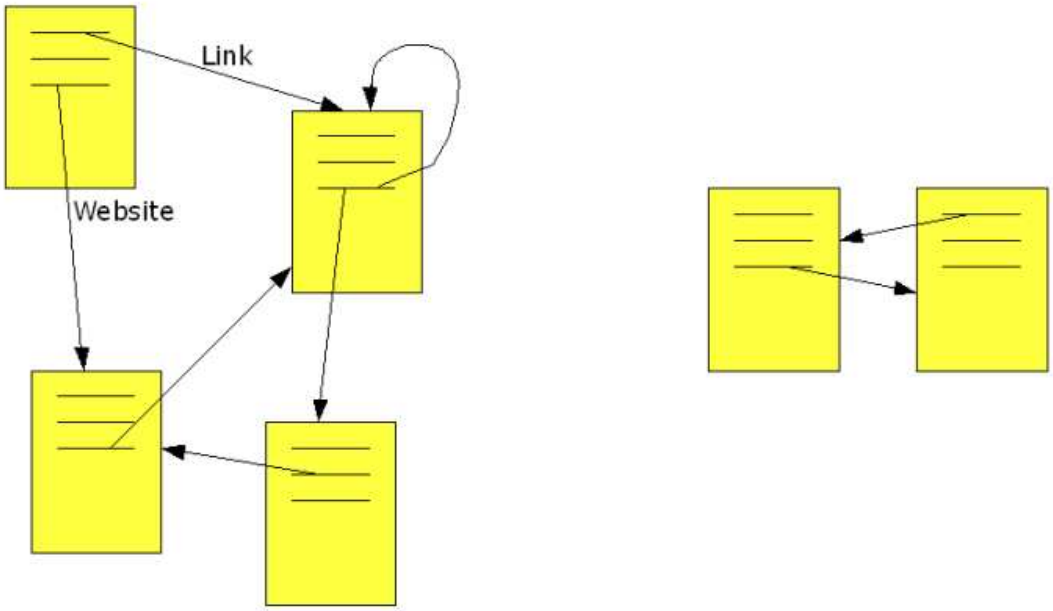


Figure 2.1: WWW as a graph [Bre98]

Due to the consistence of the WWW, crawlers have developed different strategies to crawl. In its simplest form a crawler does not check if a page has changed or not. In this case the spider crawls the page again but without extracting any new information. This leads to unnecessary burden of the servers. Some more complicated crawlers are intelligent enough to recognize that a site has not changed since the last visit. But they cannot diagnose if a linked page has changed. For example, as described in figure 2.2, if page A has not changed the site would not be crawled again but page B-I shown in the next figure may be changed and also would not be crawled [CBT05].

Some pages cannot be crawled. This is called the deep web. This can have various reasons. Some pages are just available with a previous authorization so a crawler cannot access these pages. Some dynamic pages cannot be crawled because they need special input information like query terms. A crawler is not able to find sites which are not linked to any other sites as mentioned above. This is demonstarted by the pages H and I in the next figure [CBT05].

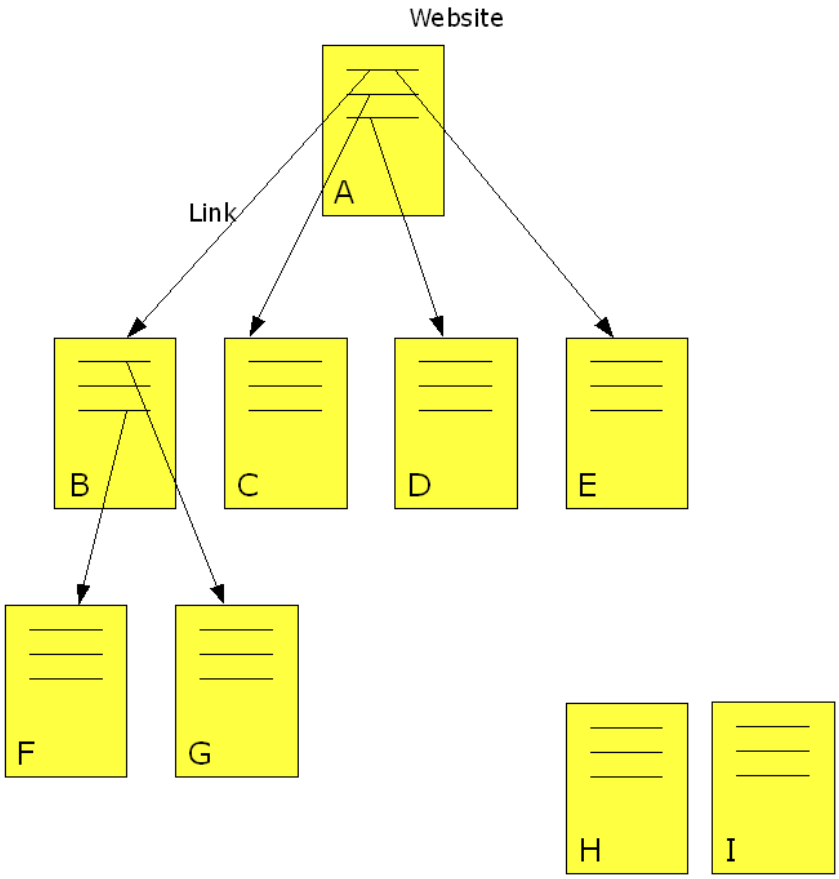


Figure 2.2: A page and its internal links [CBT05]

The WWW consists of static and dynamic web pages. A static page contains the same information on each download. The content of dynamic page is generated just in the moment of the request. The web can be divided in private and public areas. Private pages are often password protected or require other authentication mechanism. To crawl as much as possible dynamic pages it is necessary to use domain specific parameters. This is shown in the table 2.1.

	Static Pages	Dynamic Pages	
		Parameters known or not required	Parameters unknown
Private	Password or authorization required		
Public	Crawlable		Domain-specific knowledge required

Table 2.1: Different parts of the web [MN01]

2.2 Overview of Search Engines

The main goal of web search engines is to help the users finding information which is offered on websites. Each search engine executes three main steps shown in the figure 2.3.

1. The web crawler browses the web systematically.
2. With the assistance of the websites the crawler generates an index with the words contained on the websites.
3. In the last step the crawler allows users to search for words in this index [Bec07].

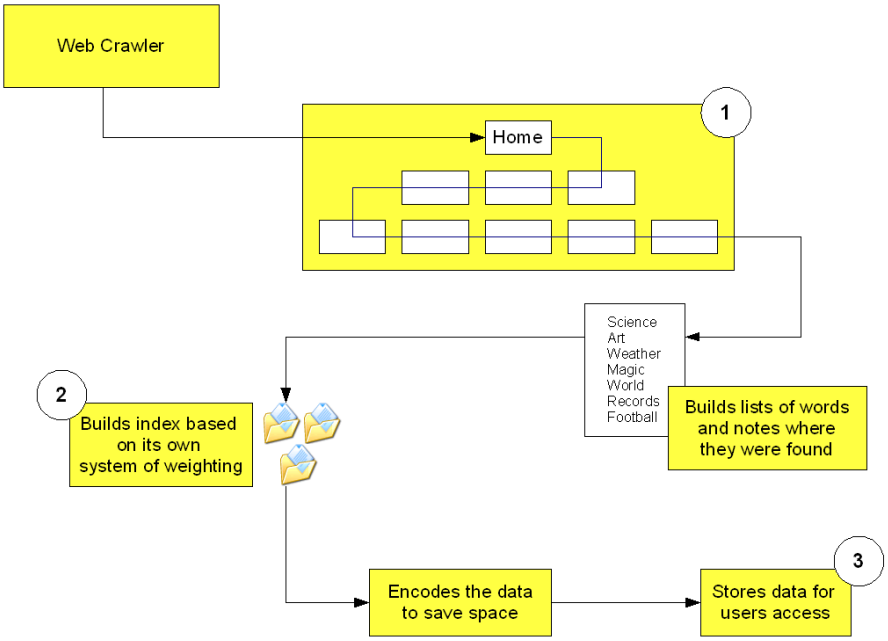


Figure 2.3: Search engines [Bec07]

2.3 Overview and Usage of Web Crawlers

Schrenk defined a web crawler as follows:

> *Spiders, also known as web spiders, crawlers, and web walkers, are specialized webbots that - unlike traditional webbots with welldefined targets - download multiple web pages across multiple websites.* (Michael Schrenk, 2007 [Sch07])

As mentioned before, literatures often use spiders and bots as synonyms for crawlers. Spiders usually work together with scrapers. The difference is that spiders follow the traces of links and scrapers pull contents from web pages. Often a web crawler contains a scraper [HC03].
Robots work continuous. For example the robot of AltaVista visits 10 million pages per day. That makes 7000 pages per minute. Imagine it is necessary to visit 7000 pages in one minute. That is a lot. If a site is listed in the result list of a search engine it is rather assured that some time ago a robot visited this page. The robot followed all linked pages and stored them in a database. If a site is very huge the crawler spread its work over more days that the traffic for the target server is not to high. Each time the crawler visits the site it copies some pages until it has the complete website [Bra01].

A crawler allows automating the accesses to resources of the WWW (World Wide Web). This enables collecting relevant information and presenting it in a user defined format. Spiders have the ability to combine different functions of different sites. A bot learns how to get specific information by filling out forms and searching in the results. They also can exam links if they are broken [HC03]. The most important usage of a web crawler is creating indexes for web search engines [Bra01]. Search engines need a crawler because they have to download the web pages to create keyword indexes inoder to reply fast to queries of users. Also price comparison tools need web scrapers to download price information from different web merchants and to compare them. Another application area of a web crawler is the retrieval of copyrighted material in the WWW. If a crawler finds such information, a Whois lookup of the domain starts to find who owns it. Another usage is that companies archive the sites of competitors. Web crawler are also used to harvest email addresses, mostly to sell them to spammers. Although the crawler opens a page or Usenet and extracts just the email addresses from the page. This subspecies is called Email-harvester. There are a lot of different ways to use web crawlers in ever day business and which makes them so important [Sch07].

2.4 Web Crawler Architecture

The simplest form of a crawler starts from a specified initial page outlined in figure 2.4. Based on this initial URL the crawler "opens" the corresponding page and scans it for other links and stores them in "to visit URL" if it is not already in "visited URL". The URLs which the crawler has already visited are stored in "visited URL". The initial page is stored in the database or anywhere else. Then the crawler visits the next URL from the "to visit"-list. Where it again stores and looks for other URLs [Cho02].

A web crawler does not move around to different computers as viruses do. A crawler stays on the same machine. The crawler just sends requested information or documents to other computers via the Internet. This is analogical to what web browsers do. Web crawlers follow links in an automated way [HC03].

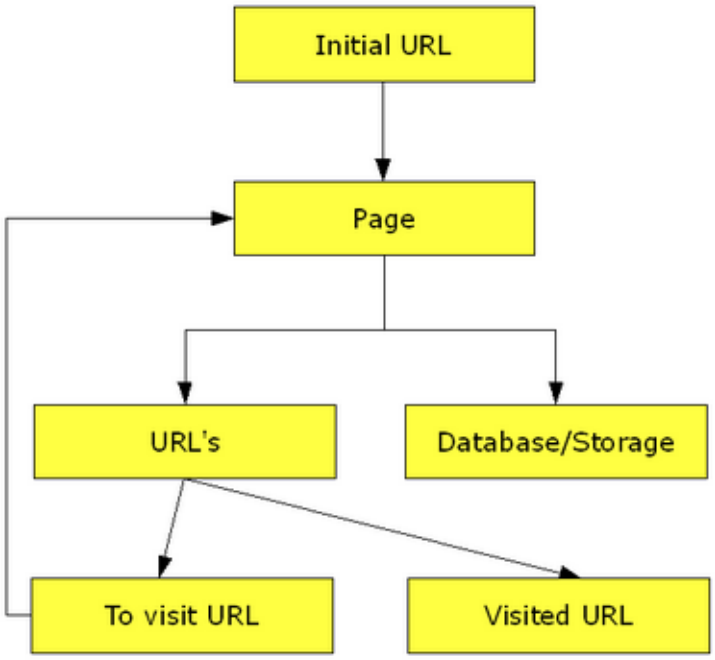


Figure 2.4: Simple architecture of a crawler [Cho02]

The crawler starts its search with an initial URL. Based on this URL it opens the page and extracts all linked URLs. All visited URLs are stored in a data structure (Visited URL). This grants that URLs are not visited again in this crawling act. The URLs which have to be visited are stored in a second data structure (To visit URL). This structure is the base for the further operation where URLs are visited and stored as "visited" and new URLs are added in the "To visit" list. The information of a page is stored in a third structure often a

document [Cho02]. It depends on the crawler, if it stores the complete site or parts of the site. The information can be stored in a simple folder or a database. In chapter 2.6 we present an example how information is extracted and stored.

The following figure 2.5 shows an example how a crawler works. All visited pages are saved in a defined store.

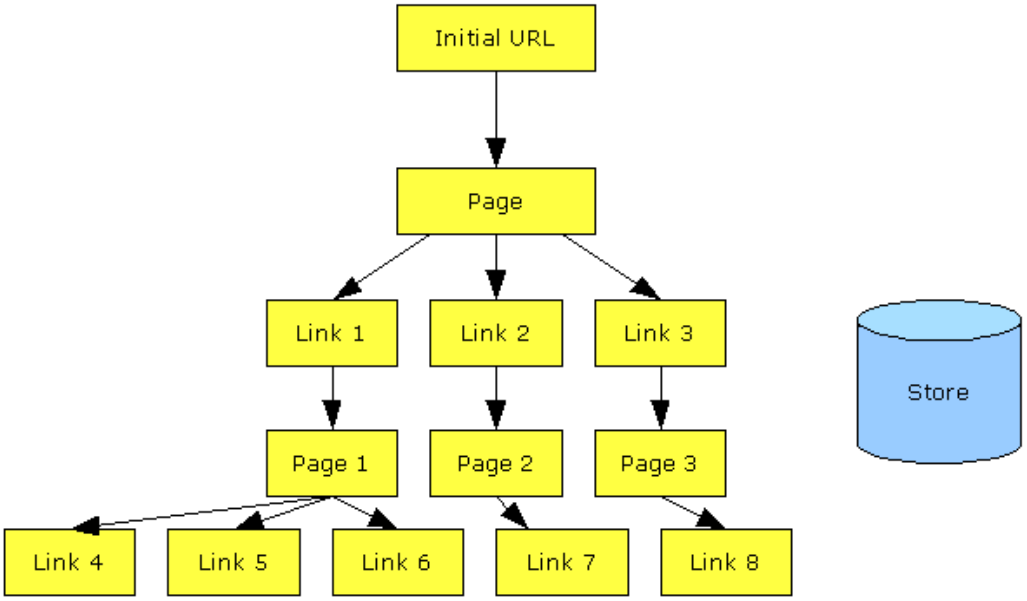


Figure 2.5: Sample of a crawling process [Cho02]

There are a lot of different algorithms defining the order of visiting the URLs. There are two popular methods: Breadth-first search and Best-first Search. Breadth-first is the simplest way for the crawling process. The URLs are visited in the order they were gathered and no special heuristics are used to decide which URL is the next one shown in figure 2.6 [Tru06].

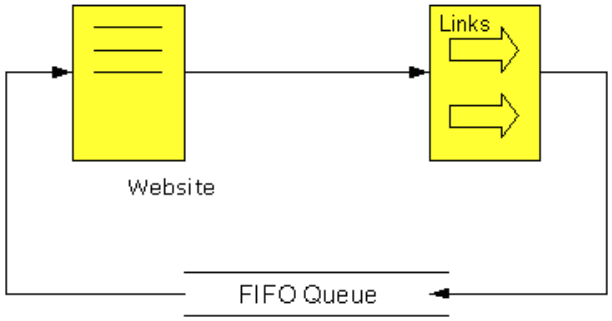


Figure 2.6: Breadth-First crawler [Tru06]

Best-first is currently the most popular search algorithm for focused crawlers. Founded URLs are sorted to a priority queue by special heuristics. The classification is done by a link estimator. URLs with a higher information concentration are visited earlier. Pages where little information is expected are placed at the end of the queue shown in figure 2.7 [Tru06].

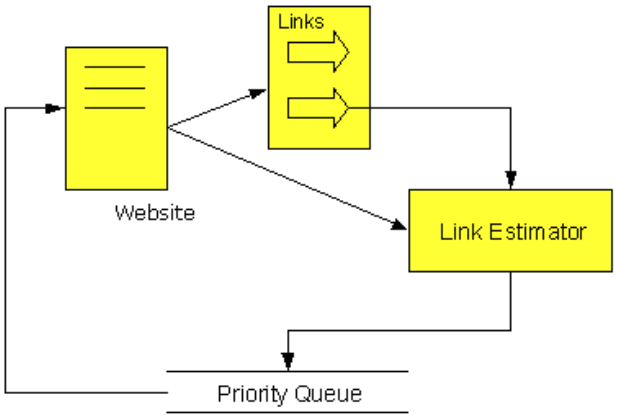


Figure 2.7: Best-First crawler [Tru06]

Best-first search has a real advantage in comparison to breadth-first because the important pages are visited first. This may be also a disadvantage if pages are classified not correctly. In this case important information are crawled at the end of the process or are even totally ignored [Tru06].

2.5 Limit Crawling Depth

A lot of links are detected during the crawling process of a spider. These links are stored in a queue. They are visited after the running process. This process can be done as long as the resources allow it. The configuration defines where a crawler starts its work. This is called the initial URL. Starting at this point the spider can accomplish all files, documents, and pages which are joined with the initial URL. This can be in a direct or indirect way. If a crawler would have enough resources, it could crawl most parts of the web, but this is not realistic. So it is important to confine the crawling field. This means the crawler works in a defined area. This is done by rules, which allow specifying which URLs are visited and which are ignored [Sch07].

Some rules limiting the crawling depth are:

- Limited number of initial URLs to start
- MIME types of documents which should be excluded from the process
- The maximum depth of directories on web pages
- Internet Protocol (IP) addresses and domain restrictions

The following figure 2.8 shows how a crawler with such a limited penetration level works.

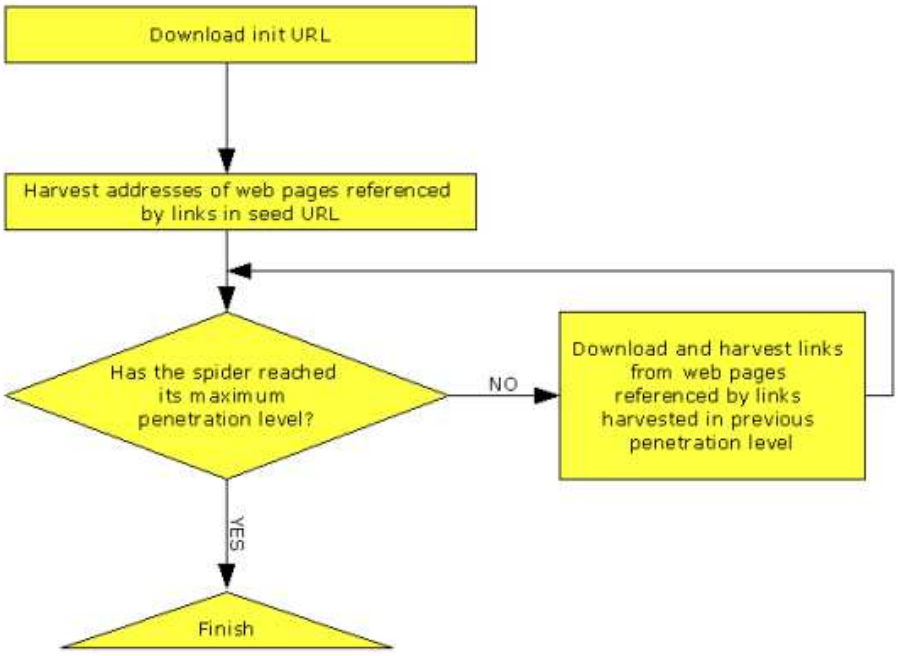


Figure 2.8: Crawler and penetration level [Sch07]

2.6 Scraping Information from the WWW

The World Wide Web has grown to a significant source of information. Most of the information is supplied as unstructured text [Pop07]. One of the reasons is that humans like to read information with a lot of layout-tags and need a menu to orientate on a website. Languages like HTML, XML (=Extensible Markup Language), ... allow us to create such user-friendly sites. On the other hand computers are not humans. If a program wants to integrate information from a website, it has to filter out the "useless" stuff like HTML-code first [HC03].

In general HTML files are less structured and more aimed to present information than to provide raw data. On the other hand XML and XHTML (=Extensible HyperText Markup Language) files have a good composition and are easy to handle for spiders. Most sites of the web are in HTML format. Some of these sites provide a "text only" or "print preview" variant, which could be used much better than normal HTML sites because the structure is less complicated [HC03].

Data extraction means to export information out of bad or unstructured sources and store it in a more structured way. Unstructured sources can be web pages, e-mails and as well data of measuring devices. A good structure is needed for the storage and further data processing. Often the extracted information has to be transformed.

Today, quite a lot of researchers are working on extracting information about types of events, entities or relationships from textual data. Information extraction is used for search engines, news libraries, manuals, domain-specific text or dictionaries. In the last years the web got quite popular as a source of information. Researchers do not focus only on structured and semi-structured text. Nowadays, they are also working on extracting information from unstructured web texts. These may be, for example, newsgroup postings, how-to sites, product and hotel reviews and many other types of textual data [Pop07].

A more complicated form of data extraction is text mining. This is used to scrap relevant information out of text files. Therefore complex linguistic and statistic algorithms are needed. Text mining is used to make text archives more efficient by automate the combination of relevant information. Text mining can be compared with data mining. Data mining is the systematic appliance

of statistic methods to match patterns. It is used to evaluate and interpret huge databases. Data extraction in contrast is used to create such huge databases.

The first barrier of information extracting is finding the desired information in an entanglement of structures. Another problem is that if a website changes the structure, the crawler usually is no longer able to gather information from this site. In this case the crawler has to be adapted to the new site. In general the structure of large websites will not be changed more often than every six months [Sch07].

Advantages of unstructured web texts:

- Redundancy: It is possible to find the same information provided from different pages. This allows comparing information from various sources and assuring the correctness.
- Broad coverage: The web contains a lot of information from products, hotels, restaurants to sport results.
- Search Engines: They provide a simple way to navigate through the web to find information [Pop07].

Disadvantages of unstructured web texts:

- Unreliable information: It can happen that a crawler uses unreliable sources which lead to wrong information.
- Ungrammatical language: The same information can be found on different pages but there could be used different spellings. For example different abbreviations for football clubs. This makes it difficult to parse the information [Pop07].

Wrappers are special procedures to extract information out of structured sources like databases. We can say that wrappers are classical data extraction tools. Another possibility is a scraper. A scraper allows the extraction out of unstructured sources. Scrapers are focused to transform unstructured data and save them in structured data bases. Screen scraping is sub version of scraping. Thereby a program extracts information from the display output of another program. The output which is scraped was created for the end user and not for other programs that is the difference to a normal scraper. The next form of a scraper is a web scraper. As the name indicates the sources are web pages.

There are many ways to scrap information. The best method is human copy-paste because barriers to prevent machine automation are not effective against humans. Sometimes this is the only way to export information from a web page. As everybody knows this is an exhausting work this is just a solution for small projects. For bigger company projects it would be too expensive. Another method is text grepping with regular expressions. Regular expressions are used to find information which match patterns. HTTP programming is also a possibility to extract information. Therefore HTTP request are sent to the web server. Further possibilities are HTTP programming, DOM parsing and HTML parsers. Mostly web scraper software is used so that it is quite easy to use.

Above we mentioned databases to store scraped information. Another possibility is to make scraper sites. Scraper sites are automatically generated from other web pages by scraping their content. A scraper is able to take information from tags and text between tags. So URLs of images and links can be scraped. For example, an image is enclosed by the <img> tags and the URL can be scraped as well as the image.

A problem is that it is not possible to assure that the scraped data is the desired one. Therefore schemes can be defined with regular expressions to validate the dates. Nevertheless there is no way to assure the correctness of scraped information. An example: A telephone number needs to be a combination of figures from 0 to 9. A regular expression can assure this. If the number is not correct because of transposed figures the scraper is not able to identify this.
The next part gives a short example of information extracting with Lixto. Lixto is a technology suite to extract information from the WWW with the help of a wrapper. Information embedded in an HTML page can be extracted and transformed in XML with a wrapper.

The most important component of a wrapper is the pattern. A pattern is a sort of a container for similar information. Example: *Date, TeamA, TeamB, FinalScore*. An own wrapper has to be assembled for each one web page because each has a different layout.

I used Lixto and the website www.soccerway.com to create an example how a scraper gathers information from a website. XPATH is used to locate the desired pattern. The goal is to extracted sport data of the 7th december 2007 and convert it to a more proper standard. The example at the next page shows one game

result per row. The table columns contain the different elements like *TeamA*, *TeamB* or others. A drag and drop filter specifies the elements and allows the extraction [BFG07].

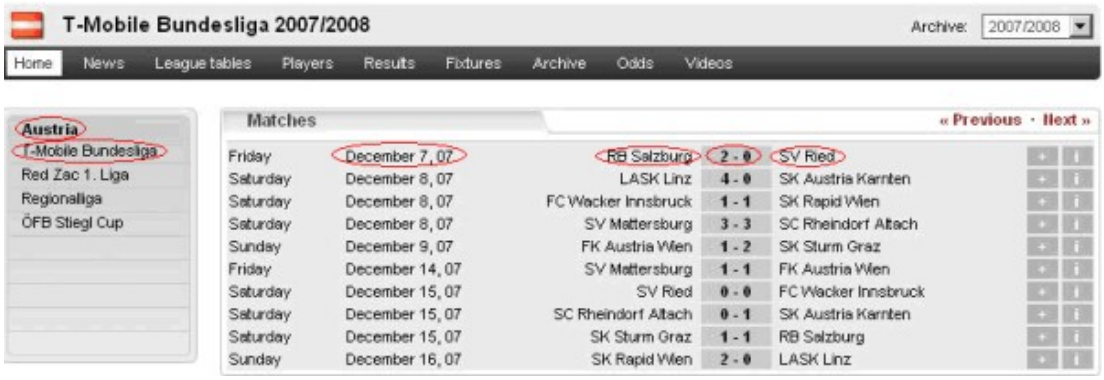


Figure 2.9: Information extraction with soccerway.com and Lixto

The Lixto VD creates a customized XML file. The date of the sport event was converted from *December 7, 07* to a more proper standard as *12/07/07*. This was done by replacing rules.

```
<Soccer>
    <Info>
        <Country>Austria</Country>
        <League>T-Mobile Bundesliga</League>
    </Info>
    <Result>
        <Date>12/7/07</Date>
        <TeamA>RB Salzburg</TeamA>
        <TeamB>SV Ried</TeamB>
        <FinalScore>2 - 0</FinalScore>
    </Result>
    <Result>
        <Date>12/8/07</Date>
        <TeamA>LASK Linz</TeamA>
        <TeamB>SK Austria</TeamB>
        <FinalScore>4 - 0</FinalScore>
    </Result>
    ...
    ...
</Soccer>
```

2.7 Crawling and Meta Tags

Sites can optimize the traceability of information by archiving the substantial information in meta-tags. These tags are located in the header region which are acquired and recalled by a web crawler. Meta-tags also contain useful briefings for web servers, web browsers and robots. They are exclusively designed to provide extra information which is not included in the site due to layout reasons. A header is invisible for web browsers. Therefore they are abused. For example it is possible to write thousands of keywords in meta tags and a crawler will list the site if someone looks for one of these keywords [Sch07].

Meta tags are titles, keywords, descriptions and robots. The most important one is the title. It is selected by nearly every crawler because search engines display them anyway. The title tag is built in the following way:
<title> title of a page </title>

The title of a page should be a significant description of the contents of a page. The title is shown in the title bar of a web browser. The title is used as the standard name for a bookmark. It is also displayed in the results list of a search engine [Sch07].

The second tag is keyword. Keywords fitting to a website and different spellings can be added. At the most it can contain 255 characters. This equals to 10 to 15 words. They are listed after the keyword content.
<meta name='keywords' content='keyword1, keyword2, etc.'>

Another important tag is description. Normally it contains a summary of the basic information of a site. It is listed in the results list of a search engine beneath the linked title. The tag is built in the following way:
<meta name='description' content='this is the description of a website'>

The last important tag is robots. It provides rules for the behavior of a crawler. But not all crawler support this tag. The tag looks like this:
<meta name='robots' content='all'>

The following table 2.2 shows the possibilities to control the behavior of robots. It displays the different variants of the entry tag. It allows controlling the indexing of a page and the following of links.

Entry	Index Page	Follows Links
all	YES	YES
index, follow	YES	YES
index	YES	YES
follow	YES	YES
nofollow	YES	NO
index, nofollow	YES	NO
noindex, follow	NO	YES
none	NO	NO
noindex	NO	NO
noindex, nofollow	NO	NO

Table 2.2: Control of bots with metatags

Furthermore it is possible to control a robot directly by the help of meta tags. However just googlebot (www.google.com), slurp (www.yahoo.com) and msnbot (www.msn.com) support the usage of their name in the name tag. The following code restricts it to disallow the access of a bot to a page:
<meta name='googlebot' content='noarchive'>

Another alternative to control a robot is the usage of robot exclusion standards which we will explain in the next subsection.

2.8 Robot Exclusion Standard

Sometimes it is not desired that a crawler can find a website, for example private sites with personal information which should not be listed by a search engine [Kos07].

Webmasters have different prospects to assure that crawlers stay away from their site. The first and most important method is to establish a robot.txt. This is called robot exclusion standard. The robot.txt is stored in the root directory of a site. Compliant robots first ask if there is a robot.txt. Unfortunately there are some wicked crawlers which do not ask *Am I allowed to enter?*. Such wicked ones can be banished. Banishing is also a feature of a crawler, which have not to be supported. This means there is again the problem that some spiders access the site anyway [Kos07].

Let us assume it is a well-behaving crawler, the crawler parses the robot.txt file line after line and follows the given rules. For example, it may be defined which parts of a site should be gathered. Each command is composed of a *User-agent* and a *Disallow* field [Kos07].

The *User-agent* field contains the following elements:

- The name of the robots which should follow the given rule.
- This could be a specific name of one or more robots or * for all robots.
- Each *User-agent* field has to be followed by a *Disallow* field.
- More than one *User-agent* can be stated.

The *Disallow* Field includes the elements given below:

- Link paths or patterns that a crawler should exclude during the gathering. This could be a full path, partial path, or an empty set. An empty set means that the crawler is allowed to gather the whole site.
- It is possible to use more than one *Disallow* field.

Here are some examples of robot.txt rules:

- Lock out all Robots from the entire server:
  *User-agent: **
  Disallow: /
- Allow all robots complete access:
  *User-agent: **
  Disallow:
- Exclude all robots from special parts of the server:
  *User-agent: **
  Disallow: /private/ Disallow: /temp/ Disallow: /abc/
- Exclude a single robot:
  User-agent: bot1
  Disallow: /
- Exclude more robots:
  User-agent: bot1
  User-agent: bot2
  User-agent: bot3
  Disallow: /
- Allow a single robot:
  User-agent: bot1
  Disallow:

A team of the University of Pennsylvania State worked on the robot exclusion standard. They implemented a tool called BotSeer to evaluate websites which support this standard. They reviewed 13.2 million websites and found 2.2 million robot.txt files which they analyzed. 30% of the websites use the robot exclusion protocol. Statistics can be found at http://botseer.ist.psu.edu [SCG08].

There are various other alternatives to block a crawler from a site. Some are blocking by the following concepts:

- Captcha: Captchas are used to distinguish between requests sent from human and computers. An image is generated with embedded letters and numbers and the user has to enter it to identify as a human user. The computer would not be able to read information out of an image [CS07].

- IP-address: Most web crawlers allow an arbitrary User-agent name. An exclusion by the User-agent name is not a sufficient method to deny bots the access to a website. A more efficient way is to exclude by IP-addresses.
- Excess traffic monitoring: A human reaches a maximum request rate per second much lower than a web crawler can. The website is monitored and requests with top level rates can be blocked [ZXL08].
  - Special tools
  - Java-Scripts
- Meta-tags (see previous chapter)

2.9 Web Crawler and Errors

There can be different errors during the crawling process. If an HTTP server cannot provide a page, it returns a header with an error code. This code is the key to identify the problem. For example 401 stands for file not found. The whole list can be found in the appendix (See chapter 7.5 HTTP Status Codes). This code is also used by the crawler [HC03].

For these special reasons some website administrators create special pages, but they are a problem for the crawler because these pages do not provide an error return code. These custom pages are called soft error pages. Unfortunately, these pages affect the crawler results. For example, if a page does not exist and the administrator established a soft error page instead of the original one, the header of this soft error page supplying 200. This means the page was successfully downloaded. The crawler now believes the required information is on the downloaded soft error page, but this is wrong [Sch07].

In the past, web crawlers caused overloads of networks and servers. Nowadays there is enough knowledge to prevent such errors. Crawlers have different ways to handle errors. Thus, we included error handling in the benchmark catalogue [Sch07].

3 Evaluation

3.1 Value Based Selection Process

The main goal of this work is to support a domain specific web crawler selection decision. To reach this, a value-based tool selection process is defined show in figure 3.1. The goal is to identify a web crawler that provides the highest value for future projects. This elaborated selection process can be used for other projects of the company. The proposed value-based tool selection process consists of the following steps:

1. **Identification of valuable web crawler benchmark features:**
   Due to the needs of the company the features are elaborated.

2. **Map value of benchmark catalogue with web crawler evaluation results:**
   After finding out the most important features the next task is to identify which crawler is applicable in which areas.

3. **Select the most valuable web crawler for a future project or develop one's own crawler:** After the successful evaluation it should be possible to decide if an existing crawler should be used. Another possible is to devleop a web crawler. The last alternative is to invest no more resources in this technology.

As described above, there is a need for some resources to support the value-based web crawler selection process:

- **Web crawler description:**
  In the beginning books, papers and existing crawler are studied. Based on the gathered knowledge a web crawler description is created.

- **Functional specification of a web crawler:**
  This contains the main features and functions which anappropriate crawler should contain.

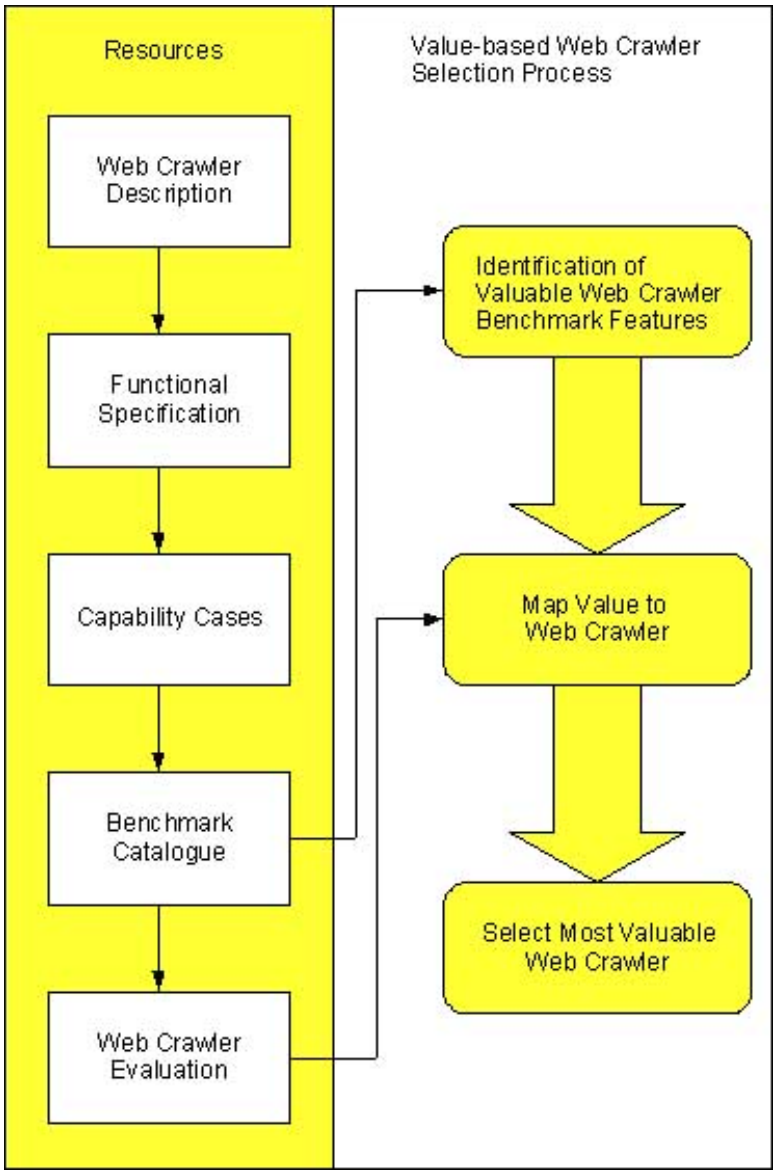


Figure 3.1: Value-based web crawler selection process

- **Capability Cases of a web crawler:**
  These cases are demonstration by typical use cases in the domain.

- **Web crawler benchmark catalogue:**
  On the basis of the functional specification and the capability cases the catalog is created. These catalogues are used as a checklist for the evaluations. If new features are discovered during the evaluation, this list is adjusted.

- **Web crawler evaluations:**
  The different web crawler are checked with the benchmark catalogue.

3.2 Benchmark Catalogue

This section contains desirable features and functionalities of web crawling and data extraction. A number of different criteria are required to point out the strengths, weaknesses and features in detail. The features were identified by performing a literature survey creating a functional specification and using capability cases. As there is a large amount of different features, we classified them in various groups. The classification based upon the functional specification. Additionally groups are named: usability, crawling standards and economic efficiency.

The features of the benchmark catalogue are classified as:

- Configuration
- Error Handling
- Crawling Standards
- Navigation and Crawling
- Selection
- Extraction
- File
- Usability
- Efficiency

The benchmark catalogue has the following advantages compared to other catalogues:

- The catalogue focuses on the requirements which are defined in the functional specification.
- Most other feature catalogues limit their lists on web crawling. This catalogue evaluates also the information extracting and delivering of fixtures or results in a proper standard.

3.2.1 Configuration

The following features describe the configuration of the web crawler (UC1: Configure crawler, UC1.1: Configure links and UC1.2: Configure rules). During the configuration it is defined which information should be gathered, where it can be found and where it should be stored. Although settings like the time zone of the crawled website needs to be defined. As the configuration is time and resource consuming it should be possible to create and use templates.

Feature 1 Creation/Modification of the data model
A data model is a specification to control the output format. A data model of a soccer fixture for example contains: Sport, Region, League, TeamA, TeamB, EventTime, EventDate. For the creation and modification there should be the following support:

- A Grahical User Interface (GUI) allows the user to define the data model of the extracting process.
- It is possible to create more than one data model (e.g.: different models for results and events).

Feature 2 Import/Export/Reuse of the data model
The import and export function allows to backup constructed data models. A data model for one league may fit to another league and should be reusable.

- The user can export a data model.
- The user can import a data model.
- The user can reuse a data model.

Feature 3 Creation/Modification of a scraping template
Scraping templates allow the user to specify the extracting process of information, e.g. to specify in which row and which column the required information is located.

- A GUI allows the user to define rules for the extracting process.
- It allows to store where required information is placed on a site.

Feature 4 Import/Export/Reuse of a scraping template
This allows the backup of constructed rules. A scraping template for one site may fit to another and should be reusable for another site by attaching it to the new site.

- The user can export scraping templates.
- The user can import scraping templates.
- The user can reuse the scraping templates.

Feature 5 Creation/Modification of URLs
The usage of URLs (Uniform Resource Locator) is essential for crawling because they are used to locate a page in the WWW.

- The user is able to define various hyperlink lists (e.g.: one for fixtures and one for results).
- The crawling tool allows to create, modify and administrate them.
- Hyperlinks should use parameters. The crawler should be able to combine parameters (e.g. the actual date).

Feature 6 Time zone
Some websites use the time from another time zone.

- A GUI allows the definition of the time zone of a site to ensure that the correct time is used.
- Some pages provide a function to switch between different time zones. The crawler should support this feature of pages.

Feature 7 Standard configurations
The system provides presettings which can be adapted easily.

Feature 8 Task administration
Recurring tasks should be managed in a task administration. Recurring tasks are used to gather fixtures automatically at fixed or specific time intervals.

3.2.2 Error Handling

During the crawling or extracting process there can be various errors. These features are checked in this chapter (UC2: Manage notifications). The crawler should run stable. Errors that occur should be handled by the tool. For example,

this could be done with notifications or protocol functions. A visual debugger can help to find the reasons for errors.

Feature 9 Intelligent adjustment
If the structure of a page changes, the crawler should still select and extract the information correctly. To evaluate this feature, a sport result page will be crawled before and after some changes. Then the resulting fixtures are compared.

Feature 10 Protocol function
The tool provides the capability to log all activities of the crawler.

- It should provide an overview of all crawled sites and information extraction.
- It should be possible to export these items.
- If results or fixtures were added or errors had occurred, the log file should note the time, name of the crawler and other additional information like type of error.

Feature 11 Notifications
During the crawling process different messages can occur.

- For each error a notification has to be created which should contain: error message number, error code, date, time, crawler, page.
- The notification should be clear and easy to understand.
- Different error (e.g.: error or advises) messages should be displayed different (e.g.: yellow for advises and red for errors).

Feature 12 Notification administration
Produced notifications need to be administrated and prepared for a user.

- They should be provided in a graphical user interface.
- It should be possible to export and archive old messages.

Feature 13 Visual debugging
There should be visual debugging facilities.

- It should be possible to set points where the crawler stops the process and waits for a user input to proceed.
- It should be possible to go step by step from this point.
- It should be possible to save a screenshot from a particular sequence.

3.2.3 Crawling Standards

This section describes the prevention of web crawlers from accessing parts of a website or the complete site. The owner of a site can add such restrictions to assure the operation of his site. Especially it will be assured how far the robot exclusion standard and its add-ons are restrained. Additionally the abidance of addressed meta tags will be checked. The evaluation should check if these features could be switched on or off and how the default is defined.

Feature 14 User-Agent
Each crawler can be identified by the user-agent field. The user-agent field provides the name of a crawler. This feature benchmarks the identification and possible settings of the user-agent field.

- It should be possible to choose the name without any restrictions for example to mask as a browser (e.g.: Mozilla/5.0).
- The user-agent field is tested with the site http://www.useragent.org/.

Feature 15 Robot exclusion standard
The crawler should support the robot exclusion standard which can be added by the owner of a page to ensure that crawlers do not disturb the regular traffic.

- The user-agent field defines if crawlers are allowed to grab the page. The crawler should be able to understand this information and should react.
- The disallow field defines if a crawler has the permission to crawl the whole page, special parts or even nothing. The crawler should react as defined.

Feature 16 Robot exclusion non standard extensions
Presently there are various add-ons for the robot exclusion standard. These non standard extensions are checked in this feature.

- A sitemap is a list with the URLs for all components of a whole website. The crawler should be able to read the auto-discovery field which describe where the sitemaps are located. This information should be integrated into the crawling process (http://www.sitemaps.org/).
- The crawl-delay field restrains a waiting period after a successful request to a server. The crawler should keep this condition.

- The robot exclusion standard does not support allow fields. The nonstandard extensions accept the mixture of allowed and disallowed fields (e.g.: disallow access to all pages but allow access to index.htm). The crawler should react as defined.

Feature 17 Robot exclusion extended standard
The robot exclusion standard does not support mechanisms to find out good and bad times and optimal intervals for the crawling requests. This does not help to overload a web server. For this purpose an extended standard was created. These extensions are checked in this feature.

- The request-rate defines the maximum accesses to pages on a domain in a defined time period. The crawler should respect this limitation.
- The visit time field defines a period when the site should be visited by a crawler (e.g.: accesses are just allowed between 3 and 5 o'clock in the morning). The crawler should react as defined.

Feature 18 Excluding by meta tags
With an HTML <META> tag it is possible to explain the crawler that it is not allowed to index the page or search for links.

- Follow Links: The crawler should know if it is allowed to follow a link.

3.2.4 Navigation and Crawling

This part describes how a crawler finds the right pages and navigates to the required information (UC3: Open website). This evaluation tries to find out if navigation is recordable and reusable. Furthermore, the crawler should be able to navigate to the next page. For example, switch to the next match day by use the link to the next page. Another important research field is the parameterizations of links. For example: www.sport.com/day/. The variable day is filled with the actual date. Some websites use cookies or pop up windows. The crawler should handle them smoothly. Furthermore it is assured that a crawler can handle log ins and other navigation elements like buttons, drop down menus, checkboxes and others.

Feature 19 Recordable crawler navigation
The navigation should be recordable that the crawler learns to navigate on different sites.

- It should be possible to record a sequence of web interactions through one or more web pages that the crawler can adapt and "learn" how to navigate through similar pages.
- The crawler is able to handle JavaScript.
- The crawler can comprehend mouse clicks.
- The crawler can fill in forms with parameters.

Feature 20 Log in
Some pages demand an authentication via access keys. The crawler should be able to log in or log out in an automated way.

- The crawler should be able to do a form log in. This is the most common type of a log in where the username and password are entered into a form at a log in page.
- It should also be possible to do a HTTP log in. This is needed if the username and the password are entered into a special prompt window opened by the browser.

Feature 21 Captcha
Captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) is a type of challenge-response test to distinguish between human and computer users. The evaluation uses this Captcha: http://recaptcha.net/plugins/php/

Feature 22 Parameterized crawling
The crawler generates links from stored parameters. This allows a direct access to required sources. E.g. first league, Austria, 6th June 2008 and the crawler generates
www.soccer.com/results.asp?country=austria_league=first_day=060608

Feature 23 Waiting periods
It should be possible to define waiting periods when opening the pages of a website. This grants the stability of the scraped website. In some situations it is desired that the extraction look more like human created, in this case random time delays in the action sequence can be added.

Feature 24 Multitasking
The crawler should be able to crawl more than one site at the same time.

Feature 25 HTTP standard codes
The crawler should be able to understand HTTP status codes and react in an appropriate configurable way. For example if a page no longer exists the web server sends the status code 404 Not Found. The crawler should generate a notification and proceed to the next page. The whole list of HTTP standard codes can be found in the appendix 7.5.

Feature 26 Filling forms
The crawler should be able to fill in forms like humans do.

- It should be able to fill in multiline text fields.
- Additionally the crawler should be able to select an entry of a drop down list.
- The crawler should be able to select radio buttons.
- It should submit the complete filled forms.

Feature 27 Pop up windows
At the moment there are some pages which open pop up windows that could disturb the crawling process.

- It should be possible to allow or block all pop up windows.
- It should be possible to allow or block pop up windows by matching the name with defined patterns.
- It should be possible to interact with pop up windows.

Feature 28 Cookies
The Crawler should handle Cookies like a browser.

- It should be possible to save cookies.
- It should be possible to administrate cookies.
- It should be possible to delete cookies.

Feature 29 Secure connections
The crawler should be able to handle with the HTTPS-protocol additionally to the HTTP-protocol.

Feature 30 Next page
It is possible to define the position of a link to the next page. This could be used to iterate each round of a league.

3.2.5 Selection

This chapter describes features which are used during the information selecting and finding. This can be done based on surrounding landmarks, on HTML attributes, on the contents itself, on order of appearance, or on semantic/syntactic concepts on websites (UC4: Use Rules). Furthermore different web technologies like HTML, XHTML, JavaScript or Flash should be supported. Often information about sport events are embedded in tables. The crawler should iterate a table row by row. The header of a website contains important information. It should be possible to gather this information.

Feature 31 Selection methods
There are different methods to extract information.

- XPATH (XML Path Language) is a language to find information in XML-documents. This language uses nodes like namespaces, text, comments and others to select them. A crawler should be able to understand and use this technology to gather specified information from HTML documents.
- Regular expressions are used to find information by syntactic rules. The crawler should support this feature.
- Node attributes (e.g.: soccerway.com provides for team A the following node attribute: <td class='dynresults2_teama2'>Girondins Bordeaux</td>) can be used to figure out a particular part of the web page which contains the needed information.
- The user can add his/her selection scripts.

Feature 32 Tables
It should be possible to extract information form a table of an HTML site. It should be possible to define rules to exclude rows for example if they are the caption of a column.

Feature 33 Technologies
The crawler should be able to understand various technologies used in the WWW:

- HTML
- XHTML
- Flash
- JavaScript

Feature 34 Capturing meta data
The crawler should be able to interpret meta information like information about the document. It is necessary to capture and manage this additional information of websites.
For example, the crawler provides the following means to store this additional information:

- Date of the last change of the site
- Keywords of the site
- Description of the site

Feature 35 Header Information
There should be a possibility to access to the header information of HTML documents. This is especially important to reconstruct where the collected information comes from.

3.2.6 Extraction

This chapter describes the features needed for the information extracting especially which technologies and methods are supported (UC5: Extract data). It is evaluated if a crawler can distinguish between fixtures and results assumed that they are outlined different at the website. Furthermore it is important to gather substrings because sometimes information is embedded in large strings. It should be possible to validate the scraped information. Proper standards are needed to enable further data processing, e.g. a unique time format.

Feature 36 Extraction condition
An extraction filter is needed to verify the founded information and to extract it just when a special condition is fulfilled. A possible extraction filter could be a verification string with the end time of a game, so the crawler just extracts the final score. This is necessary to distinguish between final results, intermediary results and fixtures.

Feature 37 Efficient extraction
Extract just required information defined by the data model and ignore unimportant ones.

- The user can extract only the desired pattern from a page.
- The crawler gathers only complete data sets of a sport event.

Feature 38 Transformation of formatting differences
Elements like time and date are displayed in different standards on different websites.

- There should be a possibility to normalize the data format to one defined standard (e.g.: Apr 13, 2008 should be transformed to 13/04/2008).
- It is possible to convert the extracted time to the UTC.

Feature 39 Validation
A validation should assured if the scraped content fits to the expectations. For example, this allows matchings if the founded results are identical to the regular expressions (e.g.: 2-2 or a 4:0 will match with this regular expression [0-9][-:][0-9]).

- Comparing text patterns with regular expressions.
- Producing notifications when they do not fit together.

3.2.7 File

This part describes how the created files with the extracted information are assembled (UC6: Convert data to a proper standard and UC7: Save File). It is necessary to have a unique file name. Otherwise it can happen that files are overwritten and data gets lost. There should be a broad support for different file formats. The structure of the files has to be fixed so that a further automated data processing is easy. As an alternative to write the information into files there should be a broad support for the storage in different databases.

Feature 40 Unique name
Each created file should have a unique name to assure that files are not overwritten. A user defined prefix should be added for example to distinguish different crawler which gather the same site at the same time. The name of the file contains:

- Sport
- Region
- League
- Date
- Time
- Site name

Feature 41 Standard file formats
It should be possible to store the scraped data in various standard file formats. This is needed to provide a smoothly integration into an existing data processing system.

- XML
- CVS
- TXT
- ASCII

Feature 42 Customize standard formats
It should be possible to change the structure of the standard formats. The architecture should be as simple as possible to ease the further processing. For

example:
<Result>
<Sport>Soccer</Sport>
<Region>Österreich</Region>
<League>T-Mobile Bundesliga</League>
<EventDate>11.07.07 19:30</EventDate>
<TeamA>Rapid Wien</TeamA>
<TeamB>Wacker Innsbruck</TeamB>
<Outcome>3:1 (1:1)</Outcome>
</Result>

Feature 43 Database connectivity
It should be possible to store located information in a databese. This can be used like an XML file for further processing of data.

- MySQL
- Oracle
- Access
- PostgreSQL
- MS-SQL-Server

3.2.8 Usability

This section describes the elegance and clarity of the interaction with the crawler. The user should be able to use the crawler intuitive after a short introduction. The tool has to be expendable and adaptable to the users' need. Good documentations and help functions are desirable. A good usability ensures a time and resource efficient extraction process.

Feature 44 Usability
The user should be able to use the crawler in an accurate way.

- The user can associate icons with related function.
- The user can handle the software without any help functions.
- The user can use the software without any introduction.
- The user interface is structured clearly and consistently.

- The user has the possibility to use an undo function.
- Short-cuts allow a fast and efficient interaction.
- The crawler offers good default settings. The program is ready to start from the beginning.
- The user can copy settings.
- When the program is started the last project is opened automatically.
- Tutorials are available.

Feature 45 Adaptability and expendability
The user should be able to extend or adapt the functions of the crawler.

- The user can adapt the menu entries to his needs.
- Plug-ins can be added.
- Macros and scripts can be added.
- Other features can be added to extend the crawler.

Feature 46 Help functions and documentation
The user should have the possibility to use help functions. The help functions may cover the following points.

- Online help is available.
- Offline help is available.
- Online documentation is available.
- Offline documentation is available.
- The user can search through the online help.
- The user can search through the offline help.
- Examples of the usage of the crawler are provided after installation.

Feature 47 Error handling
The user should be notified if an error occurs by the following items.

- In case of an error an error report is created.
- The error message is clear and comprehensible for the user.

- The program tolerates small errors (like an other date format).

Feature 48 Installation
The installing process should be as easy as possible.

- Automated installation routine with few user interactions.
- The user is able to install the crawler in 10 minutes.
- Provides a guidance to protect users from errors.
- The user is able to install the crawler without the usage of the manual.
- During the installation process no error messages occur.

3.2.9 Efficiency

These features evaluate the relation between costs and earnings of a tool. First the different system assumptions are evaluated. Another aim is to have a platform independent tool. Furthermore the usage of a proxy server should be supported. At least it will be tested if the crawler pads into prepared traps.

Feature 49 System assumption
The user should know what system is needed to run the crawler smoothly. The crawler should not jeopardize the overall computer performance. This feature evaluated what resources are used.

- Memory
- CPU
- Disk space
- Additional required Tools like DB

Feature 50 Platform independence
The crawler should run on various platforms. The crawler will be installed on one platform and it will be investigated on which other operating systems it will run.

- Windows
- Linux
- Solaris

- Mac OS

Feature 51 Support
This feature should evaluate the support possibilities for a crawler.

- Support telephone number
- Online support

Feature 52 Effectiveness
It should be possible to configure and use the crawler in a time and ressource efficient way.

- The resulting performance should be compared in relation to the effort of configuration.

Feature 53 Proxy server
The crawler tool should allow the usage of a proxy server. This allows anonymity and avoid IP restrictions of pages.

Feature 54 Crawler traps
Crawler may get into traps like infinite loops. This costs a lot of resources, lowers the productivity, and can crash the crawler. This feature checks if a crawler trap prevention algorithm is implemented.
The following script is used http://www.spider-trap.de/en_download.html for the evaluation.

3.2.10 Overview of Benchmark Features

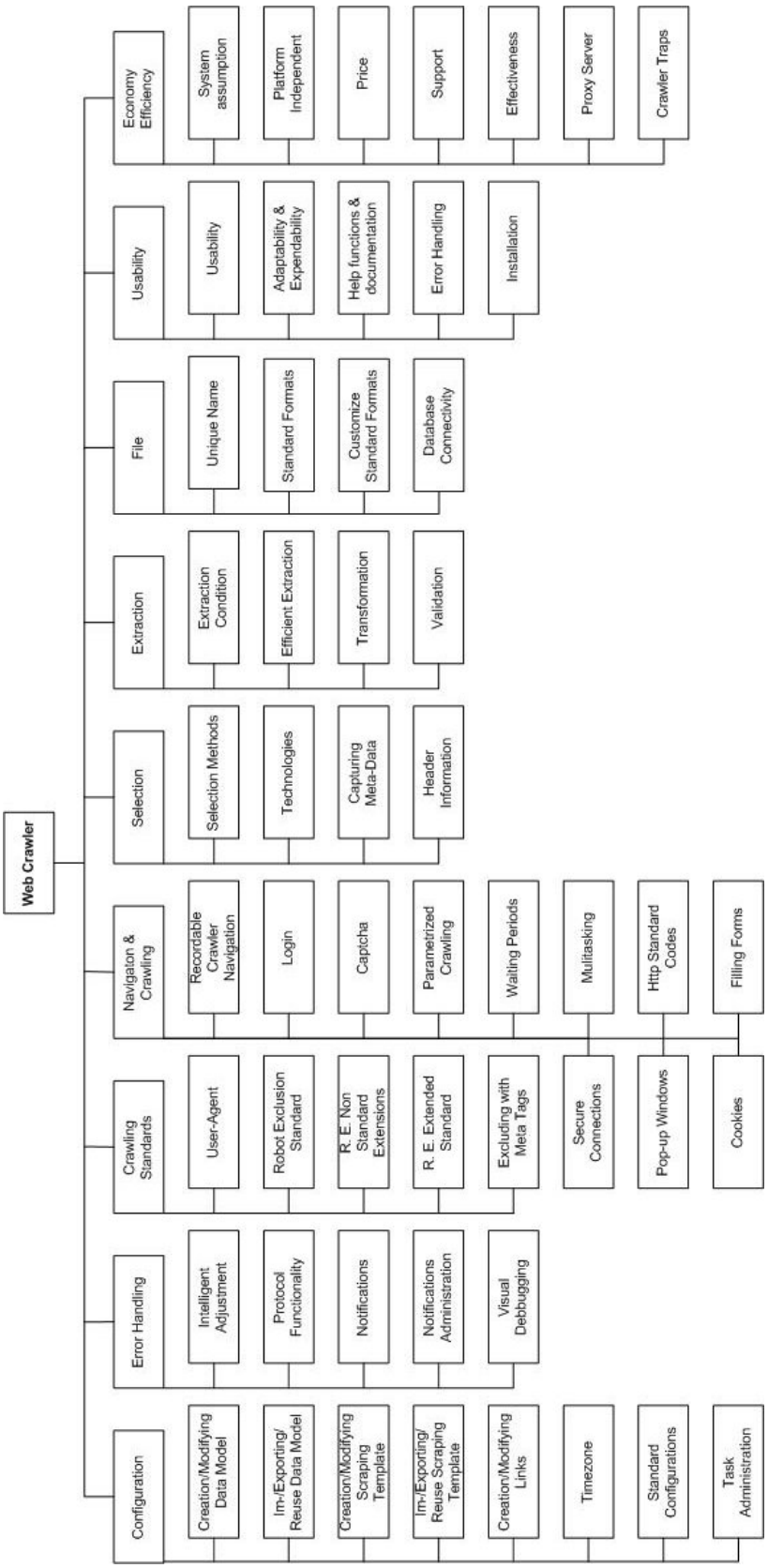


Figure 3.2: Overview benchmark features

3.3 Research Questions

For the approach described above, the following research questions should be answered by this master thesis:

RQ1: Which web crawler exist at the market?
I used search engines and the literature in the appendix 7.5 to get an overview and answer this question.

RQ2: Which web crawler fit the needs of the domain?
I defined a benchmark catalogue to answer this question. The benchmark catalogue is the basis for the web crawler evaluation.

RQ3: What are the strengths and weaknesses of the selected web crawler?
I selected eight web crawlers to answer this question. They are evaluated and benchmarked based on a benchmark catalogue which was created according to the functional specification. The strengths and weaknesses of each web crawler should be highlighted by this evaluation.

3.4 Strategy of the Evaluation

This chapter describes the strategy of the evaluation. Due to the reason that an evaluation of all existing tools is too time-consuming, eight crawler were selected. The crawlers are selected by a market overview. Then a preliminary selection based on the benchmark catalogue was created. The most important criteria were the support of the required features and the status of the development of the crawler.

The following crawlers were selected.

- Lixto Visual Developer 4.2.4
- RoboSuite 6.2
- Visual Web Task 5.0
- Web Content Extractor 3.1
- WebSundew Pro 3.0
- Web-Harvest 1.0
- Web Scraper Plus+ 5.5.1
- WebQL Studio 4.1

The next step was demanding the required tools. The company provided the ressources for the evaluation. The crawler were all installed on a computer with the following configuration:

- Intel Core2 CPU 1,86GHz
- 2GB Ram
- WinXP SP2

The benchmark catalogue was used as a checklist to evaluate the tools. A detailed description of all criteria can be found in the benchmark catalogue 3.2. During the process some new features were identified and added to the catalogue. Due to these changes the already evaluated tools had to be reevaluated.

A delivery document was created for each crawler. This document was reviewed by experts of bwin. Each document contains a short description of the tool, the advantages and disadvantages. At the end of each document detailed information and ratings on all features are listed.

4 Results

These sections contain the results of the evaluation, structured in the same way as the categories in the benchmark catalogue: Configuration, Error Handling, Crawling Standards, Navigation and Crawling, Selection, Extraction, File, Usability and Economic Efficiency.

4.1 Lixto Visual Developer 4.2.4

Producer	Lixto Software GmbH
Test version	Lixto Visual Developer 4.2.4
Acquisition costs	Inividual price
Platforms	Windows XP, Windows 2003 Server, Suse Linux Enterprise 9, Suse Linux Enterprise 10
Homepage	www.lixto.com

Table 4.1: Overview of Lixto

4.1.1 Introduction

The Lixto Suite is a tool for web information extraction. It consists of the following two tools:

- **Visual Developer:** The Lixto Visual Developer is a software tool that is responsible for navigating through the WWW. It allows the user to gather data in a structured way. It uses the Eclipse IDE as a framework and the Mozilla browser engine.
- **Visual Developer Runtime Sever:** This is a server environment for Visual Developer.

The figure 4.1 on the next page illustrates the interaction between the described tools.

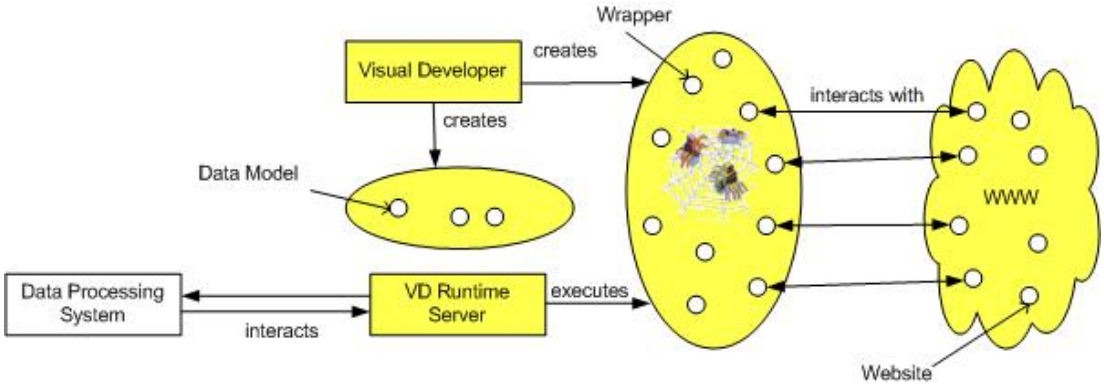


Figure 4.1: Architecture of LixtoSuite

4.1.2 Evaluation Results

Configuration

Data Models can be easily created and are displayed in a tree view. This structure defined by a data model, has different depth levels and attributes. The levels depend on the defined structure. It is possible to define the type of each element e.g.: date, integer, string and others. Additionally, elements can contain attributes which provide more detailed information. This is shown in figure 4.2 with the attribute *Type*. These data models can be exported and imported as XML file.

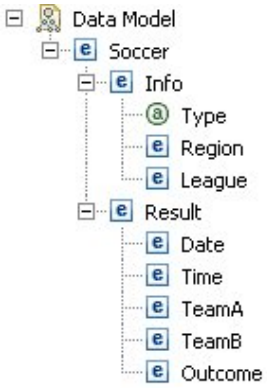


Figure 4.2: Lixto Data Model

A scraping template can be easily created by adding actions. For example an action could be: load an URL or follow a link. The complete crawler settings are saved in a *.lixvw file. Parts like a scraping template can not be exported. Negative aspects of the configuration are that there is no automatic time zone conversion and that it is not possible to configure standard templates to save the

configuration time.
The configuration of the crawler is shown in figure 4.3. In the main frame the website is displayed. Under the main frame there is a window with the HTML code. At the bottom of the screen, actions can be added. On the left side, there is an action sequence which defines how to go to the Norway soccer league and how the data model is connected with the website.

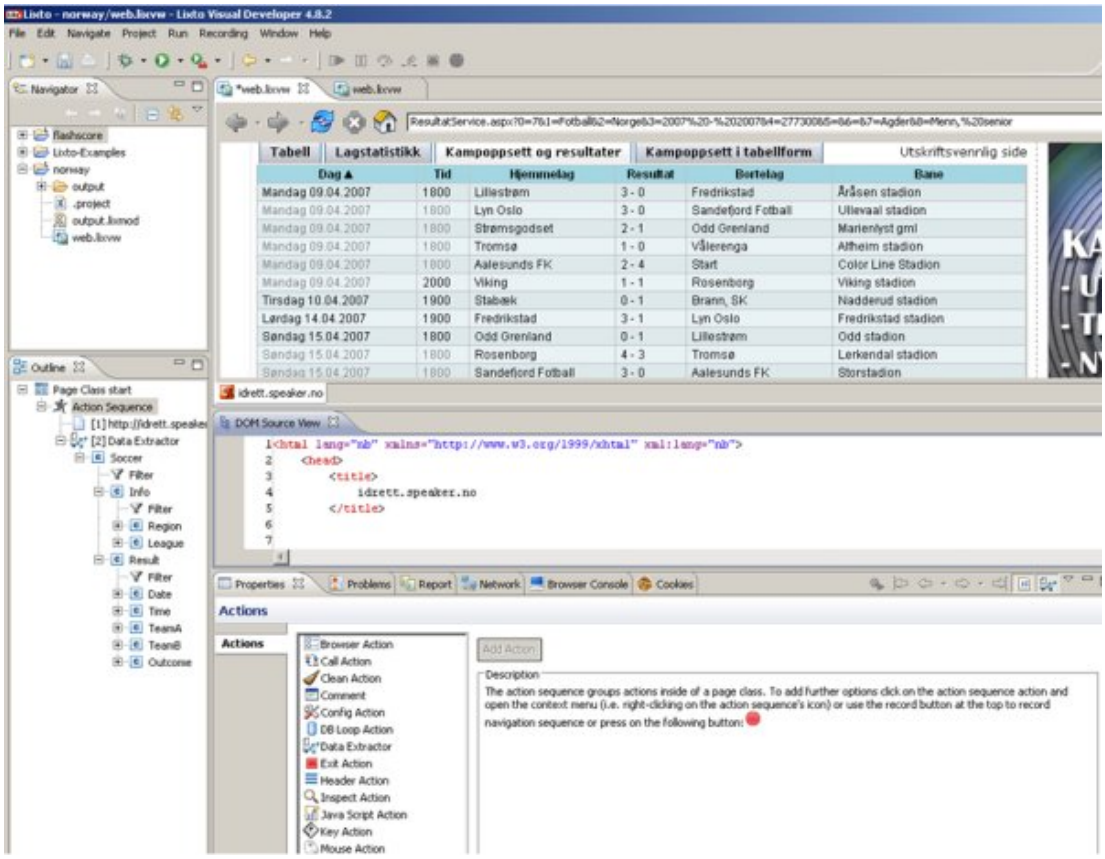


Figure 4.3: Creating scraping template

Error Handling

It is possible to debug the configured crawler. Breakpoints can be set at any time. It is possible to start and stop at a defined sequence of the crawling and extracting process.
The Lixto Visual Developer automatically logs every action. It is not possible to choose the log depth. Each message is logged in a report. Each defined action results in a notification in a report line. For example open the website www.soccer.com generates: "Executing URL: www.soccer.com". Notifications are administrated in a report view which are integrated in the main window.

Crawling Standards

The name of each crawler configuration can be chosen arbitrarily. It can be defined by a browser action. Sitemaps can be found due to the tags (head, title) in the HTML code. The robot exclusion standard, the extended robot exclusion standard and the excluding by meta tags are not supported.

Navigation and Crawling

Navigation can be recorded simply by clicking on a record button. The navigation to a desired page is recorded automatically. The URLs which should be crawled can be parameterized with an URL action (e.g.: 'www.soccer.com/' + region + '/' + league). Waiting periods can be defined with a waiting action. This allows a human like behavior. Forms can be filled, options, checkboxes and radio buttons can be selected. The filled form can be submitted. This is easily done by simply recording the actions. The crawler is able to handle pop up windows. With the switch action it is possible to switch between the different pop up windows. The crawler handles secure connections smoothly. Cookies are automatically added. They can be removed by a clean action. A next page pattern can be defined and is iterated without any problems.

Selection

Information on web pages can be selected by filters illustrated in figure 4.4. These filters identify elements on the web page. The selected information is displayed in a green box. The creation of a filter can be easily done by adding a filter to an attribute of the data model and by clicking on the corresponding content on the web page. It is possible to choose different methods to select the information: XPATH, regular expressions or scripts. Furthermore it is possible to restrict the selection by node attributes and node properties.
Information in tables can be selected without any problems like shown in figure 4.4. Rows and columns can be excluded. It is possible to access the header information. This is done by the header action. Meta information can be extracted from websites without problems. The selection of information on HTML and on XHTML websites works smoothly. A negative aspect is that information on flash sites can not be selected.

Extraction

Information can also be extracted due to a specified condition on the web pages. For example: Extract fixtures of finished games, not of running ones. Four dif-

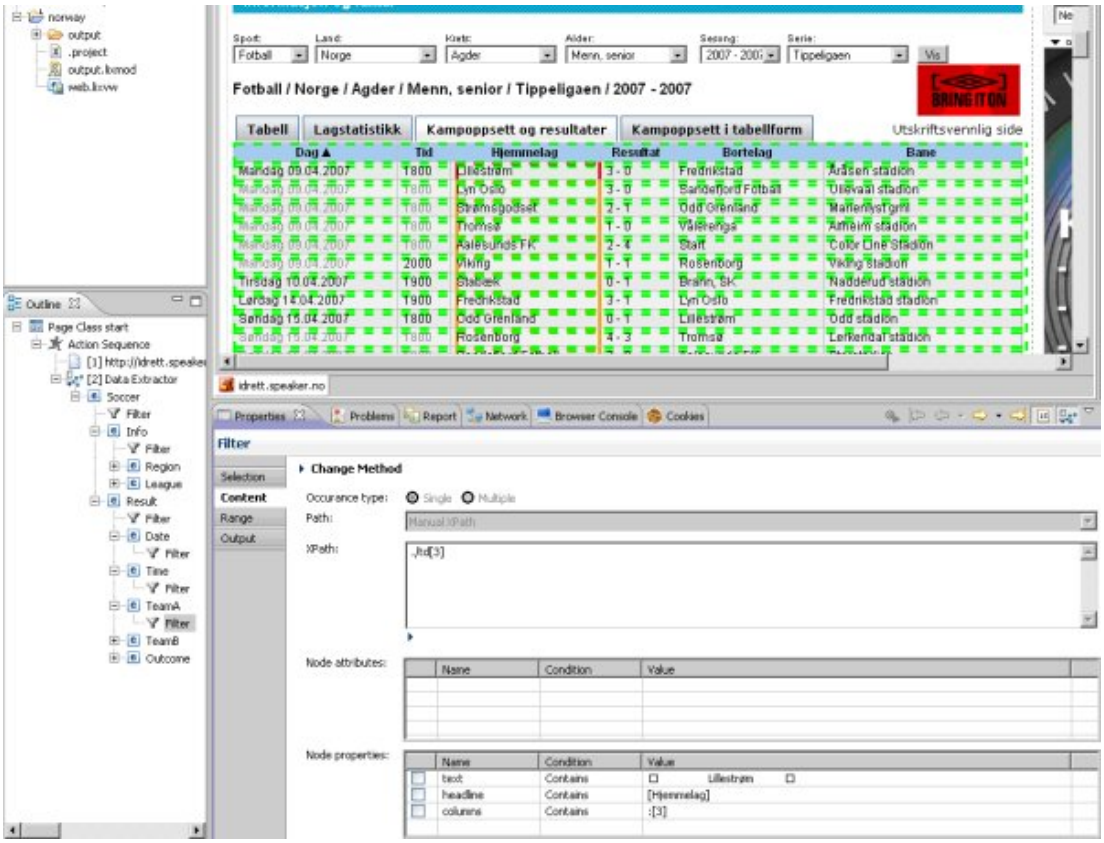


Figure 4.4: Selection of information

ferent conditions can be defined to facilitate more restrictions:

- Attributes (contains, exact or regular expression)
- Contextual (before, after, notbefore, etc.)
- Internal (contains, notcontains)
- Range (e.g. fourth up to the second-last)

Different formats can be unified to a custom standard. For example, 1:1 and 1-1 as results for a game. Therefore a string or a regular expression can be defined which should be replaced by another. For example, change the words "Juventus Turin" into the defined substitution "Juventus". Such a filter can be used for the whole page. A negative aspect is that information could not be validated with a target pattern.

File

The extracted information can be saved in a file. For this file any name can be chosen. The output format is XML. The structure of the file is defined in the data model. An XML file can be structured like the following example.

```
<Info>
    <Sport>Soccer</Sport>
    <Region>Norway</Region>
    <League>Tippeligaen</League>
</Info>
<Result>
    <Date>09.04.2007</Date>
    <Time>18:00</Time>
    <TeamA>Lillestrøm</TeamA>
    <TeamB>Fredrikstad</TeamB>
    <Outcome>3 - 0</Outcome>
</Result>
```

Furthermore, the tool supports the following databases: Oracle, MySQL, PostgreSql and other ones which work with JDBC.

Usability

The interface of the Lixto Visual Developer is outlined well. The icons are easy to understand. The appearance of the tool can be adapted and changed to the user's wishes. The tool itself could not be extended by the user. Not each function is explained in the online help. The documentation is well structured and very clearly arranged.Once an error occurs, an error message is created which is quite clear. The installation is done by a setup wizard.

Efficiency

The application runs smoothly on computers with standard configuration. A version for Windows and Linux is available. Other platforms are not supported. It is easy to learn the handling and the usage of the tool. There are various tutorials available. Some of these tutorials do not work due to structural changes of the associated website. A negative aspect is that the crawler can trap into a crawler trap. A crawler trap is a structure which causes the crawler to make an infinite number of requests.

4.2 RoboSuite 6.2

Producer	Kapow Technologies
Test version	RoboSuite 6.2
Acquisition costs	individual price
Platforms	Windows, Linux
Homepage	www.kapowtech.com

Table 4.2: Overview of RoboSuite

4.2.1 Introduction

The RoboSuite consists of several individual tools. It supplies tools to create a crawling task, administrate and monitor a crawler.

- **ModelMaker**: It is used to define a data model. For example a data model for a soccer result contains the name of the region, the name of the league, the name of the two playing teams, the date of the event and the outcome. This tool defines which information the created crawler shall extract.
- **RoboMaker**: This is the most important tool in the RoboSuite. A graphical surface allows the user to configure a robot for certain tasks and if necessary to debug them.
- **RoboRunner**: This is acommand line tool. With the aid of this tool the crawler can be executed. The majority functions of RoboServers are supported by the RoboRunner.
- **RoboServer**: RoboServer is an application for executing robots as a service to clients. It accepts requests and sends back replies to client applications.
- **RoboSuite Control Center**: RoboSuite Control Center allows the remote monitoring RoboServers and their robots.
- **RoboClient**: RoboClient is a simple client to RoboServer that allows testing various requests and to get a feel for RoboServer.
- **RoboManager**: It is tool to organize and maintain a large number of robots. RobotManager monitores the performances of the crawlers and detect failures and problems.

Figure 4.5 illustrates the interaction between the described tools.

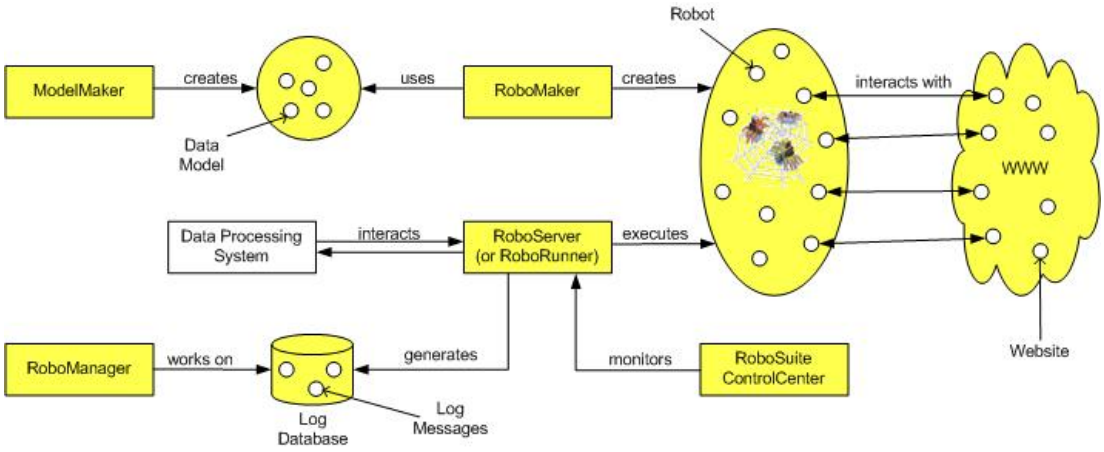


Figure 4.5: Architecture of RoboSuite

4.2.2 Evaluation Results

These sections contain the results of the evaluation, structured in the same way as the categories in the benchmark catalogue: Configuration, Error Handling, Crawling Standards, Navigation and Crawling, Selection, Extraction, File, Usability and Economic Efficiency.

Configuration

Data models can be created with the tool ModelMaker. This tool calls them objects. These objects can have various attributes with the following types: Short Text, Long Text, Character, Integer, Number, Date, Boolean, and Binary. The value of an attribute contains the required information. It is possible to configure that defined attributes have to be filled with data. Another possibility is to set default values. The saved data models can be imported into the RoboMaker. This allows an efficient reuse of the created data models. Figure 4.6 shows the ModelMaker and the attribute configuration window.
A scraping template can be created by simply adding an action like a mouse click. It is not possible to export scraping templates. Nevertheless there is a detour to export a configuration by creating a crawler task and save it. This can be used as a template.

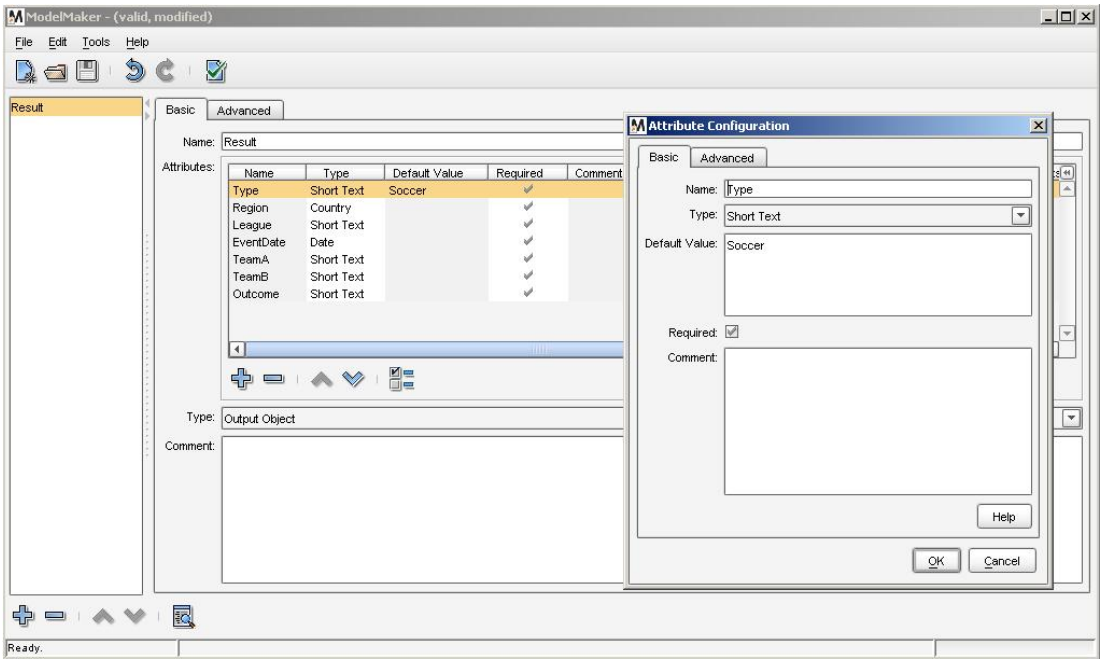


Figure 4.6: Creating data models with ModelMaker

A default time zone and a result time zone can be configured. The time is converted automatically in the desired format. A data model can be exported to a robot library file to automate tasks. The RoboServer can execute these files in certain time periods or at fixed times and crawl the required information automatically from the Web.

Multiple instances of the crawler can be grouped to ease the administration. For example the configured crawler for the web pages of the English Premier League can be combined to a group, as shown in figure 4.7. These crawlers gather the required information.

Figure 4.7: Group building and task administration

Error Handling

RoboDebugger can be used to debug a configured crawler. Breakpoints can be insert to detect errors easier. This allows running from breakpoint to breakpoint and finding errors like wrong configuration or website has been changed. A notification via e-mail can be added as action to each step of the crawler. Status and error messages generated by robots are presented in the RoboManager. The identification of crawlers that are broken because of significant web interface changes is quite easy. The crawler can handle small changes like changed formatting tags. As soon as the structure is changed this leads to an automated error.

For example, the configured rules have to be changed if the amount of colums has been changed.
The tool also provides a protocol function. This is realized by a log file which is created to note information like the crawled URL, time, error, etc. In the following example 21 results were successfully generated by the *WorldCupFifaCom.robot* and no errors occurred.
Message Number: 22, Message Type: $Object_Handling_Info$*, Message Created: 2008-04-14 16:12:02.444*
Robot URL: file:/H:/Kapow RoboSuite 6.2/bin/WorldCupFifaCom.robot, Robot Id: (none), Robot Run Id: 0
Message Text: Result summary: Total: 21, ignored: 0.

Crawling Standards

The name of each crawler configuration can be chosen arbitrarily. Sitemaps can be found due to the tags (head, title) in the HTML code. The robot exclusion standard, the extended robot exclusion standard and the excluding by meta tags are not supported.

Navigation and Crawling

Waiting periods can be defined to make sure that the crawler behaves server-friendly. This is called wait action and can be added at every point of the crawling process. There are two ways to define waiting periods. The first method is to define a time period which have to be awaited. The second one is to define a "wait for time event" which means the crawler has to wait a certain time period after a specific clock time. The crawler can handle with pop up windows.
The RoboMaker can fill out forms, choose options, checkboxes and radio buttons. The filled forms can be submitted. The crawler is able to log in via forms and HTTP log in.

Furthermore cookies are administrated in a cookie view. Cookies are added automatically if a page has been loaded. RoboMaker provides three actions for cookies: it is possible to extract cookie values, to delete and to create own cookies.
Some disadvantages are that the crawler can not handle Captchas and the program can not react on HTTP standard codes.

Selection

By the aid of regular expressions just parts of strings can be selected. Regular expressions specify words, particular characters or patterns of characters. For example if you have the string *Albanien - Kampionati shqiptar - Spieltag/Tabelle* RoboMaker can divide it into the region and the league. Figure 4.8 illustrates the RoboMaker tool. In the top of the screen the steps of the configured crawler are shown. In the center of the screen the crawled page and below the HTML code of this page is displayed. The blue selection line shows the selected row and the red selection line shows the TeamB of a soccer event. The order in which the actions were configured is shown in the upper part of the screen. The order is very important, e.g. it is necessary to log in before a menu entry can be opened.

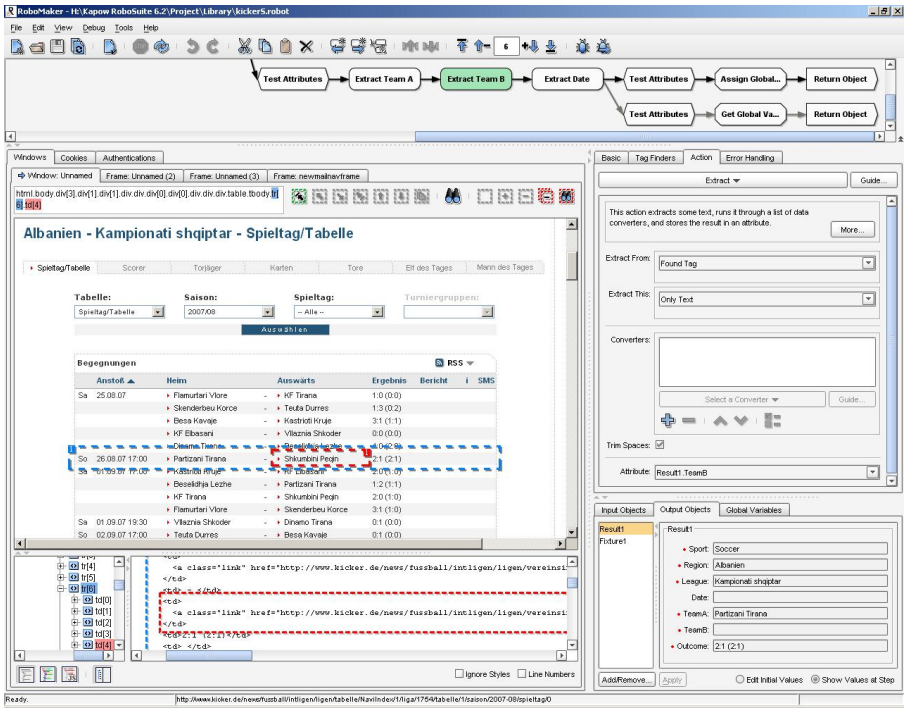


Figure 4.8: Selection of information with RoboMaker

RoboMaker uses similarities like same tags in the HTML code to recognize which part of the information should be extracted. The selection and extraction process is linked with the data model. The selection of information structured in tables operates smoothly. Therefore a loop is created and each row passed through. For example the crawler can learn that in the first column $td[1]$ the date of the event, in the second column $td[2]$ the league of the sport event is displayed and so on. Criteria can be defined to skip rows and columns. Meta and header information like the title of a page can be selected. A negative aspect is that the crawler can not handle flash content.

Extraction

The RoboMaker provides a wide range of possibilities like patterns to extract data, to validate them and to convert them into a defined format. The RoboMaker supplies a data conversion to combine various formats on the different pages. For example a date on a website has the format *Wed, July 10, '07*. This is equal to the following pattern: *EEE, MMM d, 'yy*. Now the crawler can transform it to the selected output pattern for example German: *10 07 2007*.
The tool provides a function for matching names. For example names of teams can be matched to the standard team names. The advantage would be that a further data processing program of the company does not need to match teams, leagues or other names. This is realized by creating a list with names on a website and the associated names in the target system. This is shown in the figure 4.9.

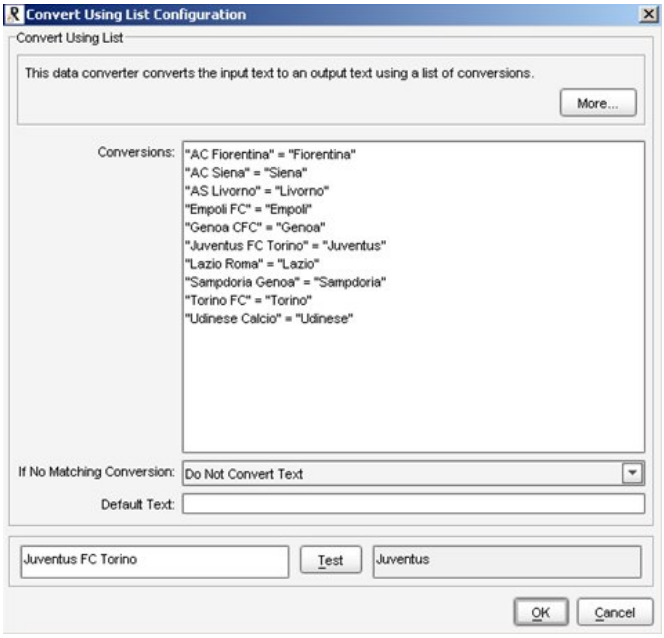


Figure 4.9: Converting names

Furthermore the crawler knows if a sport event has already ended if it is prepared properly. Test tags are used to realize this. For example it is checked if the word "FINISHED" is contained in a sport results.
Patterns are used to validate the information to ensure that the crawler does not gather "data garbage" from a page. For example the outcome has to match the following pattern: digit followed by a double dot followed by a digit. If the pattern does not match the information will not be gathered and a notification about the failed extraction is created.

File

The extracted information can be saved in files. Any name can be chosen. As the standard output format XML is configured and the structure can be changed by the data model. The RoboSuite supports the following databases: Oracle, IBM DB2, Microsoft SQL Server, Sybase PointBase Server, Solid, MySQL. For the evaluation a MySQL database was used. The data model is connected with the entry in the database.
An XML file can be structured like the following example. The structure of the file is defined in the data model by the tool administrator. Results can be stored behind one another.

```
<Result>
    <Sport>Soccer</Sport>
    <Region>Schweiz</Region>
    <League>Challenge League</League>
    <EventDate>03.08.07 19:45</EventDate>
    <TeamA>Concordia Basel</TeamA>
    <TeamB>AC Lugano</TeamB>
    <Outcome>1:1 (1:1)</Outcome>
</Result>
```

Usability

The usability is quite good of RoboSuite. Due to the consistent design of the tools interfaces and the usage of understandable icons, it is quite easy to handle the program. The tool provides a great help function. Nearly each function has a help button which links to the correct site of the online help. Furthermore there is a detailed manual and a tutorial with many example files. The installation is quite simple and done by an installation wizard with few user interactions. Additionally an installation guide is available. The crawler can be extended in the functions by the usage of its API. Error messages are quite easy to understand. However a negative aspect is that some text in the graphic user interface is cropped. The menu entries and the buttons can not be adapted to the user's need.

Efficiency

The application needs a normal standard configured computer with Windows or Linux. The system is easy to handle and supports proxy servers. Additionally a forum is available as well as a tracking system which reports problems directly to the RoboSuite team. A negative aspect is that the crawler can trap into a crawler trap.

4.3 Visual Web Task 5.0

Producer	Lencom
Test version	Visual Web Task 5.0
Acquisition costs	121,25 EUR
Platforms	Windows
Homepage	www.lencom.com

Table 4.3: Overview of Visual Web Task

4.3.1 Introduction

The Visual Web Task consists of the following two tools:

- **Task Editor**: This tool is used for the configuration of the crawler.
- **Task Runner**: It allows the executing of the configured crawler. It can be started in an automated way over the command line.

The figure 4.10 illustrates the interaction between the two described tools.

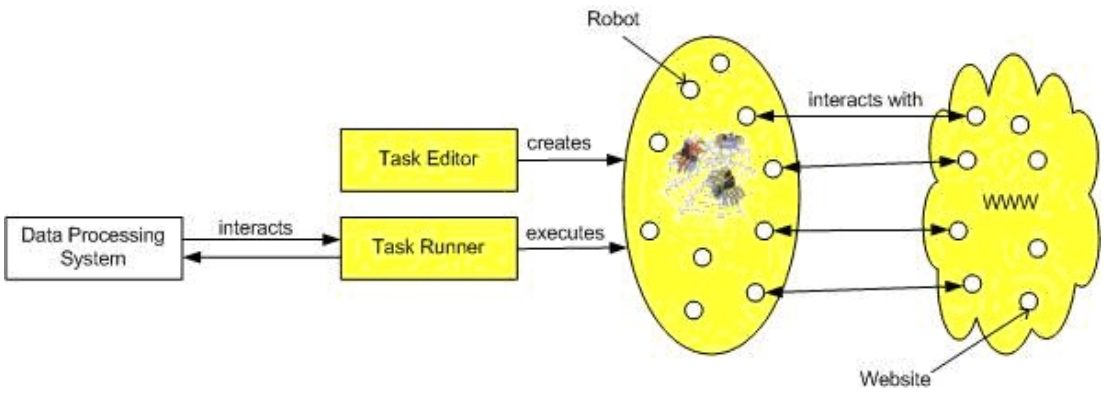


Figure 4.10: Architecture of Visual Web Task

4.3.2 Evaluation Results

Configuration

The configuration of the crawler is done by a wizard shown in figure 4.11. The first step is to choose a name for the configuration file of the crawler. The next step is to record the navigation to the website which contains the desired pattern. The table which contains the required information is selected by click on the desired row and by selecting several cells if necessary. Other rows are selected automatically. As the last step the output format can be chosen.
The data model is included in the scraping template so it is not possible to extract the data model separately. The scraping template can not be extracted.

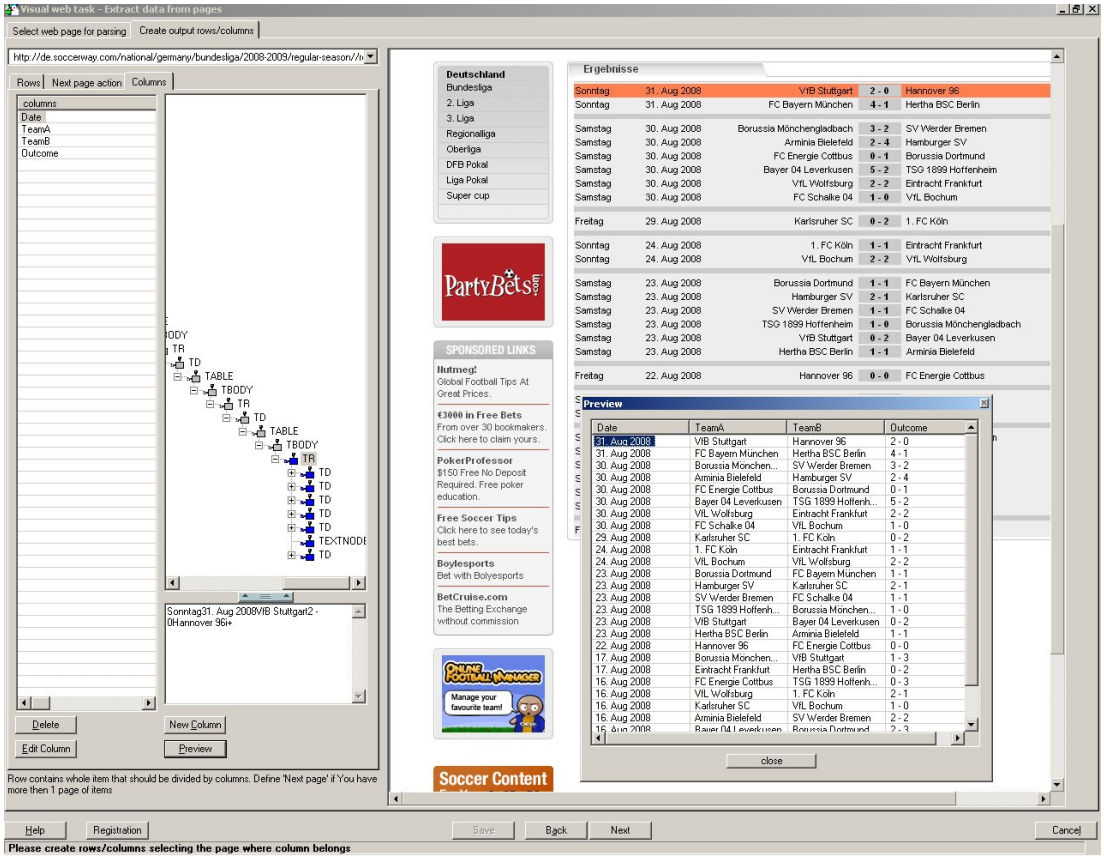


Figure 4.11: Creating of scraping template

A standard configuration for the crawler is not provided. The time zone can not be converted automatically. URLs can be added to the configured crawler. A parameterization of the URLs is not possible. The configured crawler can be started on the command line.

Error Handling

Few changes can be handled but if the location of the content changes the crawler is not able to find and to scrap the information. There is no automatic adaption. Errors are displayed in a pop up window. This is shown in Figure 4.12.
The tool does not provide any error administration. For example errors are displayed in a pop up window but there is no way to store them in a file or to debug them. The whole error administration has to be done manually. Furthermore there is no visual debugging tool available.

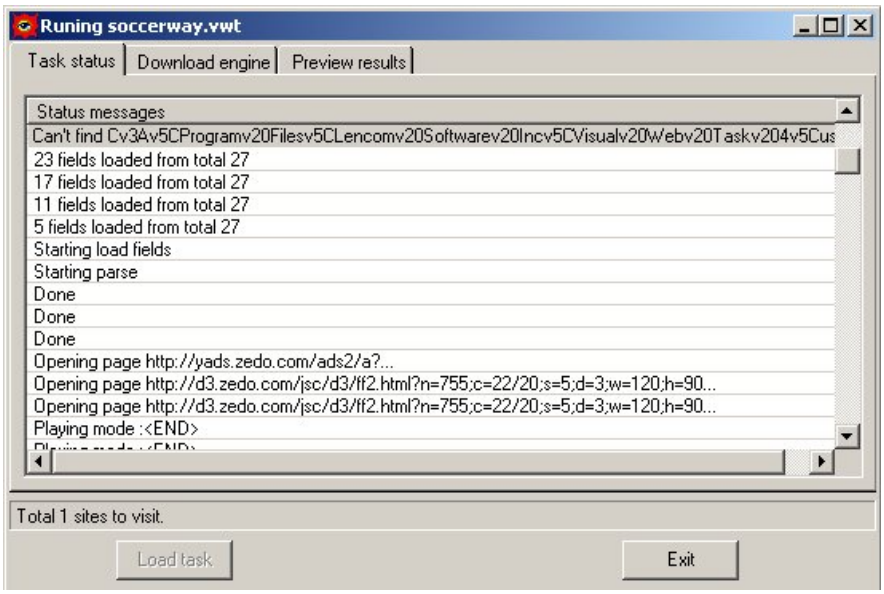


Figure 4.12: Error handling

Crawling Standards

The user-agent field contains the following information: *Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; iOpus-Web-Automation; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648).* The user-agent field captures the information which the crawler uses as his identity. For example this crawler acts in the WWW as Mozilla Firefox. It is not possible to change this information. The real identity can not be shown. The robot exclusion standard, the extended robot exclusion standard and the exclusion by meta tags are not supported.

Navigation and Crawling

The navigation to a certain page works the following way. Based on a starting page the navigation can be recorded by simply clicking the button "start-recording". The crawler records the usage of links, drop down menus and other navigation elements. The recording is stopped again by a simple click to "stop recording". The log file which contains the navigations can be displayed and shows each step. During the navigation steps waiting periods cannot be added.
The crawler Visual Web Task can fill out forms, choose options, checkboxes, radio buttons and fill out log in fields. The filled forms can be submitted. The crawler can handle HTTPS without any problems.
A next page pattern can be defined by adding a link to the crawling list. This "Next Page" action helps Visual Web Task to load the next page. For example this could be the next match day. Cookies are administrated automatically. There is no way to set, change or delete cookies.
Some disadvantages are that the crawler cannot handle Captchas and the program can not react on HTTP standard codes. A Captcha is a kind of challenge-response test to ensure that the response is not generate by a computer. Another negative aspect is that it is not possible to build parameterized links.

Selection

Information of HTML sites can be scraped inside the body part. The selection is done by HTML nodes (like <td>, <tr>). Meta and header information cannot be selected because they are not included in the body part. Empty cells of a table are scraped. There is no way to handle empty cells differently. Information placed on flash sites cannot be selected. HTML and XTML sites work smoothly.

Extraction

Figure 4.13 shows a website with the preview of the information which will be extracted. The crawler is not able to link different rows together by defined rules. For example at the website the events with the same kickoff date are listed among each other and the date field is filled out just at the first event. It would be desirable that the crawler fills out the other date fields with the appropriate information.

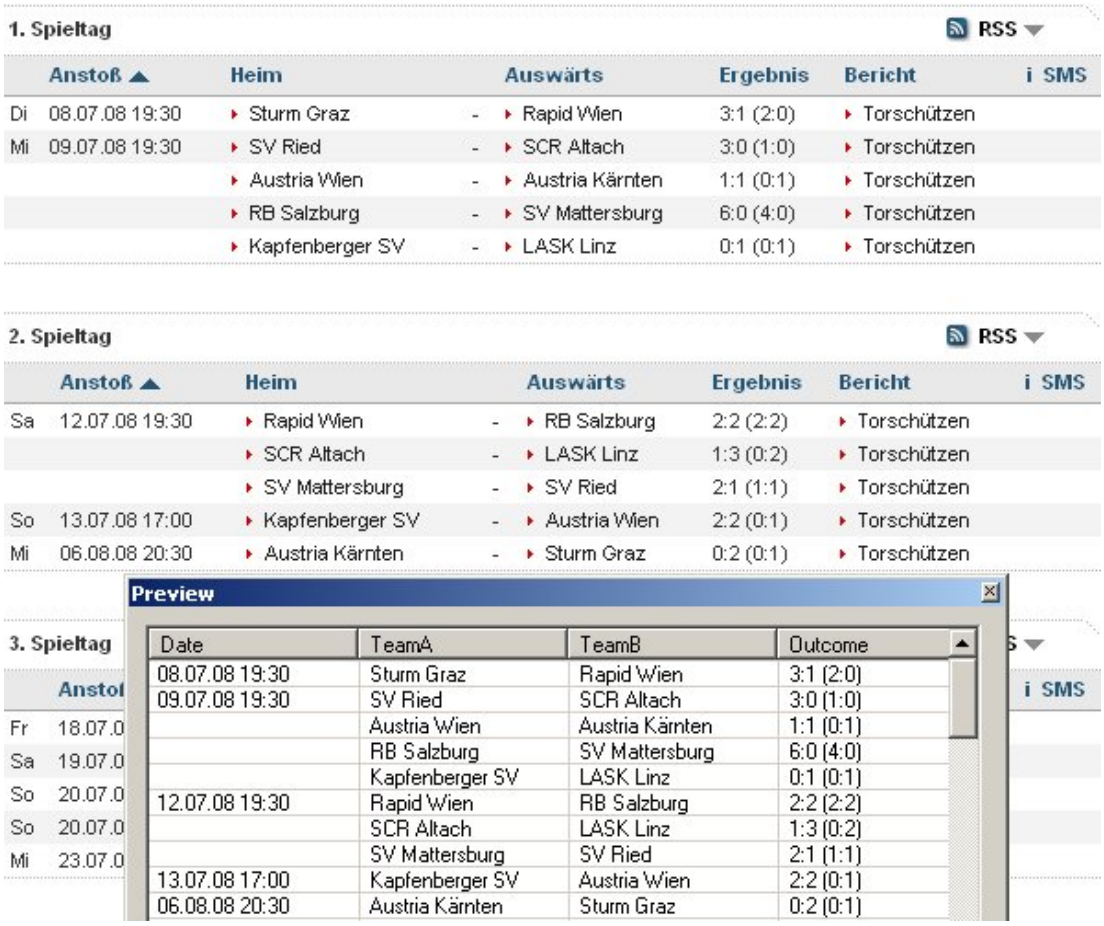


Figure 4.13: Extraction

It is not possible to set extraction conditions. For example distinguish between fixtures and results and scrap just the results. Another problem is that names cannot be transformed to a proper standard. For example it is not possible to change "Sturm Graz" to "Sturm".
The tool is not able to assure the quality of the scraped information if certain conditions are fulfilled like the date just contains digits and points but no characters. These useful functions have to be added by external tools.

File

The extracted information can be saved in files. For this file any name can be chosen but the file name can not be built with parameters. The scraped information can be stored as TXT or XML file. The customization of the structure of the stored file is limited. Furthermore, the extracted data can also be stored in an ODBC database.

Usability

It is easy to install and handle the tool. The installation is done by a wizard and the user interface is clear and easy to understand. To learn the handling a flash tutorial is available as well as a help file. Errors lead to an error message which is clear and explicit. It is not possible to expand or adapt the crawler.

Efficiency

The application needs a normal standard configured computer with Windows. The learning time is quite short. But the tool provides just main functions. Some important features are missing like the handling of proxy servers. The crawler is not able to handle crawler traps. There is no support service.

4.4 Web Content Extractor 3.1

Producer	NewProSoft
Test version	Web Content Extractor 3.1
Acquisition costs	129 USD
Platforms	Windows
Homepage	www.newprosoft.com

Table 4.4: Overview of Web Content Extractor

4.4.1 Introduction

Web Content Extractor is a web data extraction tool. The tool allows to create a project for a particular site with the similar structure of the pages. The data can be harvested from these target sites automatically and are exported to a variety of formats.
The Web Content Extractor consists of the following tools:

- **Web Content Extractor**: It is used for the configuration of the crawler.
- **Command Line Tool**: It allows to execute the crawler once configured.

The figure below illustrates the interaction between the described tools.

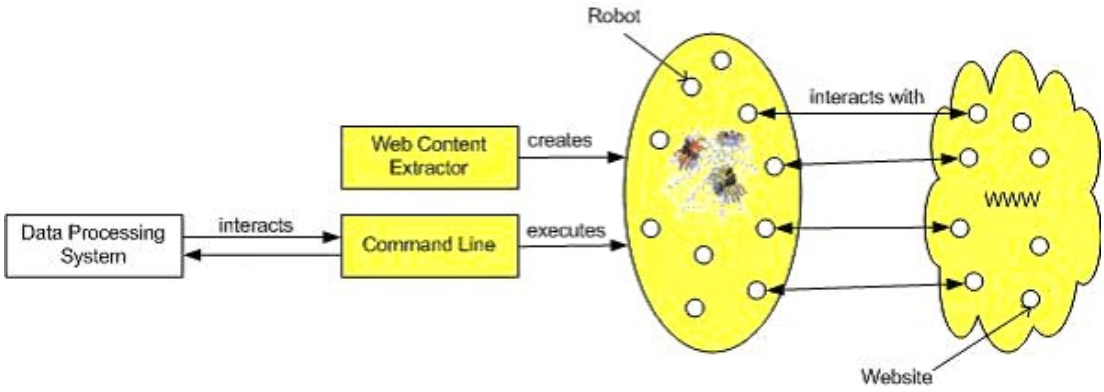


Figure 4.14: Architecture of WebContentExtractor

4.4.2 Evaluation Results

Configuration

The configuration is assisted by a wizard. For example the wizard allows to select the content which should be scraped by a simple click. The wizard is shown in figure 4.15. Fields can be marked as required. This means the scraping is successful if these fields are not empty.
The preview in figure 4.15 is divided in two parts. The left part shows the website and the selected content. The right part gives a preview of the scraped information. The data model can not be exported separately. It is embedded

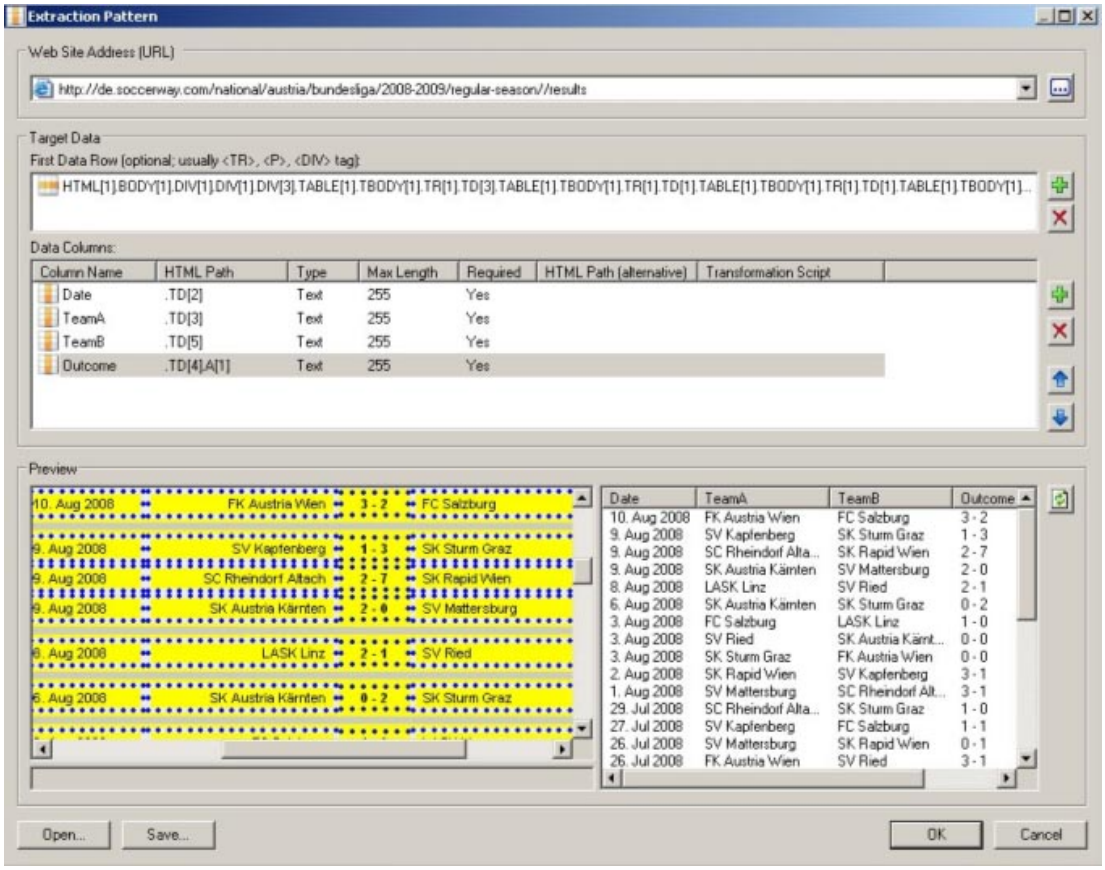


Figure 4.15: Scraping template

in the scraping template. A scraping template itself is created for one site. A template can be reused and adapted to other websites, e.g. by changing the links. A weak point is that no standard configuration is provided and that times zones can not be converted automatically.
It is possible to start the crawler from the command line. An external scheduler is needed to start tasks automated at different times or periods. The tool itself can not administrate different tasks.

Error Handling

The error handling is implemented quite badly. An error leads to a pop up window in the configuration program. If the crawler is executed in the command line a problem occurs because there is no pop up window. Pop up windows need a running configuration program. The same problem exists if tasks are executed automaticaly. This makes it difficult to find errors. Just empty XML pages indicate a problem during the crawling process.

Crawling Standards

The name of each crawler configuration can be chosen arbitrarily. Sitemaps can be found due to the tags (head, title) in the HTML code. The robot exclusion standard, the extended robot exclusion standard and the excluding by meta tags are not supported.

Navigation and Crawling

It is not possible to record the navigation to a specific site. There are two possibilities to specify the path to the required information. First the links can be parameterized and the second way is to start an automated crawling process and to specify the rules, the depth, and the waiting periods. For example follow the URL if it contains Austria or crawl maximum 3 subsections. The next page can be defined by setting the values of "follow links if the linked text contains".
The crawler is able to fill out forms and log in masks. Pop up windows do not disturb the crawling process. Cookies are set automatically and can be administrated by the internet explorer. Secure connections are handled smoothly. Captcha, Multitasking and HTTP standard codes are not supported.

Selection

The selection of information is done by the wizard shown in figure 4.16. A data record is selected in the left part. The different columns or rows can be named freely. The right part shows the automatically selected HTML node.

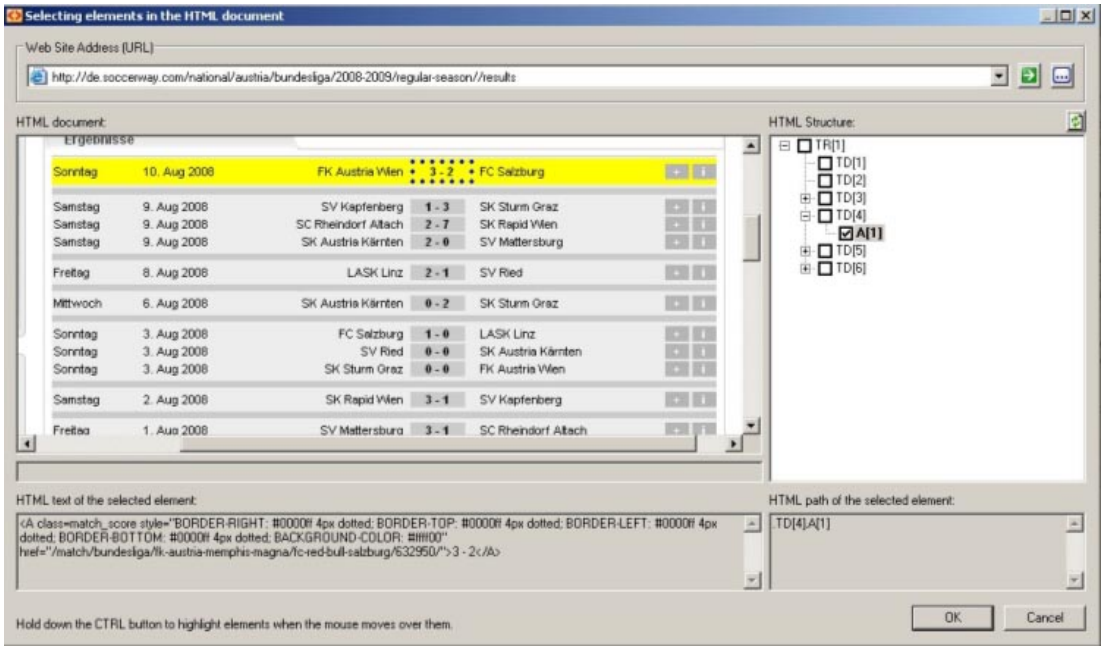


Figure 4.16: Selection of information

A configured data record is applied to other records. After this process all records are selected. This is shown in figure 4.16.
As described above information placed on HTML and XHTML pages can be scraped smoothly. Flash content can not be scraped. Meta data and header information capturing is not supported.

Extraction

A result filter (shown in figure 4.17) can be used for example for the extraction of specific table rows. The following table shows the construction of the website:

09.08.2008	SV Kapfenberg	SK Sturm Graz	1-3	Finished
09.08.2008	SK Austria Kärnten	SV Mattersburg	2-0	Running

Table 4.5: Sport information in a table

A row contains the key word "finished" if the game is over. The result filter below assures if the line contains the word finished. This filter allows the user to crawl just results of finished games. The result filter uses values to analyze the website. So with this filter it is not possible to distinguish between different formatted results for example just bold results are finished.

A transformation script can be added to convert different formattings and

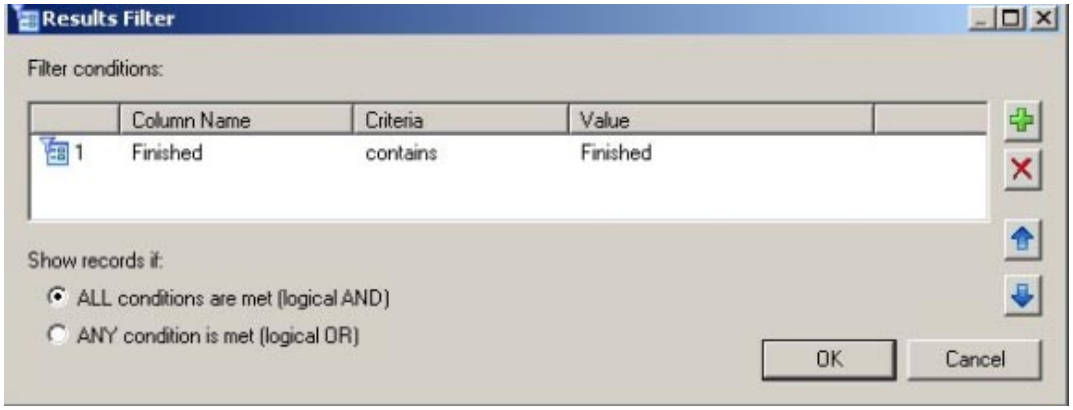


Figure 4.17: Result filter

spellings to coherent standards. The example in figure 4.18 shows the conversion of SK Rapid Wien to Rapid. The languages that can be used are Microsoft VBScript and Microsoft Jscript. This script can be used to modify data as it is extracted from a web page. The transformation script does not allow any output validation.

Figure 4.18: Transformation script

File

The filename can be chosen arbitrary. The date and timestamp can be added automatically to the filename. There is a wizard to specify the XML-file format. The Data Field box displays the information of the output file. In figure 4.19 a sample of the exported file is shown. It is possible to choose between the

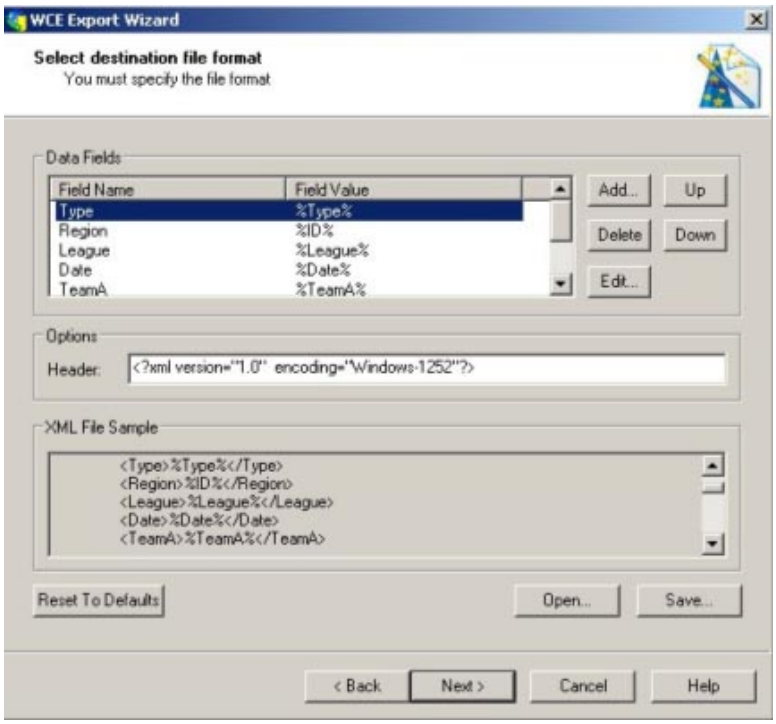


Figure 4.19: XML output configuration

following output formats: Microsoft Excel, TXT, SQL, HTML and XML. Another possibility is to write the scraped data direct in a database like Microsoft Access and MySQL.

Usability

The tool contains the major functions without any extras. This makes the tool clear and easy to handle. Furthermore it is possible to expand the tool. The user has no possibility to configure the menu to the own needs. The tool learning process is supported by a flash tutorial and a user guide. The installation is easy because a wizard is provided. A disadvantage is that the error handling is not supported.

Efficiency

For this simple tool a standard configured computer with Windows is sufficient. Other platforms like Linux or MacOS are not supported. The general usage of the crawler could be described as efficient because it is clearly designed and a telephone support as well as flash tutorials and a user guide are available.
The configuration of the proxy settings is quite simple because they are inherited from the internet explorer. The crawler is not able to handle crawler traps.

4.5 Web-Harvest 1.0

Producer	Vladimir Nikic
Test version	Web-Harvest 1.0
Acquisition costs	under BSD License
Platforms	OS Portable - needs Java Runtime Environment
Homepage	web-harvest.sourceforge.net

Table 4.6: Overview of Web-Harvest

4.5.1 Introduction

Web-Harvest is a Web data extraction tool. It can extract data from web pages and convert it to XML.

Web-Harvest uses XML files for the configuration. It describes a sequence of processes to convert HTML into usable XML.
Web-Harvest executes the following three steps:

- Download content from the specified URL.
- Convert HTML to XHTML content.
- Identify the desired pattern by XPATH.

4.5.2 Evaluation Results

Configuration

The crawler is configured by an XML file which is shown in the right part of figure 4.20. The data model is described in the XML file by XPATH expressions which identify the desired pattern. Web-Harvest supports the following three scripting languages: BeanShell, Groovy and JavaScript. These languages provide the whole range of programming structures like loops, if-else. As the whole program basically consists of a scripting language, no standard functions and standard configurations like templates are provided. For example, it is not possible to select the time zone. If this is necessary a function has to be programmed for this task.

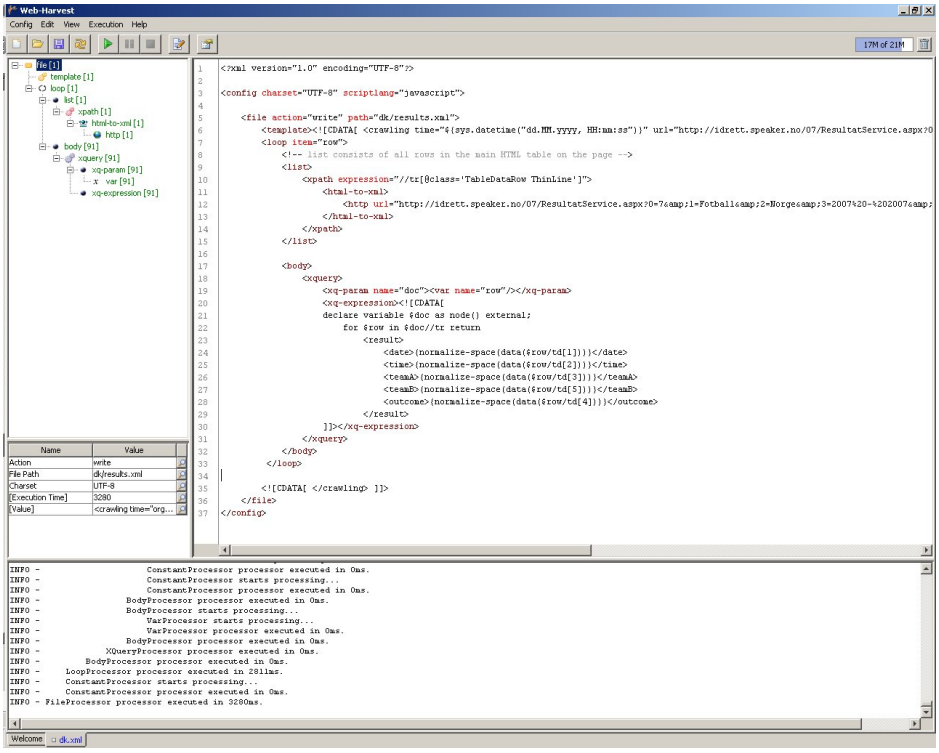


Figure 4.20: Web-Harvest configuration

Error Handling

As mentioned above the whole program just consists of a scripting language. No notifications are created by default. It is necessary to program them if notifications are required. This enlarges the configuration and programming effort.
The log view is shown in figure 4.20. The console is shown during the configuration and eases the programming. But the console does not help to handle

errors during the crawling process. Due to the reasons mentioned above no visual debugging or notification administration is provided.

Crawling Standards

The useragent field can be modified with the code below. It identifies itself to *www.google.com* as *DataCrawler*. This is done by writing the name "DataCrawler" in the HTTP header area name.

```
<http url="www.google.com">
    <http-header name="User-Agent">
      DataCrawler
    < /http-header>
< /http>
```

By standard Web-Harvest pretends to be a Firefox browser. The program does not support any exclusion standard.

Navigation and Crawling

The log in is done by the script below. An automated log in on the page *http://www.nytimes.com/auth/login* is done by the following script with the username web-harvest and the password web-harvest. Forms can be filled in the same way.

```
<http method='post' url='http://www.nytimes.com/auth/login'>
    <http-param name='USERID'>web-harvest< /http-param>
    <http-param name='PASSWORD'>web-harvest< /http-param>
< /http>
```

It is not possible to record the navigation to the desired pattern because the scraper uses XML configuration files. URLs can be parameterized. For example 'www.soccer.com/' + region + '/' + league. In this case region and league are used as variables.

Waiting periods and HTTP-standard codes are not supported. The reasons are mentioned in the chapters before.

A pattern can be defined which is used to find the next page automatically. The crawler is not disturbed by pop up windows because absolute paths are used.

Selection

The following code iterates through a table row for row. The first column contains the date, the second one the time and so on. The content is selected by an XPATH expression.

```
<xquery>
<xq-param name='doc'><var name='row'/></xq-param>
<xq-expression><![CDATA[
    declare variable $doc as node() external;
    for $row in $doc//tr return
      <result>
        <date>normalize-space(data($row/td[1]))</date>
        <time>normalize-space(data($row/td[2]))</time>
        <teamA>normalize-space(data($row/td[3]))</teamA>
        <teamB>normalize-space(data($row/td[5]))</teamB>
        <outcome>normalize-space(data($row/td[4]))</outcome>
      </result>
  ]] ></xq-expression>
</xquery>
</body>
```

The crawler supports XHTML and HTML but can not handle flash content. Meta data and header information can be selected by the crawler.

Extraction

The extraction with Web-Harvest works smoothly. It is possible to assure conditions and validations. The crawler can handle strings and substrings which make the crawling process efficient because unimportant information can be excluded. Formatting differences can be identified and balanced but lead to a higher configuration effort.

File

The name and the path can be chosen freely. Due to the scripting language it is possible to use variables in the name. The following example shows how the crawler creates a folder containing a file named with the name of the region and the league. This can also be parameterized.
<file action='write' path='results/Austria_Bundesliga.xml'>
The output format is an XML file. The structure can be chosen freely by the programmer. At this time there is no possibility to transfer information directly to databases.

Usability

The whole crawler is based on a scripting language. It is absolutely necessary to have knowledge in programming with XPATH and XML. The crawler is an open source tools. So the code is freely available. Therefore it is possible to expand it any time but this is quite time expensive as well as programmers are needed to do this.
The crawler need not to be installed because the programm extraction is sufficient. User manuals are available as well as examples and java docs. The error handling is a bit difficult. By standard there are no notifications. Errors can just be detected by the XML result files when they are empty because they are created when the script code can not be executed correctly. If notifications are needed a programmer has to add them.

Efficiency

The tool runs smoothly on a standard configured computer. It is written in Java. No special platform is needed to run the tool. It is necessary to learn the usage of the tool. Since it is based on a scripting language it is necessary to learn the syntax of the configuration files. Experiences with XPATH are advisable. The crawler can handle proxy servers but crawler traps lead to problems because the crawler does not identify them.

4.6 WebSundew Pro 3

Producer	SundewSoft
Test version	WebSundew Pro 3
Acquisition costs	699 USD Single Pro License
Platforms	Windows
Homepage	www.websundew.com

Table 4.7: Overview of WebSundew

4.6.1 Introduction

The WebSundew tool consists of three tools. It supplies a tool for creating a crawler, a command line tool to launch the crawler and a tool to schedule the crawler.

- **WebSundew Pro**: The user can configure a crawler for certain tasks over a graphical user interface. The complete navigation, selecting and extracting process is defined here.
- **Console**: The console is a kind of command line tool to execute the crawler.
- **Scheduler**: This tool allows to search and extract data periodically and automatically.

Figure 4.21 illustrates the interaction between the tools described before.

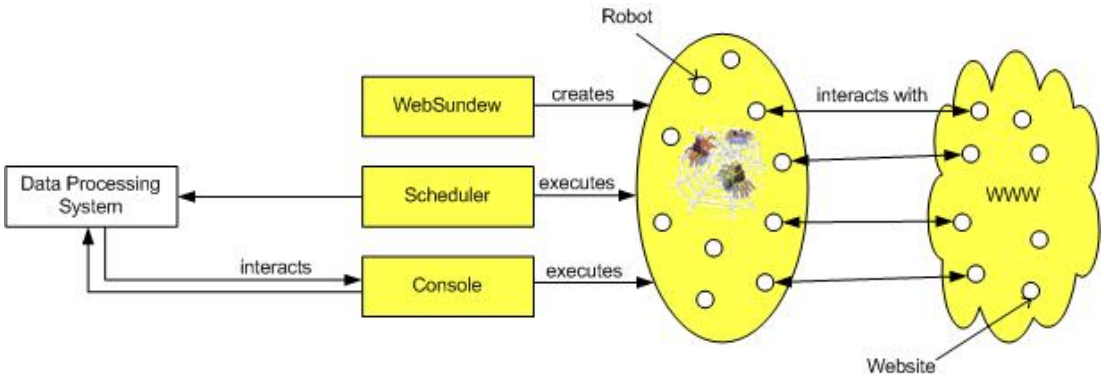


Figure 4.21: Architecture of WebSundew

4.6.2 Evaluation Results

Configuration

A data model is created during the selection of information from web pages. This is done by clicking on the content and assigning a name for the data model. Scraping templates are created for a certain page. It is not possible to export and reuse them. Furthermore there is no way to specify standard configurations. This leads to a time consuming for the configuration because models for similar pages have to be reconfigured again.
Links are administrated by the Web Macro. The Web Macro contains the navigation path to the desired pattern. The crawler is not able to recognize time zones and convert times from pages in a selected time standard.

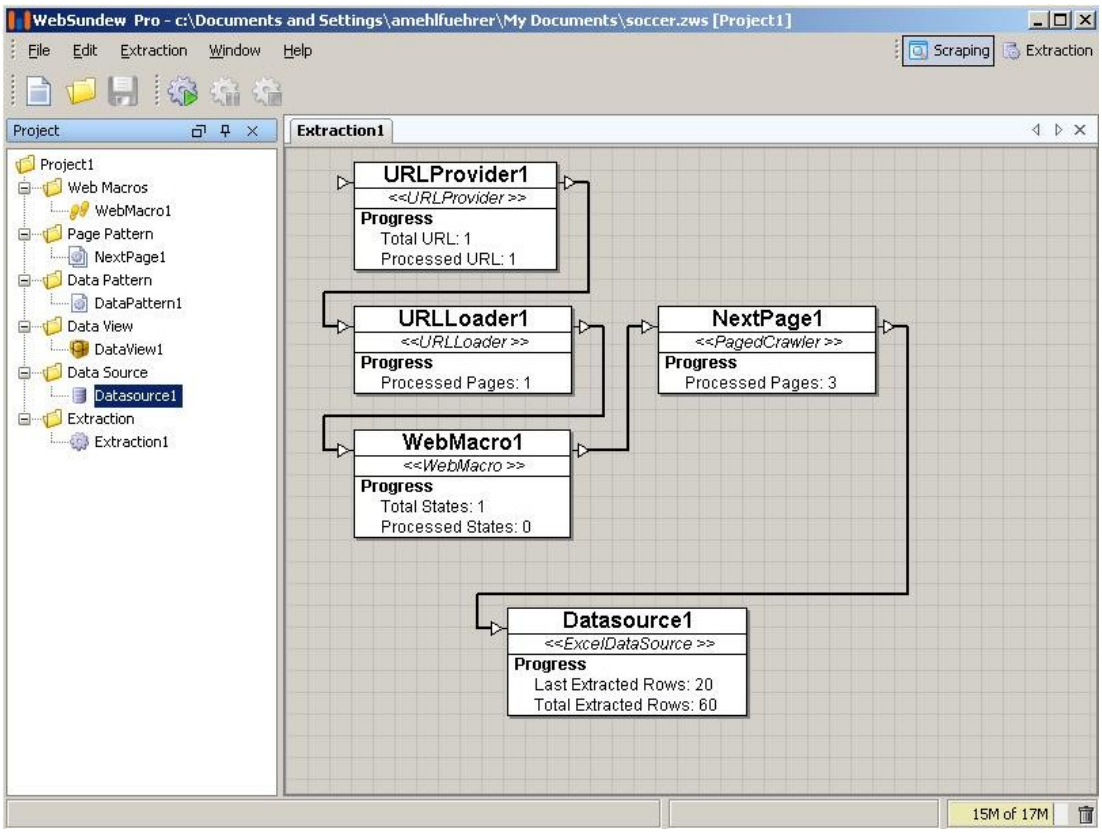


Figure 4.22: Extraction with WebSundew

The complete extraction process is shown in figure 4.22. An URL is loaded followed by the Web Macro which navigates to the desired pattern. In the example the results of three pages are extracted by using a NextPage pattern. Altogether there will be extracted 60 games results in the example.

Error Handling

The error handling is poorly implemented. Errors are written in a log view. There is no way to debug configured tasks. Notifications are generated automatically. For example if a page was crawled error-free a notification in a log view is created. This contains the notification time, date, source and a description of what was done. It is not possible to configure which information is logged and when it should be logged.

Crawling Standards

The name of the crawler cannot be chosen. Sitemaps are found due to the tags in the HTML code. A sitemap contains a list of links within a domain. The robot exclusion standard, the extended robot exclusion standard and the exclusion by meta tags are not supported.

Navigation and Crawling

The navigation can be recorded. It is possible to store how forms are filled, and how checkboxes and options are chosen. Additionally, parameters can be used to fill out a form with different information. It is possible to define where the link to the next page is located. In WebSundew this is called the NextPage pattern. This enables visiting pages consecutively; for example different event times of a league at different pages. Cookies are administrated automatically. A connection via HTTPS works smoothly.

Selection

The selection of information structured in tables works almost without problems. Problems occur if tables contain empty cells which means they are equal to the cell above. Such a structure is shown in table 4.8. In this example the date and the time of both events are equal. The crawler is not able to identify the date and time of the second entry of the context.
The selection takes place line-by-line. This is called data *iterator pattern*. Furthermore corresponding content can be selected from a detail page. This selection is called detail data pattern.

24.05.2008	18:00	Bayern München	Leverkusen	2:0
		Juventus Turin	AC Milan	1:0

Table 4.8: Example empty cells

Problems appear if parts of strings need to be selected. For example: *Austria - Bundesliga - Spieltag 1.* The crawler can only select the whole string but not just some words of the string.
Further problems occur by crawling flash pages. They are not fully displayed and it is not possible to select content from them.

Extraction

Some negative aspects stick out during this extraction. It is not possible to match different notations with the target system. The tool does not allow an extraction of table parts. Furthermore there is no way to ensure that a table really contains the expected information (for example is the date equal to the actual date).

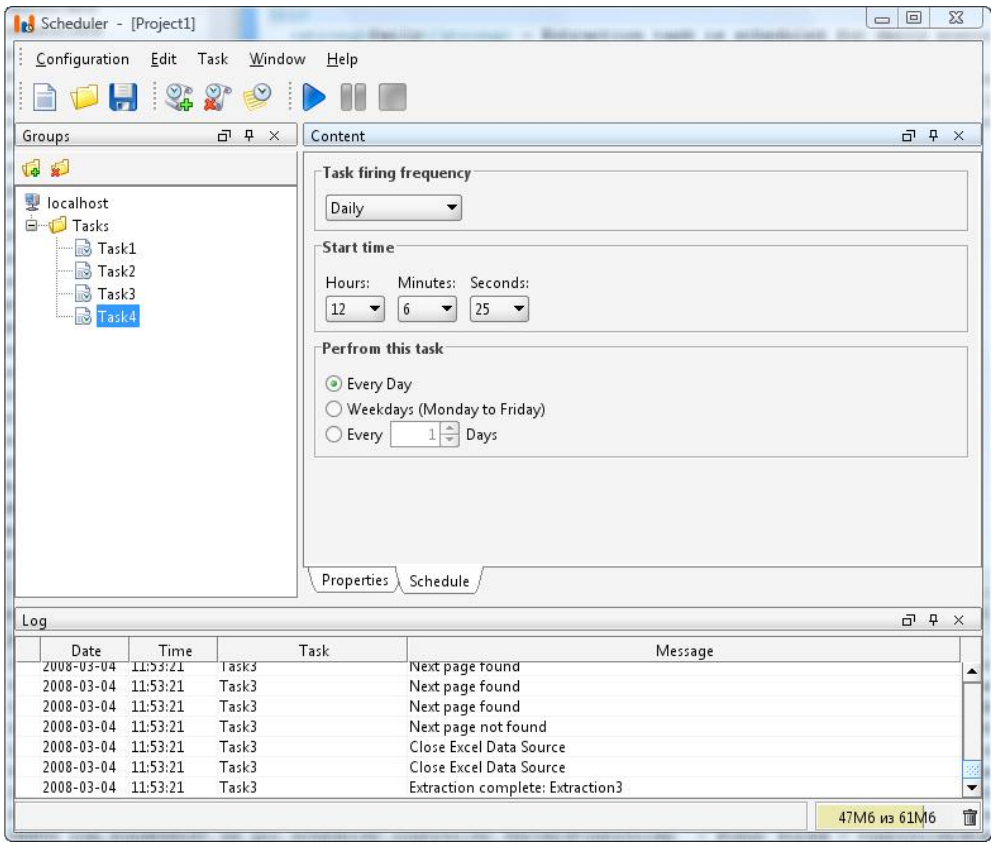


Figure 4.23: Task automation with scheduler

A configured scraper can be started by the command line tool or by the Scheduler. The Scheduler (illustrated in figure 4.23) can be used to automate tasks. Tasks can be started periodically, on certain days and times.

File

Figure 4.24 shows the configuration of the crawler output. The name of the output file can be customized with variables and templates. Possible file formats are XML, CSV and XLS.

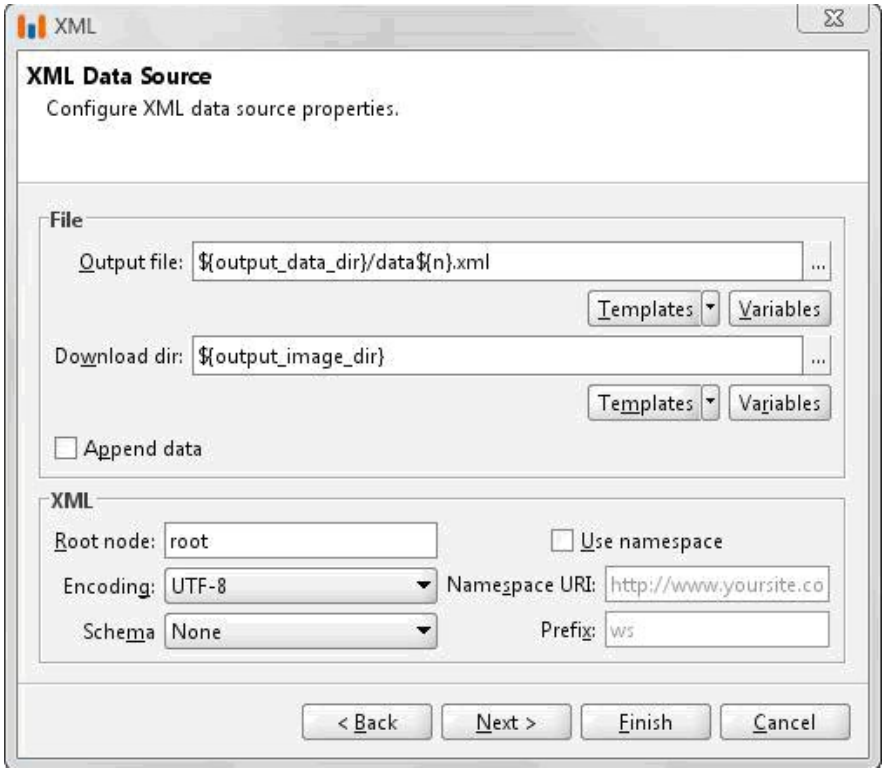


Figure 4.24: Crawler output configuration - XML

The tool does not support any database. According to the online help the following formats should be supported: MS SQL, MySQL, Oracle and other databases if there is a suitable JDBC driver. The screenshots in the online help show the support but we were not able to find this function in the tool.

Usability

The interface of the tools is quite clear. The installation is simple and done by a wizard. There are flash tutorials available to learn how to handle the tool.
Negative aspects are the missing possibilities to adapt the menu entries by the users and to enlarge the tool. Furthermore the help documentation is incomplete because some help pages are blank.

Efficiency

The crawler needs a normal standard configured computer with Windows 2000, XP or Vista. The system is easy to handle and supports proxy servers. The learning periode is quite short. The configuration could be done faster by the usage of scraping templates. A forum is available to post questions. There is also a possibility to trace problems. A negative aspect is that the crawler can be caught by a crawler trap.

4.7 Web Scraper Plus+ 5.5.1

Producer	Velocityscape
Test version	Web Scraper Plus+ 5.5.1
Acquisition costs	749 USD
Platforms	Windows 2000, XP, Vista
Homepage	http://www.velocityscape.com/

Table 4.9: Overview of Web Scraper Plus+

4.7.1 Introduction

WebScraperPlus+ is a web extraction and automation suite. It takes data from web pages and puts it into a database or an Excel spreadsheet. The figure 4.25 illustrates the architecture.

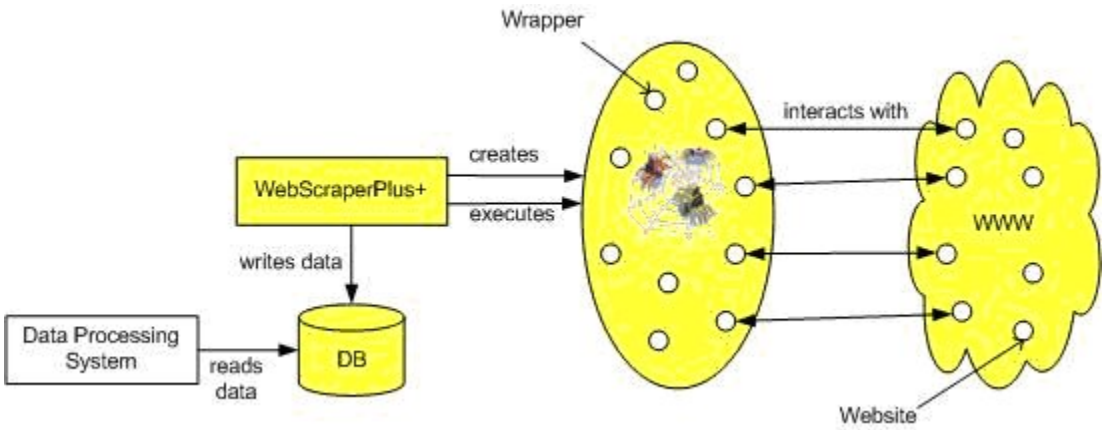


Figure 4.25: Architecture of WebScraperPlus+

4.7.2 Evaluation Results

Configuration

During the configuration of a scraping template the data models are created. WebScraper calls them dataset. The dataset is embedded in the scraping templates. So a separate importation or exportation is not possible. The embedded data model is shown in figure 4.26. The whole process is done by pointing and clicking. This makes the configuration quite simple. The scraping process can be automated for similar websites just by adding the links to the template. A

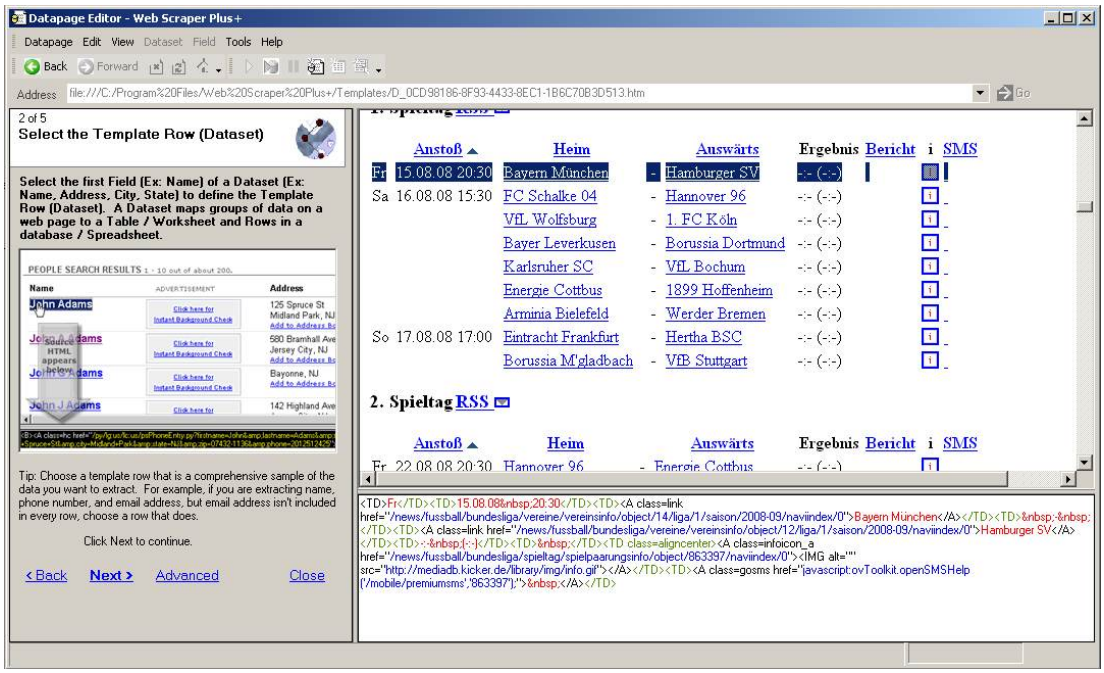


Figure 4.26: Selection of a dataset

standard configuration is not available, but a wizard is provided to configure the main steps quite fastly. The different crawler jobs can be configured by a wizard and they are displayed in a special task overview. A console allows to run different tasks.

Error Handling

It is possible to debug the process of pattern selection to find errors. The following figure 4.27 shows a data record which is stored exactly in one row. This is used to locate errors. The beginning of the line is marked green. Begin and end tags which embed the text are marked red and are defined by the user. The desired text pattern is marked blue.

The crawler handles smaller changes on websites. A significant change of a site

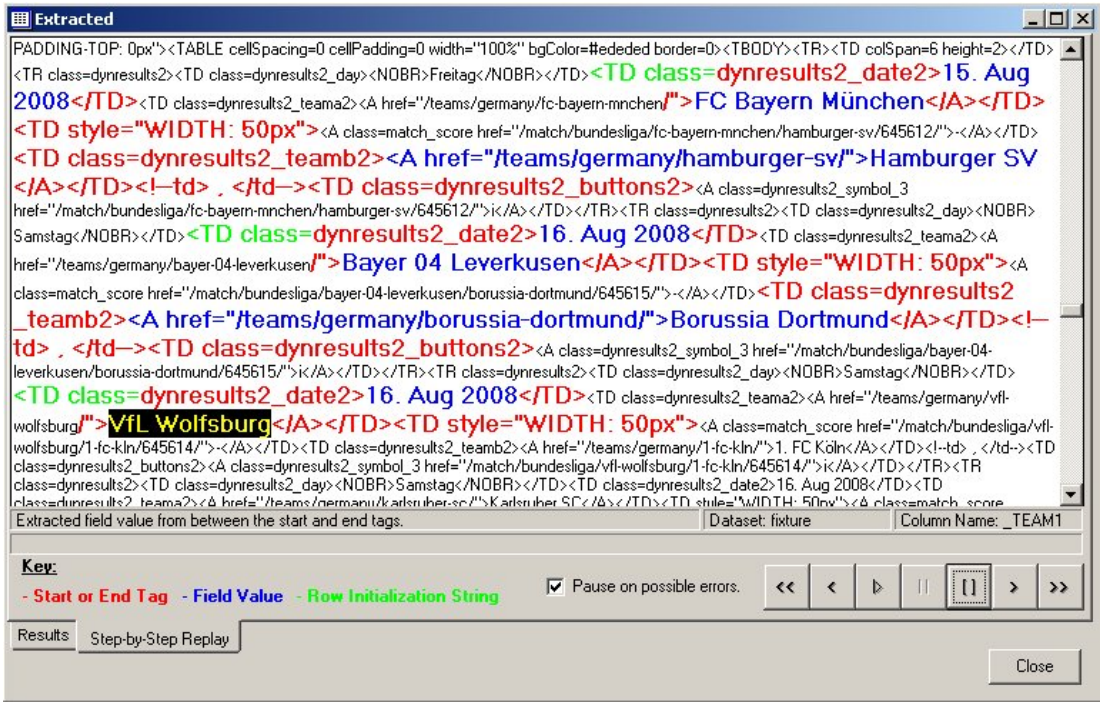


Figure 4.27: Debugging

leads to errors. The crawler protocols each step in a log file but it is not possible to create additional notifications for the user. A visual debugger is provided. It is a step by step replay of the process.

Crawling Standards

The crawler is able to pretend to be another user agent. This is done by changing the value of the user agent field. By default *Mozilla/4.0* is chosen. The robot exclusion standard and all extended versions are not supported. The exclusion by meta tags is not possible.

Navigation and Crawling

The crawler uses direct links. A direct link is the absolut path to the desired pattern. So it is not necessary to record the navigation to the desired pattern. The configuration of the crawler can be done easier by the parameterisation of links. It is possible to set restrictions like for hostnames, filenames etc. The crawler is able to process different tasks at the same time in an automated way. The next page can be crawled automatically by the definition of a next page pattern.
WebScraperPlus is able to fill out forms. Therefore a "form submit task" with the desired information can be created. The crawler can do log in on websites. The user has to create a task which submits forms. Cookies, pop up windows and secure connections are handled smoothly. Cookies can be stored separate and viewed. Captchas, waiting periods and HTTP standard codes are not supported.

Selection

The selection of the information works by clicking on the desired pattern. The HTML code before and after the text pattern is used to locate the desired text on the page. Longer strings mean a better crawling result because it can be located easily. If the strings are short it can happen that the crawler can not identify it explicitly and is not able to crawl the desired pattern. The selection of table content works smoothly as well as the crawling of HTML and XHTML sites. Due to the fact that HTML code is used to find the required information, it is not possible to crawl flash pages. The selection of the header information is supported but capturing of meta data is not possible.

Extraction

The extraction is efficient because just the desired pattern is scraped and different formattings can be transformed in a unified company standard. Extraction conditions and the validation of information can not be configured.

File

The user can choose the name of the output file arbitrarily but the output file format is limited to Excel and databases. XML is not supported. Hence it is impossible to change the structure and the formatting of the scraped information. The following databases are supported: SQL Server, MySQL, delimited text files, Access, and others with ODBC drivers.

Usability

The tool is easy to configure because wizards are provided. It is useful to have knowledge in SQL because the results can be stored in a database. However a broad range of help functions is available like online help, video tutorials, faqs, forum, examples and additional training packages. The Microsoft .NET Framework 2.0 has to be installed. Then the crawler can be installed. The installation process is done by a wizard. The tool can not be extended or adapted to the user's needs.

Efficiency

It is easy to automate the processes because URLs can be readout from an Excel file. So it is possible to command the crawler to gather all URLs which are contained in an Excel file. For the crawler itself a standard configured computer is adequate. The software supports only Windows systems. There is a broad support because an additional support package can be purchased as well as free forums are provided.
The time to learn the handling is quite short because a lot of learning materials like video tutorials are provided. The configuration of a proxy server is easy because all settings are inherited from the internet explorer. The crawler can not handle crawler traps.

4.8 WebQL Studio 4.1

Producer	QL2
Test version	WebQL Studio 4.1
Acquisition costs	Annual: 12,000USD, Perpetual: 48,000USD
Platforms	Windows, Linux
Homepage	www.ql2.com

Table 4.10: Overview of WebQL

4.8.1 Introduction

WebQL Studio is a software tool for extracting data from electronic sources like web pages. For this extraction a programming language named WebQL is required. WebQL is an SQL dialect.
The syntax of a selection statement can contain the following clauses:

- **select clause:** The select clause is mandatory, all other clauses are optional. It is followed by one or more output expressions. This is the information which is scraped.
- **into clause:** It describes where to send output data.
- **from clause:** It describes where to retrieve data and the format of that data.
- **where and having clauses:** It can filter the output records.
- **group by and sort by clauses:** It can aggregate and sort data.
- **conforms clause:** It can normalizes the data with a given schema.
- **terminate clause:** It can halt the query when a condition is fulfilled.

Below there is a syntax definition of the selection:

select as [*select_alias*][*unique*]
output_expr [*as output_alias*][, *output_expr*2[*as output_alias*2]]...
[*into destination*]
[*from source*]
[*where filter_expr*]
[*group by group_expr*[, *group_expr*2]...]
[*having filter_expr*]
[*sort by sort_expr*[, *sort_expr*2]...]
[*conforms to schema*]
[*terminate*[*with term_with_expr*] *if term_if_expr*]

4.8.2 Evaluation Results

Configuration

The configuration is done with the programming language WebQL. Figure 4.28 shows an example of the configuration.

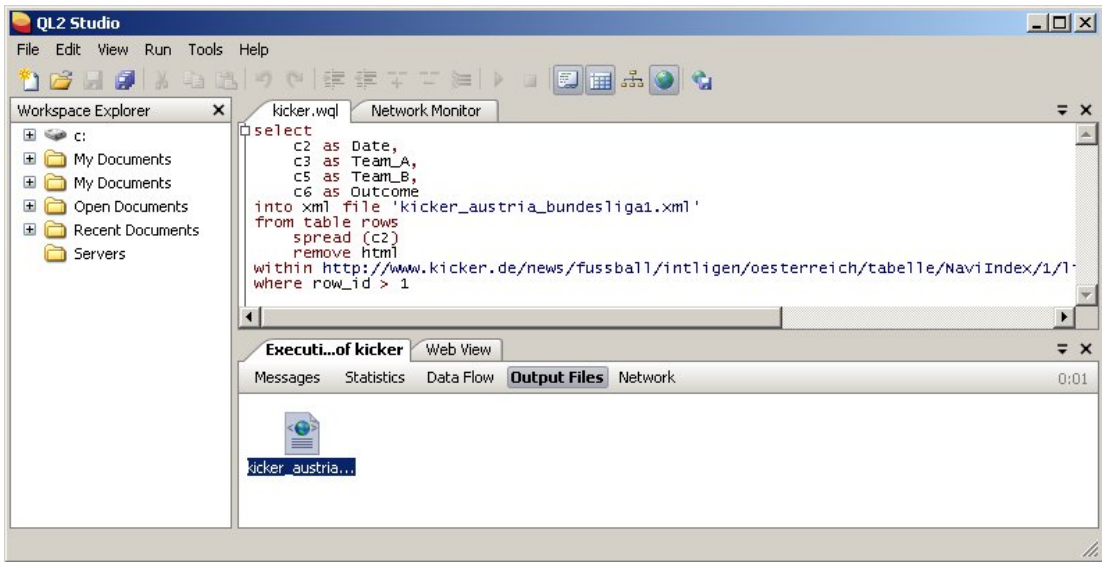
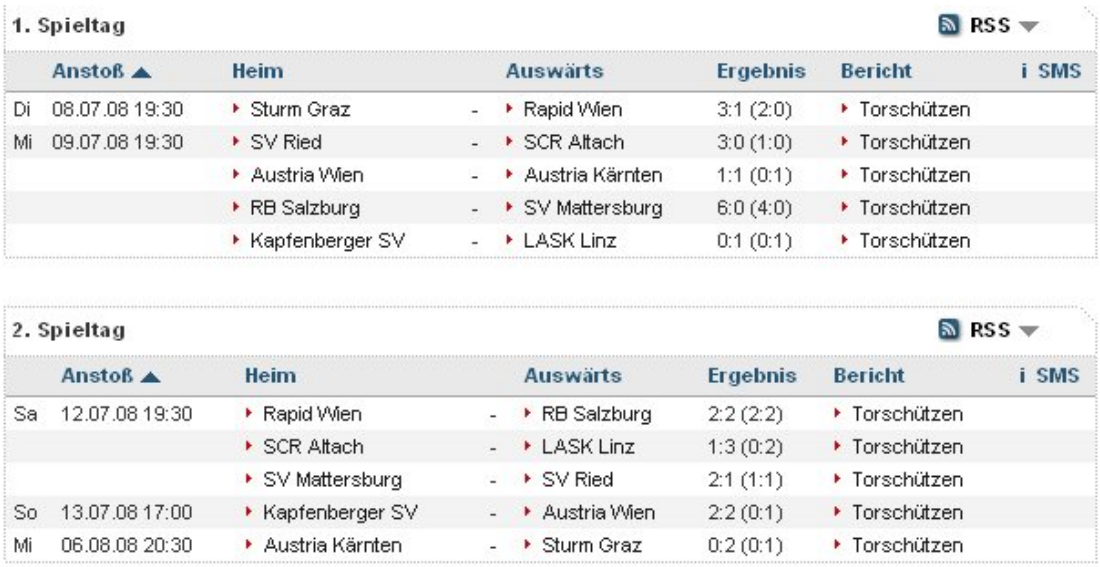


Figure 4.28: WebQL Studio configuration

The select part shown in figure 4.29 (initiated with the word select) contains the definition for the information which should be scraped. The following information should be extracted: date, team names and the result. The information is located in a table. The date (c2) can be found in the second column. The columns 3 and 5 contain the names of the teams (c3, c5) and column 6 contains the result (c6). The gathered information is stored in a file by the command *into XML file*

'kicker_austria_bundesliga1.xml'.

The command *spread (c2)* fills out the date cell automatically because on the



1. Spieltag

RSS

	Anstoß	Heim		Auswärts	Ergebnis	Bericht	i SMS
Di	08.07.08 19:30	Sturm Graz	-	Rapid Wien	3:1 (2:0)	Torschützen	
Mi	09.07.08 19:30	SV Ried	-	SCR Altach	3:0 (1:0)	Torschützen	
		Austria Wien	-	Austria Kärnten	1:1 (0:1)	Torschützen	
		RB Salzburg	-	SV Mattersburg	6:0 (4:0)	Torschützen	
		Kapfenberger SV	-	LASK Linz	0:1 (0:1)	Torschützen	

2. Spieltag

RSS

	Anstoß	Heim		Auswärts	Ergebnis	Bericht	i SMS
Sa	12.07.08 19:30	Rapid Wien	-	RB Salzburg	2:2 (2:2)	Torschützen	
		SCR Altach	-	LASK Linz	1:3 (0:2)	Torschützen	
		SV Mattersburg	-	SV Ried	2:1 (1:1)	Torschützen	
So	13.07.08 17:00	Kapfenberger SV	-	Austria Wien	2:2 (0:1)	Torschützen	
Mi	06.08.08 20:30	Austria Kärnten	-	Sturm Graz	0:2 (0:1)	Torschützen	

Figure 4.29: Website with results

website the date is filled out just ones for each day. Furthermore rows like the headline can be excluded. *Remove HTML* excludes disturbing HTML code in cells.

URLs of web pages with the same structure can be added to this query. A scheduler is available to automate the crawling process of websites. This allows to scrap pages at a certain time or time periods.

Error Handling

A visual debugging mode is not available. Queries can be debugged with the *out_log* function which adds the query result to a log file. The protocol function is realised by log files. It is possible to monitor a query using the SOAP method *getJobStatus*. This method shows the status of the executed queries. After the execution of a query two files are generatedat at the WebQL server. The first is the result file and the second one is a diagnostic file with information about errors.

Crawling Standards

The user-agent field can be chosen arbitrarily. So the crawler can appear for example as Firefox. WebQL normally masquerades as a version of Internet Explorer 6.0. This default value can be easily changed to other values. In this example

WebQL identifies itself to www.kicker.de as Mozilla web browser.
*select * from www.kicker.de*
user agent 'Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:0.9.8) Gecko/20020204'
Sitemaps can be found due to the tags (head, title) in the HTML code. The robot exclusion standard, the extended robot exclusion standard and the exclusion by meta tags are not supported.

Navigation and Crawling

It is not possible to record the crawler navigation, because the configuration is done by a query. The following sample query adds a depth limit of 3. All pages which can be reached from the startpage over 3 links can be visited:
crawl of http://www.kicker.de
to depth 3
following if url matching 'austria'
Some websites offer links to alternate between the different gaming days. WebQL uses these links with the following code:
following if url_content matching 'next matchday'
WebQL handles cookies automatically. Cookies can be set manually as well. The default cookie policy can be changed. Forms can be filled out. Therefore the name of the text field where the information should be filled in is required. If there are more forms on one page, WebQL numbers them consecutively. The log in to pages works similar. Secure connection over HTTPS works smoothly.
The WebQL studio can perform one query at a time. A WebQL Server is needed to perform more queries at a time. The WebQL Server provides a queue for the execution of more queries. It is possible to limit the number of active queries at a time. After reaching this limit the server sets a waiting period before starting further queries.
The crawler can not handle Captchas and the program can not react on HTTP standard codes. If an HTTP error code is retrieved, WebQL produces an error message.

Selection

There are three ways to select information on websites for the crawling process. The first method is to select content from tabels. WebQL numbers the tables so that each table has an own identification number. This allows an exact indication of the place, where the required information can be found. The number of a table

can be indicated for the crawling process. It works the same way for columns and rows because they are numbered too.
The second method is to specify the HTML code before and after the desired pattern. For example the country and the sport league of the event should be selected. This pattern is embedded in the HTML code as followed:
<h2 class='titelohnebildsaison'>Österreich - tipp3 - Bundesliga - Spieltag/Tabelle </h2>
The selection is described in the following way:

select
    item1 as Country,
    item2 as League
from
    pattern '<h2 class=titelohnebildsaison>(.+?)-(.+?) - Spieltag/Tabelle </h2>'
within
    http://www.kicker.de/news/fussball/intligen/oesterreich/tabelle/liga/1705/

The first appearance of *(.+?)* is equal to the country *Österreich*. The second one stands for the league *tipp3 - Bundesliga*.
The last method is to select content by the usage of regular expressions. This is quite similar to the method described before. Instead of a text string before and after the extracted text a regular expression is used.
More than one selection clause can be connected together with various different joins. The header information can be easily retrieved by the following query:
*select * from headers within http://www.kicker.de/*
    head
The crawler can not handle flash content.

Extraction

The condition of an extraction can be defined in the where clause. For example if only the fixture of a soccer event should be gathered the following can be set: *where Outcome = '-:- (-:-)'*.
The times on a website can be transformed to a desired time zone by the aid of the *reformat_datetime* function. This allows also the transformation of different formattings into one proper standard. The following example converts *Thursday, August 23rd, 2008 20:20:00* into *23.10.2008 12:20:00 Europe/London*. This is done by a pattern which contains the format of the source date on the website, the target format and the target time zone: *reformat_datetime('Thursday, Au-*

gust 23rd, 2008 20:20:00', 'fmDay, Month ddth, yyyy hh24:mi:ss', 'mm.dd.yyyy hh24:mi:ss','America/Los_Angeles', 'Europe/London')
It is also possible to extract substrings of a string. It is not possible to validate extracted information.

File

WebQL can output data to URLs at different locations:

- file: writes the data to a local file. Directories that don't exist are created automatically.
- ftp: uploads the output document to a server using the FTP protocol.
- mailto: sends the output document as an e-mail (see below).

The default output format is CSV. Other formats are available, including Excel and XML.
An INTO sub clause is needed to output data. This example stores data in an XML file. *into XML file 'output.xml'*
WebQL allows writing the extracted information into a database through ODBC. The destination is: name of the table @ connection. For example:
into 'dbtest'@'odbc:Driver={Microsoft Access Driver (.mdb)};DBQ=c:\test.mdb;'*

Usability

The QL2 server provides an API which allows the developer to embed the data extraction and transformation capabilities in custom applications. The tool provides a great documentation with a lot of simple and understandable examples. The usability of the tool is quite good, but a user should bring knowledge in regular expressions and SQL. The installation is quite simple and done by an installation wizard with few user interactions. A database has to be installed in addition to the tool.

Efficiency

The application needs a normal standard configured computer with Windows or Linux. The system is easy to handle and supports proxy servers. SQL knowledge and experience with regular expressions is advisable for the person who configures WebQL. QL2 provides a basic and advanced training for the software users. A query is excecuted fast and the time to configure a site is short if the user knows the main commands. The crawler is not able to handle crawler traps.

5 Tool Comparison

The tool comparison section describes first the general strength and weaknesses of all evaluated web crawler tools. Afterwards the tools are compared.

5.1 General Strengths

This subsection describes the general strengths based on the evaluated web crawler tools.

Good for gathering lots of information
The three tools RoboSuite, Lixto Visual Developer and WebQL provide good features for gathering information from web pages. The crawling process can be compared with a vacuum cleaner gathering all information from the web which matches predefined rules. A crawler is not an intelligent system. They just do what they are configured for.

Various output types of the generated information
All evaluated tools allow to save scraped information in XML files except Web Scraper Plus. Additionally each tool provides some other output formats like TXT, CSV, SCSV. Furthermore some tools support the connection to databases like Access, Oracle, MySQL and so on.

Good support for different forms of navigation
Most of the evaluated scrapers can navigate flexible through websites. The navigation can be recorede easily. Some tools provide a record by clicking. The record is started and stopped by clicking a button. Between these times the navigation is recorded. It is possible to fill out forms and log in fields as well as checkboxes and other navigation objects. Only Web-Harvest has to be configured specially by scripts to fulfil these tasks.

Easy to use

Most tools are quite easy to use. It is advisable to have computer science knowledge. A broad documentation and flash videos are available to make the introduction as simple as possible. Some tools require special knowledge. For Web-Harvest general programming knowledge is required. WebQL demands SQL and regular expression knowledge and RoboSuite requires a broad knowledge in regular expressions.

Good support for proxy servers

A proxy server routes the whole web crawling traffic to one configured computer. This makes the crawling process more anonymous because the web server obtains the information of the proxy server. The web scraper information is hidden. All tools except VisualWebTask support the handling of proxy servers.

5.2 General Weaknesses

This subsection describes the general weaknesses based on the evaluated web scraper tools.

No quality assurance

The scraper is as good as it is configured. It is possible to give the scraper a bit of logic with the help of regular expressions. For example a regular expression can assure if a result contains numbers. But it can not be assured that the information is correct. So the quality of the scraped information can vary. There has to be an external quality assurance to grant a high quality level.

No intelligent adjustment

The scraper can not adjust if websites change. The structure of a page is needed to locate the desired pattern. The configured rules can not be applied if a website's structure is changed. Such an attempt is ended by an error message and no information is scraped from the changed website.

Problems with extracting information from flash pages

Flash is used for animations and interactive web pages. Sport betting companies use flash for their live bets. Some scrapers can navigate to flash pages but at the moment there is no possibility to crawl information from these pages. This is due to the reason that the Adobe Flash Standard was released in March 2008.

No prevention algorithm against crawler traps
A crawler trap causes that a web crawler makes an infinite number of requests leading to a crash. None of the evaluated tools have a prevention algorithm to detect a crawler trap.

No support for Captchas
A Captcha is a Turing test to distinguish between computer programs and human users. Websites do this by displaying a generated string in a picture. The user enters the string. Currently no evaluated tool is able to handle these Captchas. Concerning this matter I have to mention that I personally did not see any Captchas on sport betting pages.

No support of robot exclusion standard
This advisory protocol rules the access of crawlers to websites. The website operator can restrict the access by this protocol. Not any evaluated tool supports the robot exclusion standard. This means the desire of the website operators are not respected but there is no real disadvantage because by default all websites can be crawled.

5.3 Comparison of Tools

The evaluation was done by an evaluation catalogue. This catalogue was created due to the functional specification which represents the needs of the domain specific company. As described above, the evaluated tools are compared with each other to find out, which tool supports which category of features best. A more detailed rating can be found in the appendix.
There are two diagrams. Figure 5.1 shows the evaluation results of the best four tools. Figure 5.2 shows the other four tools. The average of each tool and category represents the evaluation results.
The tables 5.1 and 5.2 below show the average rating of each category per tool. The rating is as follows: (0) No support by the tool, (1) Supported by a workaround, (2) Tool support with restriction, (3) Good support with room for improvements and (4) Full support by the tool.

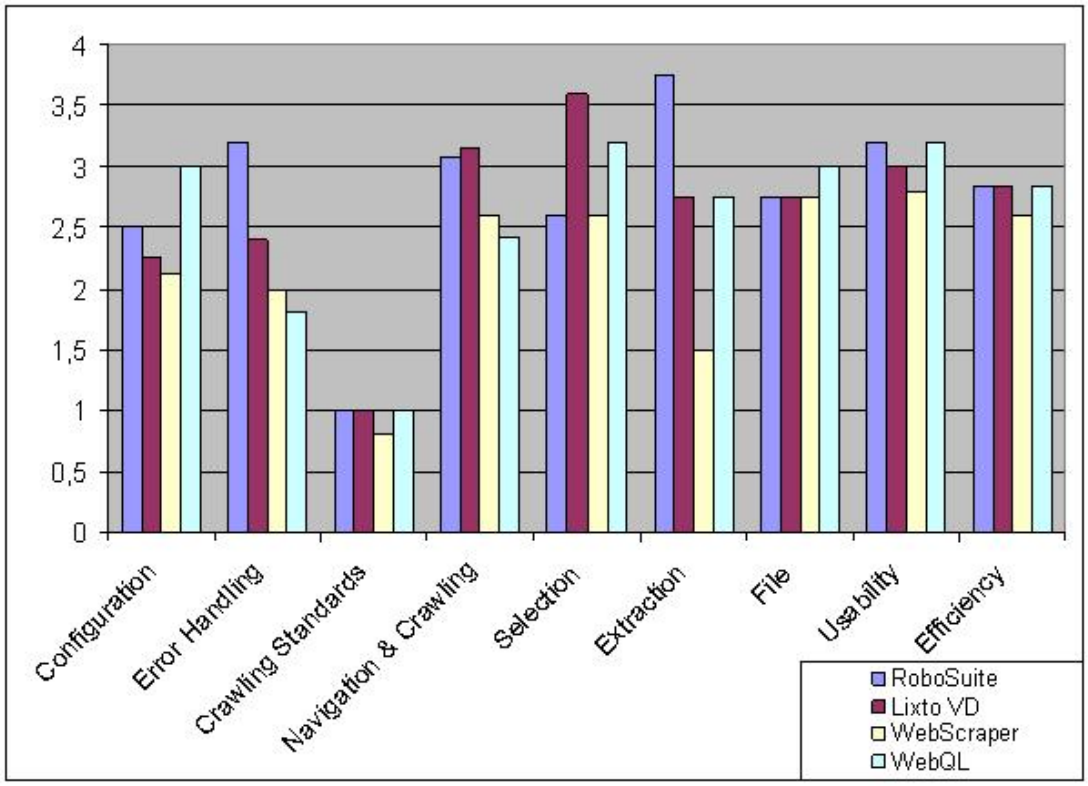


Figure 5.1: Comparison of the four best evaluated tools

Category	RoboSuite	Lixto VD	WebScraper	WebQL
Configuration	2,50	2,25	2,13	3,00
Error Handling	3,20	2,40	2,00	1,80
Crawling Stds.	1,00	1,00	0,80	1,00
Navigation	3,08	3,16	2,60	2,41
Selection	2,60	3,60	2,60	3,20
Extraction	3,75	2,75	1,50	2,75
File	2,75	2,75	2,75	3,00
Usability	3,20	3,00	2,80	3,20
Efficiency	2,83	2,83	2,60	2,83

Table 5.1: Average rating of the four best tools

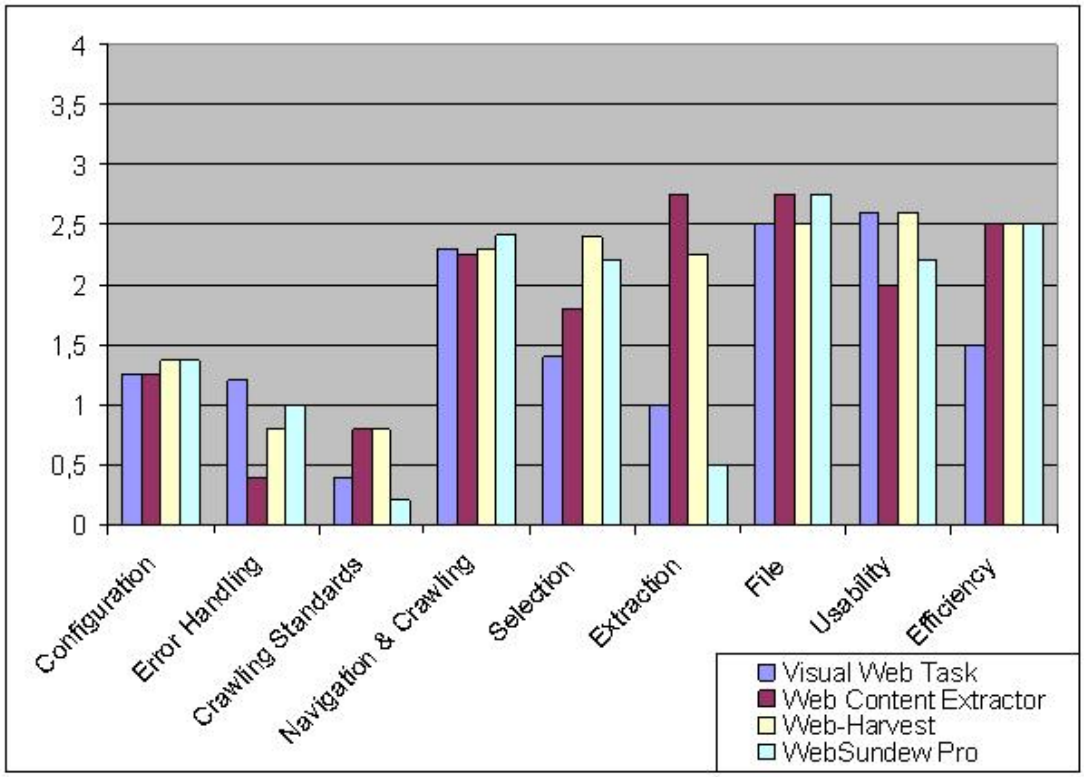


Figure 5.2: Comparison of the four weakest evaluated tools

Category	Visual Web Task	Web Content Extractor	Web-Harvest	WebSundew Pro
Configuration	1,25	1,25	1,38	1,38
Error Handling	1,25	1,25	1,38	1,38
Crawling Stds.	0,40	0,80	0,80	0,20
Navigation	2,30	2,25	2,30	2,42
Selection	1,40	1,80	2,40	2,20
Extraction	1,00	2,75	2,25	0,50
File	2,50	2,75	2,50	2,75
Usability	2,60	2,00	2,60	2,20
Efficiency	1,50	2,50	2,50	2,50

Table 5.2: Average rating of the four weakest tools

The ratings of the feature in every category are summarized and the average for the category is calculated. The averages of all categories are summarized to the total rating for one tool. The table 5.3 shows that the first three tools got far more points than all the other tools. This is not a big surprise, because these three tools are used by big companies. Some examples: RoboSuite is used by Bank of America, Intel and Deutsche Post. Lixto VD is use by Continental, hotel.de and SAP. WebQL is used by Microsoft, United Airlines and priceline.com.

Tool	Rating
RoboSuite	24,91
Lixto Visual Developer	23,74
WebQL	23,19
WebScraper	19,78
Web-Harvest	17,53
Web Content Extractor	16,50
WebSundew Pro	15,14
Visual Web Task	14,15

Table 5.3: Total score

The following sections provide a comparison of the evaluated tools concerning the above defined feature categories. This is a summary of the whole evaluation. The meanings of the used words, methods and functions can be found in the evaluation chapters and the appendix.

Configuration

WebQL has the highest rating in this category with 3,0 points. The configuration is done by the programming language WebQL. The code can be reused and tasks can be automated because templates can contain programming code. This makes the configuration for similar websites faster.

WebSundew, Lixto, WebScraperPlus and RoboSuite are configured by selecting the content from websites by clicks. This results in a quite easy configuration. The scraping template cannot be exported because they are embedded in the configuration files. RoboSuite is the only tool which can export the data models. RoboSuite got the second place with 2,5 points. Some reason for this rating is the time conversion function, the good automation of tasks and the possibility of multitask processes. All other tools do not support these functions.

Webharvest and WebQL are configured via programming languages. Webharvest is configured by XML files. This tool has an average rating because there is no way to export the data model or the scraping template, no standard configuration is provided as well as automatical executions of tasks.

WebContent Extractor and Visual WebTask provide wizards for the configuration. The configuration with a wizard makes the tool inflexible. They do not provide any standard configurations or possibilities to convert the given times. The data model cannot be exported and reused. These two tools have the worst rating in the category configuration with 1,25 points.

Error Handling

The best solution provides RoboSuite, where users can administrate notifications and error messages in an own application. This application allows a great handling by a protocol function, breakpoints, e-mail notification and so on. It makes it easy to find errors and problems as well as to solve them. RoboSuite has the highest rating with 3,2 points.
Lixto, WebScraperPlus and WebQL provide log files and visual debugging modes. They are rated on place 2 and 3. WebSundew provides a log file for the error handling. WebContentExtractor and Visual Webtask indicate errors by pop up windows. Only Webharvest has no log files and no visual debugging.

Crawling Standards

All tools do not support the robot exclusion standard, the extended robot standard and the exclusion by meta tags, because they specify more as a scraping tools than a crawler.
RoboSuite, Lixto, WebQL and WebContentExtractor can handle sitemaps and the file names can be choosen arbitrarily. Hence, these tools have a very high rating. All other tools support either the free name declaration or the sitmaps. So they have an average rating.
The last tool is Visual WebTask because no above mentioned feature is supported.

Navigation and Crawling

All tools have no problems to fill out forms and do log in. They are able to handle secure connections and next page patterns.
None of the evaluated tools provides a solution for Capcthas and no one can handle HTTP standard codes.
The highest rating got Lixto with 3,16 points. Lixto supports all other evaluated features. The second place goes to RoboSuite which has a badly implemented parameterized crawling. The other tools got average ratings. In summary the evaluated tools supply a good or excellent navigation and crawling result. The worst rating is 2,25 which is quite reasonable.

Selection

All scrapers can handle websites with tables. The highest rating got again Lixto with 3,6 points. It was possible to handle the majority selection cases of the evaluation catalogue. WebQL is as flexible as the winner tool of this category but the user needs to know the query language. RoboSuite, Webscraper and Webharvest got an average rating. The worst rating got WebSundew,

Web Content Extractor and Visual Web Task because they use a wizard and are inflexible. The most tools can handle normal HTML and XHTML. Flash is not supported by any tool. The selection methods are quite good implemented.

Extraction

The highest rating got RoboSuite with 3,75 points. That is the highest rating of all categories. Only RoboSuite allows a validation of the scraped content. Lixto and WebQL are rated to the second place with 2,75 points because they are not able to validate information. All other tools have problems with different formattings and the transformation in a proper standard and have a rating between 1,0 and 2,25. The weakest rating got WebSundew with just 0,5 points.

File

This category has a lot of high ratings. The highest rating got WebQL with 3 points. All the other tools have ratings between 2,5 and 2,75.
Standard file formats and unique names are supported by all scrapers. Only VisualWebTask and Webharvest have some bad ratings for these features. Six scrapers support the connection to databases. Just WebSundew and Webharvest do not provide any connectivity to databases.

Usability

All tools provide a good usability. The winners again are RoboSuite and WebQL with 3,2 points.
All scrapers got a high rating for an easy installation and for the help functions and the documentation. WebSundew, WebHarvest and WebContentExtractor have a bad or no implemented of the error handling. WebQL, RoboSuite and Webharvest are the only scrapers with good facilities for the expandability and adaption.
The general usability of the different tools is quite high. Only Webharvest got a very bad rating because the operation has to be done by a programming language only. Also WebQL got a bad rating because it also requires some coding parts. The worst rating got WebContentExtractor with 2 points, because the error handling is not implemented.

The following figure 5.3 summarizes some representive features of the evaluation. These features were significant for the final result of the evaluation and the represent the needs of the company. Error handling is required because the company needs to know if data from a site is not gathered successfully. The information gathering process has to be done in an automatic way and is represented by task adminstration. It is sufficient that data is extracted in an efficient way and a transformation into a unique standard is possible. Databases or files like XML are needed to store scraped content. In the graphic below more stars stand for better support of the feature. For the background knowledge this means: 3 stars - no knowledge and 1 star - computer science knowledge required.

		Features						
		Error Handling	Task Administration	Database Support	File Formats	Extraction Conditions	Data Transformation	Background Knowledge
Tools	RoboSuite	☆☆☆	☆☆☆	☆☆☆	☆☆☆	☆☆☆	☆☆☆	☆
	Lixto Visual Developer	☆☆	☆☆☆	☆☆☆	☆☆☆	☆☆☆	☆☆☆	☆
	WebQL Studio	☆☆	☆☆☆	☆☆☆	☆☆	☆☆☆	☆☆☆	☆
	Web Content Extractor	☆	☆	☆☆	☆☆☆	☆☆☆	☆☆	☆☆
	Web Scraper Plus+	☆☆	☆☆	☆☆☆	☆	☆	☆	☆☆☆
	WebSundewPro	☆	☆☆☆	☆	☆☆☆	☆	☆	☆☆☆
	Visual Web Task	☆	☆	☆☆☆	☆☆	☆	☆	☆☆☆
	Web-Harvest	☆	☆	☆	☆☆	☆☆	☆	☆

Figure 5.3: Rating overview of representative features

RoboSuite, Visual Developer and WebQL scored well because they require computer science knowledge which makes them flexibly applicable. This is shown in the figure above by the column background knowledge.

6 Conclusion and Further Work

The results of my evaluation are described in chapter 4. This chapter contains the conclusion and my recommendation for further work based on the results.

Summarized, three tools can be announced as the winners because they fit best to the criteria in the benchmark catalogue. These tools are RoboSuite, WebQL and Lixto. The evaluation was done by a benchmark catalogue due to the needs of the domain specific company.

The winner tools differ compared to the loser tools in the following points:

Error Handling

The error handling is solved in many different ways. The winning tools have a quite good error handling. Compared to the other evaluated tools, they do not have any error handling or it is weak implement.

Configuration

The loser tools operate without any previous knowledge of the user. They are less flexible because mostly everything is done by wizards. That allows no configuration for difficult jobs. The three winner tools need previous knowledge like SQL, XPATH or regular expressions. The usage of programming languages makes it easier to configure complicated settings.

Navigation and Crawling

The winner tools support more possibilities to navigate through websites.

Each of the three winner tools has its own personal advantage:

- The Lixto developers are in good contact with the Vienna University of Technology and the company is located in Vienna. That means required functions could be a part of a new Lixto version.
- RoboSuite has the best error handling implementation.
- WebQL hast he fastest crawling process.

As we have already seen, scrapers are good for gathering huge amount of data but they are not able to process data. The scrapers delivers excellent results as long as the structure of the website does not change. Therefore it is necessary to have a good quality assurance for this technology.
I think it is necessary to do further investigations about the costs by the usage of a scrapers. It should be compared if a data input team costs more than the usage of a scraper.
The following important question which influences the costs have to be answered first:

How often do big sporting websites change their structure?
This question effects the configuration and maintenance effort. My personal experience is that sport websites with a lot of leagues do not change so often than smaller sport sites.

How many sports and leagues should be supported?
The cost raises the more sports and leagues the scraper supports because the configuration effort gets higher. It is necessary to crawl results from minimum two trustful websites. This guaranties a certain level of drop out time and a good quality due to automated comparisons between the results from different websites. As minimum one special trained person is needed to monitor and adapt the scraper in regular time periods.

Is there a data processing and quality assurance system which could be used to reduce the developing costs?
The software is needed to match datasets and processes them for example team names. It is necessary to investigate if a software solution is already provided. If such software already exists at bwin the software could be adapted to the new requirements. Furthermore external investigations should be done because if a fitting software can be acquired. The last possibility is to develop an own data processing and gathering system.

I made the following formula to calculate the costs:
First calculate the average configuration effort of one league (CF) and multiply it with the number of the leagues which have to be configured (QL). Furthermore this result has to be doubled because it is necessary to have a second alternative source for each league. The next step is to add the cost for adaption of the configuration if websites change their structure (CW). Additionally the operation costs for the scraper (AC) and the data processing system (AD) have to be included.
The scraper project costs are divided by the costs for the data input team (DC) to set them in relation.

ROI = DC / (CF*QL*2 + CW + AC + AD)
CF = Average Configuration Effort per League
QL = Quantity of Leagues
CW = Configuration Effort from Structure Changes
AC = Acquisition Costs of the Crawler
AD = Acquisition Costs of the Data Processing and QA System
DC = Costs of the Data Input Team

If the cost calculation delivers a positive result I would recommend the development of a prototype. The prototype could work as described in figure 6.1.

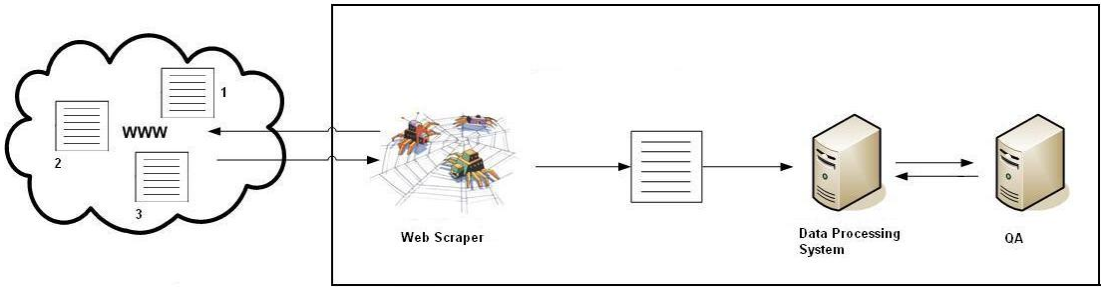


Figure 6.1: Recommendation for web scraper

As described above scrapers are "stupid". They just can gather information from websites and they are not able to assure the quality of the content. For example wrong or no data is gathered if the structure of the website changed or the website contains wrong information. To minimize this risk I would recommend configuring more websites for the same league. This causes more configuration effort and creates a dependency to different websites. The results of different

websites could be compared with a quality assurance component of the data processing system. The data processing system could start the scraper at the time a sport event should be finished and could gather the results.
Conceivably could be a combination of data feed providers and an automated data gathering of a scraper. This variation minimizes the dependency to websites which are gathered by the scraper and to the feed providers. An interesting question is if feed providers receive their information from websites. This means the dependency to websites will not be fixed by a combination of these two technologies.
A scraper can be used in different ways for bwin. As this project shows one possibility is to use the scraper as data provider for fixtures, results and other data. In the future if it is possible to crawl content from flash pages an application scenario like live bet analysis of competitors is possible. Specially collect data of the coverage and availability of the offered bets.
It is a consideration to start this project for technological and strategically reasons to use the web as a data provider.

7 Appendix

7.1 Functional Specification

This chapter describes the behavior of the system with the aid of use case diagrams and actors. Furthermore the use case diagram is presented as an activity diagram.

7.1.1 Actors

There are two actors the web crawler administrator and the data process system. The first actor is responsible for the right administration and configuration of the crawler. The web crawler administrator is a special skilled employee for the special needs which are required to work with and to configure the web crawler. The data process system however handles with results and fixtures. This system could match and compare the gathered information from different pages.

Web Crawler Administrator

- Configures web crawler

Data Processing System

- Gets fixtures about sport events
- Gets results about sport events

7.1.2 Use Case Diagram

The following use case diagram shows the system of the web crawler. The web crawler administrator configures the crawler and monitors the created notifications. A detailed description of the single use cases can be found in chapter 7.2.

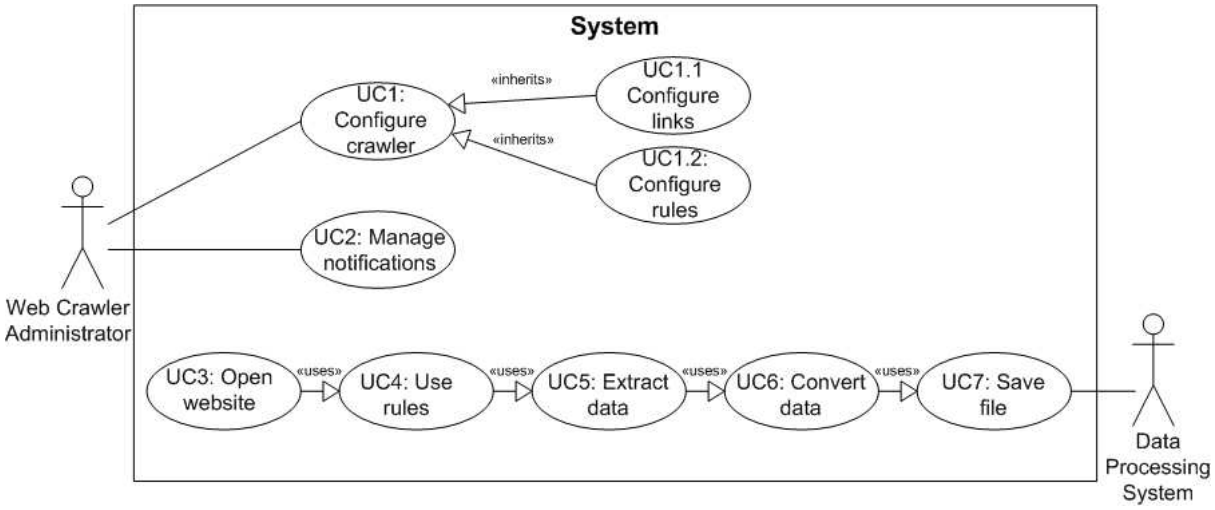


Figure 7.1: Use case diagram: data crawler

A data crawler behaves the following way:

- The crawler opens the corresponding page by navigating through the website. This could be done by recording and play back a navigation sequence or a link assembled by parameters.
- With defined rules the crawler selects the desired pattern. These rules are named scraping or extraction template.
- The selected information is extracted. Although the time can be converted to UTC (Coordinated Universal Time).
- Subsequent the extracted information is transformed in a structured format like XML (=Extensible Markup Language).
- Last the file is stored at a certain directory where the data process system can access it.

7.1.3 Functional Requirements

The use case diagram from chapter 7.1 is shown below as the activity diagram. During the information gathering process there can appear different errors. For each error a notification has to be created.

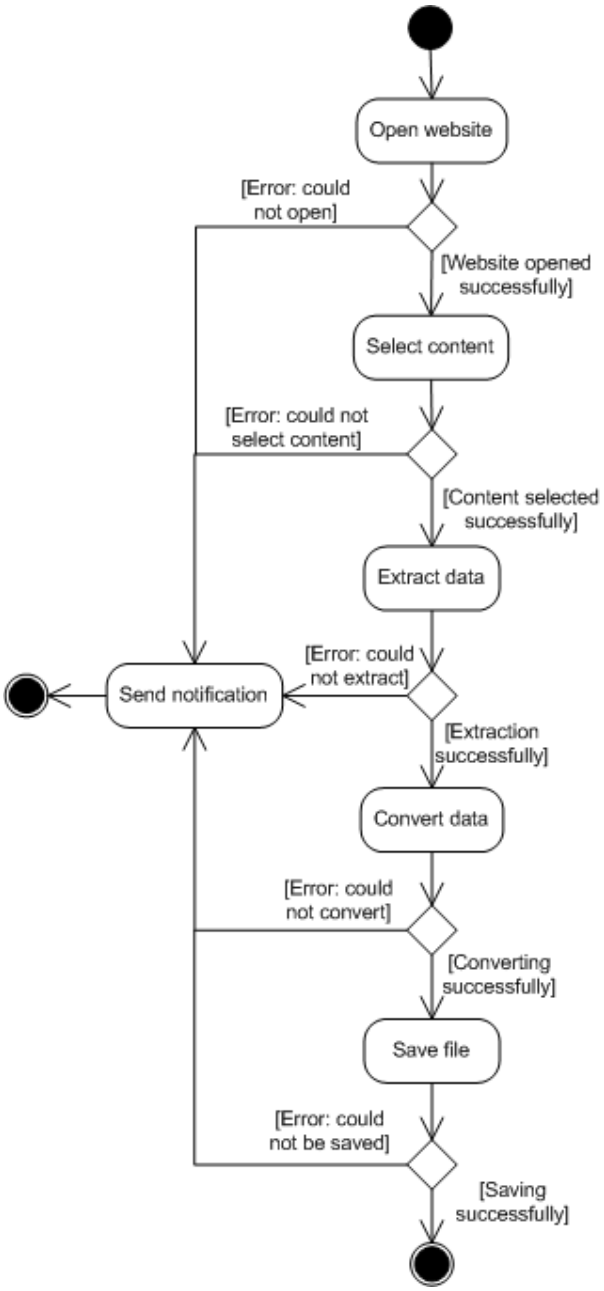


Figure 7.2: Activity diagram: data crawler

7.2 Functional Requirements

The functional requirements describe how the future sport book data crawler for bwin should act. It should provide proper and reliable information about fixtures and results of a sport event. The following use cases are extracted from the use case diagram show in chapter 7.1. Each use case is depicted detailed as an activity diagram and explained explicit by a use case description.

7.2.1 Configure Crawler

Actors	Web Crawler Administrator
Priority	high
Preconditions	Web crawler is installed.
Postconditions	The crawler is configured successfully. It is possible to start the crawling process.
Trigger	Web Crawler Administrator wants to configure the web crawler.
Short description	Web Crawler Administrator configures the Web crawler.
Main success scenario	[**A1**] Web Crawler Administrator starts the crawler and is in the configuration menu. [**A2**] Web Crawler Administrator configures a new data model. [**A3**] Web Crawler Administrator configure links -> UC1.1 [**A4**] Web Crawler Administrator configure rules -> UC1.2
Extensions	[**R2a**] Web Crawler Administrator wants to change a data model. [**R2a1**] Web Crawler Administrator deletes or edits an existing data model. [**R2a2**] Web Crawler Administrator wants to configure links. ->[A3]
Requirements	[**R1.1**] Web Crawler Administrator is authorized to configure the crawler. [**R1.2**] System provides standard configurations which can be adapted easily. [**R2.1**] If Web Crawler Administrator does the described action the system shall store the data model. For example for a fixture of a soccer event the following sport information: Sport, Region, League, Team A, Team B, Event Time, Event Date [**R2.2**] The system should be able to handle different data models. When a fixture or a result was defined the data model looks different. It also varies because of the different types of sports. [**R2.3**] It should be possible to reuse the data model on the same site for another league.

Table 7.1: Configure crawler

7.2.2 Configure Links

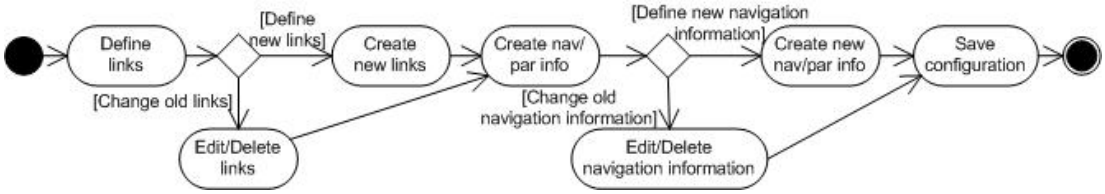


Figure 7.3: Activity diagram: configure links

Actors	Web Crawler Administrator
Priority	high
Preconditions	Web Crawler Administrator is authorized to configure web crawler.
Postconditions	The links to websites are saved.
Trigger	Web Crawler Administrator wants to configure the links to websites for the web crawler.
Short description	Web Crawler Administrator defines links for crawling.
Main success scenario	[**A1**] Web Crawler Administrator defines new links of websites for leagues. [**A2**] Web Crawler Administrator defines for each website the recordable navigation or parameter information (this helps to navigate to the correct site of a league) [**A3**] Web Crawler Administrator saves configurations.
Extensions	[**A1a**] Web Crawler Administrator wants to change a link. [**A1a1**] Web Crawler Administrator deletes or edits a link for a league. [**A1a2**] Web Crawler Administrator wants to define navigation information. ->[A2] [**A2a**] Web Crawler Administrator wants to change parameter or navigation information. [**A2a1**] Web Crawler Administrator deletes or edits parameter or navigation information of a site. [**A2a2**] Web Crawler Administrator wants save the configurations. ->[A3]
Requirements	[**R1.1**] If Web Crawler Administrator do the described action the system shall store: Link of preferred website for league, Navigation or/and parameter information [**R1.2**] Websites for a league can differ if fixtures or results are needed. [**R1.3**] The links to the websites should use parameter (national/international, name of country, name of league, ..) to get automatically to the correct league.

Table 7.2: Configure links

7.2.3 Configure Rules

Actors	Web Crawler Administrator
Priority	high
Preconditions	Web Crawler Administrator is authorized to configure information which is required for a successful crawling. The Links were configured successfully.
Postconditions	A scraping template with rules how to gather the information of a site is saved.
Trigger	Web Crawler Administrator wants to configure which data should be scraped.
Short description	The information of a sport event can differ from league to league and if fixtures or results are needed. As a result a scraping template for a site is created.
Main success scenario	[**A1**] Web Crawler Administrator defines where the information is located on a website. [**A2**] Web Crawler Administrator defines the time zone of the website. [**A3**] Web Crawler Administrator enters rules of a website (scraping template). [**A4**] Web Crawler Administrator adds a scraping template to a website. [**A5**] Web Crawler Administrator saves the configuration.
Extensions	[**A1a**] Web Crawler Administrator changes where the information is located on a website. [**A2a**] Web Crawler Administrator changes the time zone of the website. [**A3a**] Web Crawler Administrator changes the entered rules of a website (scraping template) [**A4a**] Web Crawler Administrator removes a scraping template from a website.
Requirements	[**R1.1**] If Web Crawler Administrator does the described action the system shall store the position of the information. [**R2.1**] The crawler should calculate the time to UTC with defined rules. [**R3.1**] If User do the described action the system shall store rules for each website [**R3.1**] The rules should contain where the required information is placed on a site. [**R3.2**] It should be possible to reuse the rules on the same site for another league.

Table 7.3: Configure rules

7.2.4 Manage Notifications

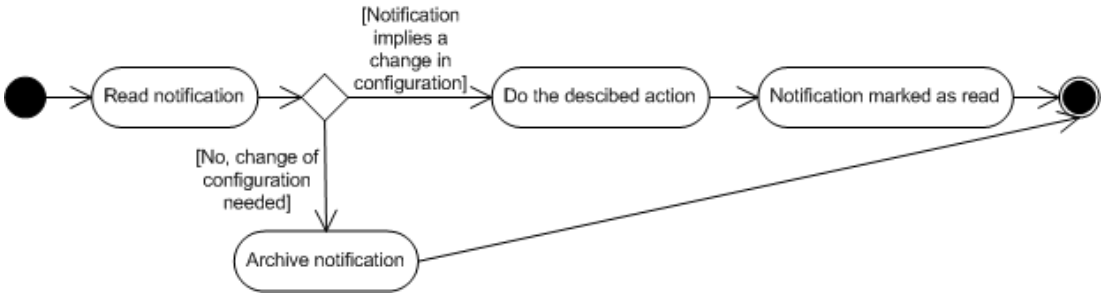


Figure 7.4: Activity diagram: manage notifications

Actors	Web Crawler Administrator
Priority	medium
Preconditions	Web crawler is installed.
Postconditions	The notifications are read
Trigger	Web Crawler Administrator wants to read notifications.
Short description	Web Crawler Administrator can manage appeared notifications.
Main success scenario	[**A1**] Web Crawler Administrator reads a notification. [**A2**] Web Crawler Administrator does the described actions. [**A3**] Notification is marked as read.
Extensions	[**A2a**] Web Crawler Administrator wants to archive notifications. [**A2a1**] Web Crawler Administrator archives a notification.
Requirements	[**R1.1**] There can appear different notifications: Opening error, No alternative site available, Navigation error, Incomplete information, Results still not available, Matching error, Data/Time transformation error, Converting error, Saving error, Other errors or advises [**R1.2**] The notification should be provided in a graphical user interface. [**R1.3**] Notification should be clear and easy to understand.

Table 7.4: Manage notifications

7.2.5 Open Website

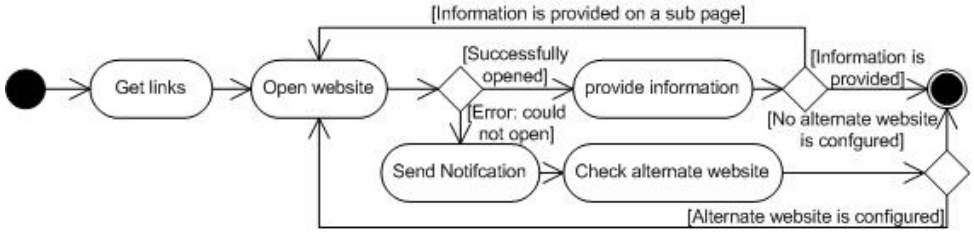


Figure 7.5: Activity diagram: open website

Actors	System
Priority	High
Preconditions	Web crawler is configured for crawling (UC1: Configure crawler, UC1.1: Configure links & UC1.2: Configure rules).
Postconditions	Website for crawling is opened.
Trigger	In periodic time intervals or at a defined time.
Short description	Web crawler opens a website.
Main success scenario	[**A1**] Web crawler gets link of configuration. [**A2**] Web crawler tries to open website. [**A3**] Website is opened. [**A4**] The required information is provided on this page.
Extensions	[**A2a**] The website can not be opened. [**A2a1**] Send notification about opening error. [**A2a2**] Check if an alternative site is configured. [**A2a3**] Alternative site is available. [**A2a4**] Web crawler opens alternative website. ->[A2] [**A2a3a**] No alternative site is available. [**A2a3a1**] Send notification that no alternative site is available. [**A4a**] The required information is not provided on this page [**A4a1**] Crawler navigates to sub page. This is done by a recorded play back. -> [A2]
Requirements	[**R1.1**] The links to the websites should be able to use parameters, so the correct league is got automatically, and navigation to sub pages is not needed. [**R2.1**] It should be possible to define waiting periods during opening websites to assure not an overload of the site. [**R2.2**] The crawler should behalf on the robot exclusion standard. [**R2.3**] The crawler should be able to understand HTTP status codes. [**R4a1.1**] The navigation should be recordable that the crawler learns to navigate on different sites.

Table 7.5: Open website

7.2.6 Use Rules

Actors	System
Priority	High
Preconditions	Website for crawling is opened (UC3)
Postconditions	Selected the required information by applying rules.
Trigger	The site is opened successfully.
Short description	The Crawler selects the information which should be extracted if a result is needed. If a fixture is needed the use case is the same only without the check if the fixture ended.
Main success scenario	**[A1]** Web crawler tries to use defined rules on the opened website. **[A2]** The rules apply to the site. **[A3]** Crawler checks that the required information (e.g.: the information needed for a result) defined in the data model for this event is provided on this site. **[A4]** Crawler checks if an event ended (by checking the syntax of the scraped website, e.g.: the word "finished") **[A5]** The event ended. **[A6]** The required information of an event is selected.
Extensions	**[A2a]** Rules do not apply to the site **[A2a1]** Send notification about applying error. **[A2a2]** Crawler stops using his rules and goes to the next league. -> [A1] **[A3a]** Some information of an event are missing **[A3a1]** Do not select any information from this league **[A3a2]** Send notification about incomplete information. **[A3a3]** Crawler stops using his rules and goes to the next league. ->[A1] **[A4a]** Results are needed but the information about the results is not available on the site. **[A4a1]** Do not select any information from this league **[A4a2]** Send notification about that a result is still not available **[A4a3]** Crawler stops using his rules and goes to the next league. ->[A1]
Requirements	**[R1.1]** It should be possible to select information even when the information is unstructured. **[R3.1]** Just complete information of sport events should be delivered (This could be done with the help of regular expressions). **[R4.1]** The used rule process should be able to check if an event ended (Some pages display if an event ended, for example change of color of the result, or with the word "Finished", ...). **[R6.1]** There should be a highlighting in the GUI of the selected information.

Table 7.6: Use rules

7.2.7 Extract Data

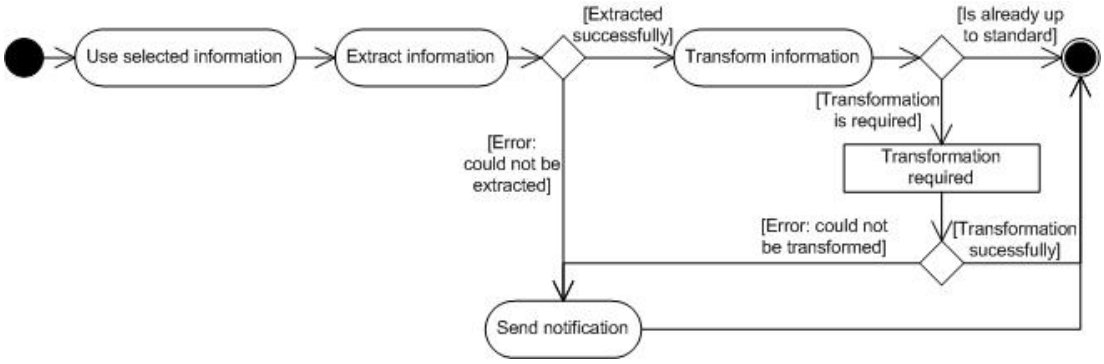


Figure 7.6: Activity diagram: extract data

Actors	System
Priority	High
Preconditions	The information is selected for extracting (UC4).
Postconditions	The crawler extracts the required information of a site.
Trigger	The rules fit to the site.
Short description	Data about a sport event is extracted from a website.
Main success scenario	**[A1]** Web crawler uses selected information of sport events (UC4). **[A2]** Crawler tries to extract the information from the website. **[A3]** Crawler extracted the information about sport events from the website out. **[A4]** The extracted data is already in a proper standard.
Extensions	**[A2a]** Data could not be extracted. **[A2a1]** Send notification about extracting error. **[A4a]** Some information like date or time need to be transformed to a proper standard and/or to UTC. (e.g.: Friday 7 Dec 2007 -> 07/12/2007) **[A4a1]** The data was successfully transformed. **[A4a1a]** The information can not be matched with the time or date standard. **[A4a1b]** Send notification about data/time transformation error.
Requirements	**[R2.1]** Only needed information about sport event is extracted. **[R2.2]** Crawler should be able to extract information from different technologies (HTML, Flash, ...) **[R2.1]** There need to be a validation if an extracted string is like the expected (e.g.: a result needs to start with a digit followed by a separation character like : and ends with another digit)

Table 7.7: Extract data

7.2.8 Convert Data

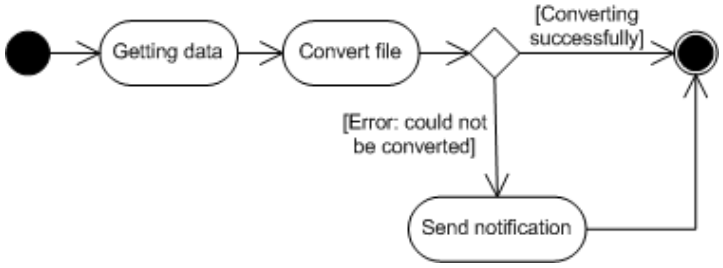


Figure 7.7: Activity diagram: convert data

Actors	System
Priority	High
Preconditions	Data about sport event has been parsed
Postconditions	File in a proper standard is created
Trigger	The parsing process was successfully
Short description	The parsed data is used to create a file in a proper standard.
Main success scenario	[**A1**] The converting process gets the data (UC5). [**A2**] System tries to convert the data. [**A3**] The data is converted to a file in a proper standard.
Extensions	[**A3a**] File could not be converted. [**A3a1**] Send notification about converting error.
Requirements	[**R2.1**] The created file will differ depending if it is a fixtures or result. [**R2.2**] Each created file should have a unique name (e.g.: Sport + Region + League + Date + Time + Site name). [**R2.3**] The file should be created in a defined standard (like XML). [**R2.4**] The User should be possible to customize the structure of the files.

Table 7.8: Convert data

7.2.9 Save File

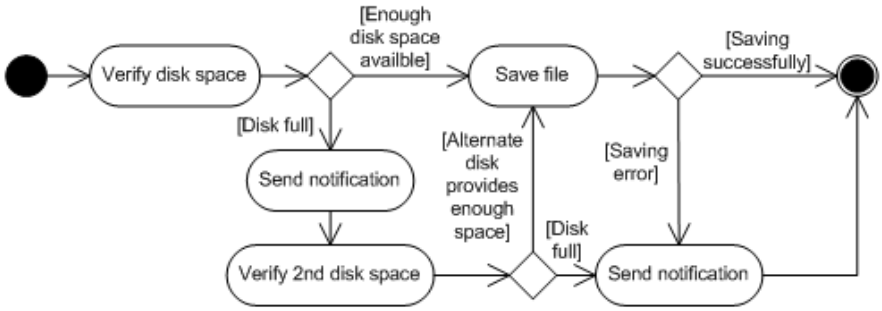


Figure 7.8: Activity diagram: save file

Actors	System
Priority	Normal
Preconditions	A file in a proper standard with information about sport events exist
Postconditions	The file is saved to a specific folder
Trigger	New file is created
Short description	File with information about sport events is saved to a specific folder where the Data processing System can access it.
Main success scenario	**[A1]** The data is converted to a proper standard (UC6) **[A2]** Verify the free disk space at the specific folder. **[A3]** The file is tried to be saved in a specific folder. **[A4]** File is saved successfully.
Extensions	**[A2a]** There is not enough free disk space available. **[A2a1]** Send notification that the disk is full **[A2a2]** Try to save the file local to another disk. **[A2a3]** File saved successfully in local folder. **[A2a3a]** A saving error appears **[A2a3b]** Send notification about saving error. **[A3a]** The file could not be saved **[A3a1]** Send notification about file saving error **[A3a2]** Try to save the file to another disk. **[A3a3]** File saved successfully. **[A3a3a]** A saving error appears **[A3a3b]** Send notification about saving error.
Requirements	**[R2.1]** The user is able to define the folders where data should be stored in a user interface.

Table 7.9: Save file

7.2.10 Remaining Activity Diagram

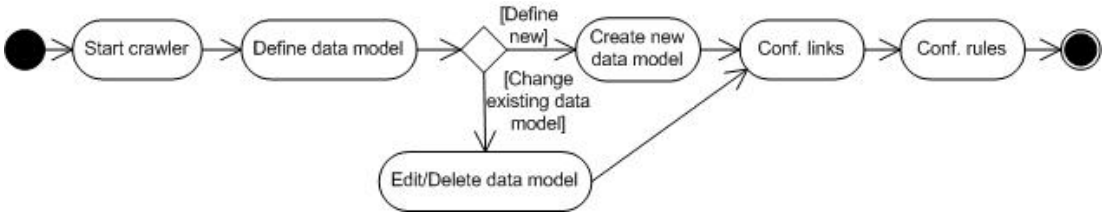


Figure 7.9: Activity diagram: configure crawler

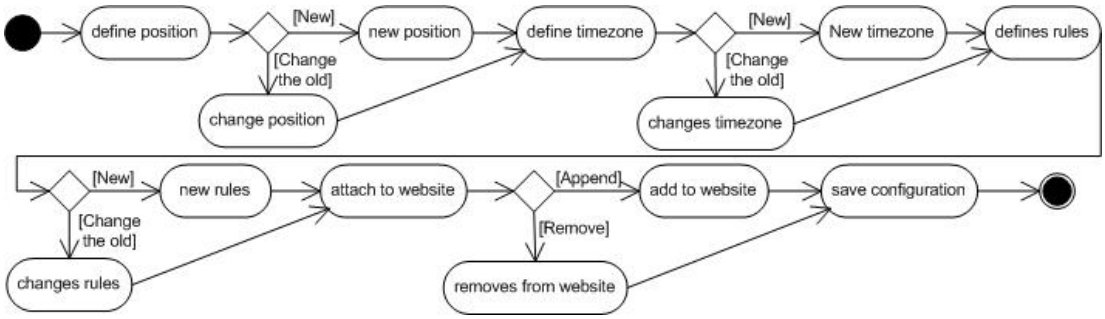


Figure 7.10: Activity diagram: configure rules

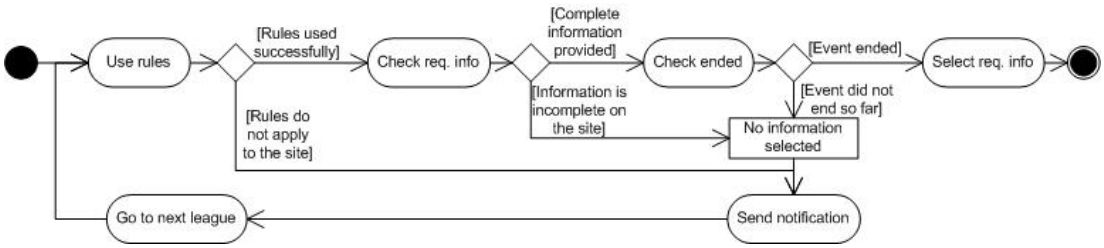


Figure 7.11: Activity diagram: use rules

7.3 Non Functional Specification

This chapter contains the features and functions of web crawler, which are necessary to extract information successfully from websites. These criteria's are worked out by literature investigations and the try out of different crawler. The criteria are chosen to allow an efficient, fast and user friendly handling. The following results can go above or below the named criterion limits. This leads to more or less points in the evaluation. The non functional requirements have been arranged to must and nice to have non functional requirements.

7.3.1 Must Non Functional Requirements

[**RN 01**] The system should be crash free working.
[**RN 02**] It should be effectiveness. The resulting performance should be compared in relation to the effort of configuration.
[**RN 03**] The administration of the crawler should be done over a graphical user interface.
[**RN 04**] System requirements should be as low as possible.
[**RN 05**] The crawler should be extensible. It should be possible to add features, plug-ins and other forms of customizations.
[**RN 06**] The crawler should be able crawl more than one site at the same time.
[**RN 07**] The availability has to be up to 99,6%.
[**RN 08**] Every interaction has to be logged in detail to be able to relate to errors.
[**RN 09**] It should be able to backup the configuration files of the crawler.

7.3.2 Nice To Have Non Functional Requirements

[**RN 10**] The installing process should be done in max 10 minutes.
[**RN 11**] The documentation should be easy to understand.
[**RN 12**] The crawler should run on various platforms.
[**RN 13**] The price should be as low as possible (Open source, License for one workstation,...).
[**RN 14**] Maintainability: How intense is it to install updates, etc
[**RN 15**] The crawler should be flexible and adaptable to changes on a site.
[**RN 16**] An online help should be available.
[**RN 17**] Usability criteria regarding Jacob Nielsen should be fulfilled.

7.4 Detailed Ratings

Each feature was evaluated and the rating is as follows:

Rating	Description
0	No support by the tool
1	Support via workaround
2	Tool support with restriction
3	Good support with room for improvements
4	Full support by the tool

Table 7.10: Ratings

7.4.1 Lixto VD

The following tables show the rating (abbreviated by an "R") and description of how the Lixto Visual Developer supports this feature (feature number abbreviated by an "FNr").

FNr	Feature	R	Description
1	Creation/Modifying of data model	4	Data models can be created easily. They are displayed in a tree view.
2	Importing/Exporting/ Reuse of data model	4	Data models can be imported and exported as XML schemas.
3	Creation/Modifying of scraping template	2	A scraping template can not be created. Anyway it is possible to configure the crawler in a general way.
4	Importing/Exporting/ Reuse of scraping template	1	The complete crawler settings are saved in a *.lixvw file. Parts like a scraping template can not be exported.
5	Creation/Modifying of URLs	4	URLs can be created, built or extended by parameters.
6	Time zone	0	This feature is not supported.
7	Standard configurations	0	This feature is not supported.
8	Task administration	3	A crawler can be executed in an automated way.
9	Intelligent adjustment	2	Few changes are handled smoothly by the crawler, but broad changes lead to an automated generated error message.
10	Protocol function	2	It is not possible to choose the log depth. Each message is logged in a report.

11	Notifications	3	Each defined action result in a report line. For example open the website www.soccer.com generates: "Executing URL: www.soccer.com".
12	Notification administration	2	Notifications are administrated in a report view. This overview is not structured very well.
13	Visual debugging	3	It is possible to debug the configured crawler. Breakpoints can be set at any time. It is possible to start and stop at a defined sequence of the crawling and extracting process.
14	User-Agent	4	The name of the crawler is arbitrary. It can be defined with a browser action.
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	1	Sitemaps can be found due to the tags in the HTML code. There is no explicit function for this feature.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.
19	Recordable crawler navigation	4	The navigation is recorded if the user clicks on a recording button. Every action is logged and can be replayed.
20	Log in	4	The crawler can do the log in on websites.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	4	URLs can be parameterized with an URL action (e.g.: www.soccer.com/ + "region" + "/" + "league").
23	Waiting periods	4	Waiting periods can be defined with a waiting action.
24	Multitasking	4	With the server application pages can be crawled simultaneous.
25	Http standard codes	0	The user can not influence the HTTP request behavior.
26	Filling forms	4	Forms can be filled, options, checkboxes and radio buttons selected. The filled form can be submitted. This is easily done by simply recording the actions.
27	Pop up windows	3	With the switch action it is possible to switch between the different pop up windows.
28	Cookies	3	Cookies can be removed with a clean action.
29	Secure connections	4	The crawler handles secure connections smoothly.

30	Next Page	4	A next page pattern can be defined and is iterated without any problems.
31	Selection methods	4	The selecting of information works with XPATH, regular expressions or scripts.
32	Tables	4	Information in tables can be selected without any problems. Rows and columns can be excluded.
33	Technologies	2	Flash is not supported. HTML and XHTML are supported.
34	Capturing meta data	4	Meta information can be extracted from websites smoothly.
35	Header Information	4	The header information can be extracted by a header action.
36	Extraction condition	4	There can be defined four different extraction conditions: Contains, Not Contains, Around, Not Around.
37	Efficient extraction	3	Just the defined information is extracted.
38	Transformation of formatting differences	3	A substitute function can be created. A String or a regular expression can be defined which should be replaced. For example change the words "Juventus Turin" into the defined substitution "Juventus". Such a filter can be used for the whole page.
39	Validation	0	This feature is not supported.
40	Unique name	3	The filename can be chosen arbitrary. It is not possible to use parameters for the filename.
41	Standard file formats	3	The output format is XML.
42	Customize standard formats	2	The output format can be changed by the aid of the data model. Nested XML structures can not be created.
43	Database connectivity	3	The following databases are supported: Oracle, MySQL, PostgreSql and other which works with JDBC.
44	Usability	3	The interface of the Lixto Visual Developer is outlined well. The icons are easily to understand.
45	Adaptability and expendability	2	The appearance of the tool can be adapted and changed to the user's wishes. The tool itself could not be extended by the user. External tools can be added.
46	Help functions and documentation	3	Each function is not explained in the online help. The documentation is well structured and very clearly arranged.

47	Error handling	3	For an occurred error an error message is created which is quite clear.
48	Installation	4	The installation is done by a setup wizard.
49	System assumption	4	The application runs smoothly on computer with standard configuration.
50	Platform independent	3	A version for Windows and Linux is available. Other platforms are not supported.
51	Support	3	To learn the handling there are various tutorials available. Some of these tutorials do not work due to structure changes.
52	Effectiveness	3	It is easy to learn the handling and to usage the tool.
53	Proxy Server	4	It is possible to configure a proxy server.
54	Crawler Traps	0	There is no prevention algorithm against crawler traps.

Table 7.11: Lixto detailed rating

7.4.2 RoboSuite

The following tables show the rating (abbreviated by an "R") and description of how the RoboSuite supports this feature (feature number abbreviated by an "FNr").

FNr	Feature	R	Description
1	Creation/Modifying of data model	4	A data model can be created easily.
2	Importing/Exporting/ Reuse of data model	4	The tool provides possibilities to save and open data models.
3	Creation/Modifying of scraping template	2	A scraping template can not be created. It is possible to configure the crawler in a general way.
4	Importing/Exporting/ Reuse of scraping template	1	The general settings can be saved and reopened. This saves configuration steps.
5	Creation/Modifying of URLs	2	URLs can be created. Another possibility is to crawl URLs which contain specified names. It is not possible to build together a link with parameters.

6	Time zone	3	A default time zone and a result time zone can be configured. The time is converted automatically in the desired format. The definition of the time zone is quite hidden in the tool.
7	Standard configurations	0	This feature is not supported.
8	Task administration	4	Crawler can be grouped for example by the name "England Premiere League". This group contains for example ten crawler for the different pages of the "England Premiere League". These can be executed in an automated way.
9	Intelligent adjustment	2	Few changes can be handled smoothly by the crawler, but broad changes lead to an automated error.
10	Protocol function	3	A logfile can be created to note information like crawled URL, time, etc. The configuration of a log file could be easier.
11	Notifications	4	A notification via e-mail can be added as action to each step of the crawler.
12	Notification administration	4	Status and error messages generated by robots are viewed in the RoboManager. The identification of crawler that are broken because of significant web interface changes is quite easy. There are different symbols for: successful execution, warnings, errors, no messages generated and not run so far.
13	Visual debugging	3	By the aid of the RoboDebugger a configured crawler can be debugged. Breakpoints can be set at any time. The execution of a loop through table rows could be faster.
14	User-Agent	4	The name of the crawler is arbitrary.
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	1	Sitemaps can be found due to the tags in the HTML code. There is no explicit function for this feature.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.

19	Recordable crawler navigation	4	The navigation is automatically recorded if mouse clicks, drop down menus or other navigation steps are done. Java Scripts can be executed by a "javascript execution" action.
20	Log in	4	The form and HTTP log in operates without any problems.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	2	The tool provides the function to find sites on the basis of given criteria. For example: all pages from a specified domain, specified strings or specified suffix. An URL can not be built with parameters. To find the required information the site menu and direct links can be used.
23	Waiting periods	4	Waiting periods can be defined at any time of the crawling process. There are two ways to define time periods. The first method is to define a time period which have to be awaited. The second one is to define a "wait for time event" which means the crawler has to wait a certain time period after a specific clock time.
24	Multitasking	4	Different tasks can be executed at the same time by run multiple instances of RoboServer. These instances are administrated by the ControlCenter.
25	Http standard codes	0	This feature is not supported.
26	Filling forms	4	Forms can be filled, options, checkboxes and radio buttons selected. The filled form can be submitted.
27	Pop up windows	3	Each frame and pop up window is displayed in the overview. RoboMaker chooses the main window as the standard frame. The other windows are sub windows. Not used windows can be closed by a close action. A function which closes all not needed windows at once would be helpful.
28	Cookies	4	Cookies are administrated in a cookie view. Cookies are added automatically if a page was loaded. RoboMaker provides three actions for cookies: it is possible to extract cookie values, to delete and to create own cookies.
29	Secure connections	4	The crawler handles secure connections smoothly.
30	Next Page	4	The crawler handles a next page pattern smoothly.

31	Selection methods	3	To select text, tags can be defined. Between this tags the text is selected and extracted. The selection takes place by clicking on the required information. It is not possible to select nodes with XPATH.
32	Tables	4	Information in table can be selected without problems. Certain rows can be excluded due to specified criteria.
33	Technologies	2	Flash is not supported. HTML and XHTML are supported.
34	Capturing meta data	2	Meta-Information can be selected for the extraction. There is no explicit function for this feature.
35	Header Information	2	The selection of header information like title of the page works smoothly. There is no explicit function for this feature.
36	Extraction condition	3	With the action tag or the test attribute it is possible to check if a specific condition is true (e.g.: is the word "FINISHED" contained in a result). There is no possibility to save a extraction condition and use it in other crawling tasks.
37	Efficient extraction	4	Just the defined information is extracted.
38	Transformation of formatting differences	4	RoboMaker provides a data converter. It is easy to configure it because just the input and output pattern has to be defined.
39	Validation	4	A validation takes place by a pattern. If result is fits with the pattern the information is extracted otherwise a notification can be created.
40	Unique name	3	For the file with the extracted information can be chosen parameterisable and flexible filenames. The name of the file is defined with RoboRunner as an argument (fileName=resultsGermanBundesliga.xml)
41	Standard file formats	3	The following output formats can be configured XML (default), CSV, SCSV.
42	Customize standard formats	2	The output format can be changed by the aid of the data model. Nested XML structures can not be created.
43	Database connectivity	3	The RoboSuite supports the following databases: Oracle, IBM DB2, Microsoft SQL Server, Sybase PointBase Server, Solid, MySQL.

44	Usability	3	The interface of the different tools is consistent and the icons are understandable. Sometimes the text is cropped.
45	Adaptability and expendability	2	There is no support for enlarging the crawler by plugins. The menu entries can not be adapted to the users need. For extending the crawler there is an API available.
46	Help functions and documentation	4	A wide range of help functions is available. For each tool a manual is available and tutorials are provided.
47	Error handling	3	For an occurred error an error message is created which is quite clear.
48	Installation	4	The installation is simple. Additionally, an installation guide is provided. The installation and configuration is done quite fast.
49	System assumption	4	The application runs smoothly on computer with standard configuration.
50	Platform independent	3	A version for Windows and Linux is available. Other platforms are not supported.
51	Support	3	A forum is available to post questions. There is also a possibility to track problems. These tracks are delivered directly to the RoboSuite team. There is no telephone support.
52	Effectiveness	3	The learning time is short so it takes just few times to configure the first crawling tasks. Configurations could be done faster by the usage of scraping templates.
53	Proxy Server	4	There is a possibility to configure a proxy server.
54	Crawler Traps	0	There is no prevention algorithm against crawler traps.

Table 7.12: RoboSuite detailed rating

7.4.3 VisualWebTask

The following tables show the rating (abbreviated by an "R") and description of how the VisualWebTask supports this feature (feature number abbreviated by an "FNr").

FNr	Feature	R	Description
1	Creation/Modifying of data model	3	The data model is created during the creation of the scraping template. So the data model is embedded in the scraping template.
2	Importing/Exporting/ Reuse of data model	0	This feature is not supported.
3	Creation/Modifying of scraping template	2	A scraping template is created for one site and can be assigned to other sites.
4	Importing/Exporting/ Reuse of scraping template	0	This feature is not supported.
5	Creation/Modifying of URLs	3	An URL belongs to one scraping template.
6	Time zone	0	This feature is not supported.
7	Standard configurations	0	This feature is not supported.
8	Task administration	2	The crawler can be started over the command line. An external tool can be used to start the crawler in certain time periods.
9	Intelligent adjustment	1	Few changes can lead to problems because content from the page can not be scraped. The crawling task has to be adapted to the changes.
10	Protocol function	2	The crawling process to the required website can be shown in a log view.
11	Notifications	3	Notifications are displayed in a window during the crawling process.
12	Notification administration	0	This feature is not supported.
13	Visual debugging	0	This feature is not supported.
14	User-Agent	2	The user-agent field contains the following information: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; iOpus-Web-Automation; InfoPath.1; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648). It is not possible to change this information.

15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	0	This feature is not supported.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.
19	Recordable crawler navigation	4	The navigation can be recorded by simply click the button "start recording".
20	Log in	4	A log in at a website works smoothly.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	0	This feature is not supported.
23	Waiting periods	0	This feature is not supported.
24	Multitasking	4	Different tasks can be executed at the same time.
25	Http standard codes	0	This feature is not supported.
26	Filling forms	3	Forms can be filled out and submitted.
27	Pop up windows	3	Pop up windows are loaded but the crawler automatically chooses the main page for the crawling process.
28	Cookies	2	Cookies are administrated automatically. There is no way to set, change or delete cookies.
29	Secure connections	4	The crawler can handle HTTPS without any problems.
30	Next Page	4	A next page pattern can be defined and is iterated without any problems.
31	Selection methods	2	The information is selected by HTML nodes like ¡td¿ or ¡tr¿. Only information in the body part of a HTML site can be selected.
32	Tables	3	Information in table can be selected. Empty cells of a table are scraped empty.
33	Technologies	2	Flash is not supported. HTML and XHTML are supported.
34	Capturing meta data	0	This feature is not supported.
35	Header Information	0	This feature is not supported.
36	Extraction condition	0	This feature is not supported.
37	Efficient extraction	4	Just the defined information is extracted.
38	Transformation of formatting differences	0	This feature is not supported.
39	Validation	0	This feature is not supported.

40	Unique name	2	The file name can be chosen arbitrary. It can not be build with parameters.
41	Standard file formats	3	The following file formats are supported: TXT and XML.
42	Customize standard formats	2	The customization of the file is limited.
43	Database connectivity	3	The extracted data can be stored in an ODBC database.
44	Usability	3	The interface of the tool is quite clear. Just the most necessary functions are included.
45	Adaptability and expendability	0	This feature is not supported.
46	Help functions and documentation	3	Each function is not explained in the online help. The documentation is well structured and very clearly arranged.
47	Error handling	3	For an occurred error an error message is created which is quite clear.
48	Installation	4	The installation is done by a setup wizard.
49	System assumption	4	The application needs a normal standard configured.
50	Platform independent	2	Windows
51	Support	0	There is no support program available.
52	Effectiveness	3	The learning time into the tool is quite short. But the tool provides just the some few main functions.
53	Proxy Server	0	This feature is not supported.
54	Crawler Traps	0	This feature is not supported.

Table 7.13: VisualWebTask detailed rating

7.4.4 Web Content Extractor

The following tables show the rating (abbreviated by an "R") and description of how the Web Content Extractor supports this feature (feature number abbreviated by an "FNr").

FNr	Feature	R	Description
1	Creation/Modifying of data model	3	A data model is created during the selection process. This is done by clicking on content and assigned to a name of the data model.
2	Importing/Exporting/ Reuse of data model	0	This feature is not supported.

3	Creation/Modifying of scraping template	2	A scraping template is created for one site and can be assigned to other sites.
4	Importing/Exporting/ Reuse of scraping template	0	This feature is not supported.
5	Creation/Modifying of URLs	3	Different links can be added to the scraping template.
6	Time zone	0	This feature is not supported.
7	Standard configurations	0	This feature is not supported.
8	Task administration	2	It is possible to perform the crawler from the command line. Tasks can not be performed automated in certain time periods. An external tool has to be used.
9	Intelligent adjustment	1	Few changes can lead to problems. This could be a forgotten closed tag.
10	Protocol function	0	This feature is not supported.
11	Notifications	1	Notifications are just displayed in the configuration window.
12	Notification administration	0	This feature is not supported.
13	Visual debugging	0	This feature is not supported.
14	User-Agent	4	Default: Mozilla/4.0 The name of the crawler is arbitrary.
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	0	This feature is not supported.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.
20	Log in	4	A log in at a website works smoothly.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	3	The tool provides the function to find sites on the basis of given variables.
23	Waiting periods	3	A delay between requests for web pages from the same server can be added.
24	Multitasking	0	This feature is not supported.
25	Http standard codes	0	This feature is not supported.
26	Filling forms	3	Forms can be filled out.
27	Pop up windows	3	Pop up windows are ignored by the crawler.

28	Cookies	3	Cookies are automatically set and can be deleted in the internet explorer.
29	Secure connections	4	The crawler can handle HTTPS without any problems.
30	Next Page	4	The next page which should be crawled can be defined with setting the Follow links if linked text contains to the correct value.
31	Selection methods	3	Information can be selected by defining an example on the website. The HTML path and other attributes are created automatically.
32	Tables	4	Information in table can be selected without problems. Certain rows can be excluded with a result filter.
33	Technologies	2	Flash is not supported. HTML and XHTML are supported.
34	Capturing meta data	0	This feature is not supported.
35	Header Information	0	This feature is not supported.
36	Extraction condition	4	An extraction condition can be set with a result filter.
37	Efficient extraction	4	Just the defined information is extracted.
38	Transformation of formatting differences	3	For modifying extracted data from a web page a script can be added. The script language either can be Microsoft VBScript and Jscript.
39	Validation	0	This feature is not supported.
40	Unique name	4	The filename can be chosen arbitrary. The date and timestamp can be added automatically to the filename.
41	Standard file formats	3	Microsoft Excel, TXT, HTML, XML.
42	Customize standard formats	2	The formatting of the XML file can be settled by a wizard.
43	Database connectivity	2	MySql, Access
44	Usability	3	The usability of the tool is quite good. The tool is very clearly designed because just the most necessary functions are included.
45	Adaptability and expendability	0	Menu entries can not be adapted to the users need. There is no possibility to enlarge the tool.
46	Help functions and documentation	3	There are four flash tutorials available. A user guide for the tool is available.
47	Error handling	0	This feature is not supported.
48	Installation	4	The installation is done by a setup wizard.

49	System assumption	4	The application runs smoothly on computer with standard configuration.
50	Platform independent	2	Only a version for Windows is available. Other platforms are not supported.
51	Support	2	A support via email is possible.
52	Effectiveness	3	The learning time is short. The tool is small and clearly designed.
53	Proxy Server	4	The crawler inherits the proxy settings of the internet explorer.
54	Crawler Traps	0	There is no prevention algorithm against crawler traps.

Table 7.14: WebContentExtractor detailed rating

7.4.5 Webharvest

The following tables show the rating (abbreviated by an "R") and description of how Webharvest supports this feature (feature number abbreviated by an "FNr").

1	Creation/Modifying of data model	2	A data model is defined in the XML configuration file.
2	Importing/Exporting/ Reuse of data model	2	Data models can not be imported and exported. They can be copied from another configuration file.
3	Creation/Modifying of scraping template	2	The scraping process is defined in the XML configuration file.
4	Importing/Exporting/ Reuse of scraping template	2	A scraping template can not be imported and exported. They can be copied from another configuration file.
5	Creation/Modifying of URLs	3	URLs can be added in the configuration file..
6	Time zone	0	This feature is not supported.
7	Standard configurations	0	This feature is not supported.
8	Task administration	0	This feature is not supported.
9	Intelligent adjustment	2	Few changes are handled smoothly by the crawler, but broad changes lead to an automated generated error message.
10	Protocol function	2	It is not possible to choose the log depth. Each message is logged in a report view.
11	Notifications	0	This feature is not supported.

12	Notification administration	0	This feature is not supported.
13	Visual debugging	0	This feature is not supported.
14	User-Agent	4	The name of the crawler can be chosen arbitrary.
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	0	This feature is not supported.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.
19	Recordable crawler navigation	0	This feature is not supported.
20	Log in	4	It is possible to log in on websites.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	4	URLs can be parameterized (e.g.: "www.soccer.com/" + region + "/" + league).
23	Waiting periods	0	This feature is not supported.
24	Multitasking	1	A tool which starts the crawling process needs to be implemented.
25	Http standard codes	0	This feature is not supported.
26	Filling forms	4	Form fields can be filled out.
27	Pop up windows	4	Pop up windows do not disturb because the absolute path can be opened.
28	Cookies	3	It is possible to define a cookie policy.
29	Secure connections	4	The crawler handles secure connections smoothly.
30	Next Page	4	A next page pattern can be defined.
31	Selection methods	3	The information is selected with XPATH.
32	Tables	3	It is possible to select table content. So just parts are extracted.
33	Technologies	2	Flash is not supported. HTML and XHTML are supported.
34	Capturing meta data	2	This information can be selected.
35	Header Information	2	This information can be selected.
36	Extraction condition	3	It is possible to extract if a certain condition is fulfilled.
37	Efficient extraction	3	The defined information can be extracted as well substrings can.

38	Transformation of formatting differences	1	Formatting differences can be unified but this is bonded to a high writing effort.
39	Validation	2	Expressions can be assureed if they contain certain patterns by the aid of regular expressions.
40	Unique name	4	The name of the file is user defined.
41	Standard file formats	4	The structure can be chosen freely.
42	Customize standard formats	2	Presently scraped information can only be written in XML files.
43	Database connectivity	0	This feature is not supported.
44	Usability	1	The user has to have basic experience in programming, XPATH and XML.
45	Adaptability and expendability	4	The source code is free available and can be expanded any time.
46	Help functions and documentation	3	User manual, examples and java doc is available.
47	Error handling	1	An empty file is created if an error occurs.
48	Installation	4	It is not necessary to install the software. The extraction of the program is the complete installation process.
49	System assumption	4	The tool runs really smoothly on standard configured computers.
50	Platform independent	4	The tool is written in Java. So it is platform independent.
51	Support	2	Support is given in a forum or via e-mail to the developer.
52	Effectiveness	1	It is necessary to learn, how to use the tool. Especially the syntax of the configuration files has to be learned as well as XPATH experience is advisable.
53	Proxy Server	4	It is possible to setup a proxy server.
54	Crawler Traps	0	There is no prevention algorithm against crawler traps.

Table 7.15: Webharvest detailed rating

7.4.6 WebQL

The following tables show the rating (abbreviated by an "R") and description of how WebQL supports this feature (feature number abbreviated by an "FNr").

1	Creation/Modifying of data model	3	A data model is created in the select clause of a query.
2	Importing/Exporting/ Reuse of data model	2	A data model itself can not be exported and reused.
3	Creation/Modifying of scraping template	2	A scraping template does not really exist. In WebQL a query is a kind of a scraping template.
4	Importing/Exporting/ Reuse of scraping template	3	The written query code can be reused by coping it into a new query.
5	Creation/Modifying of URLs	3	URLs of web pages with the same side structure can be added to a query.
6	Time zone	4	The times on a website can be transformed to a desired time zone by the aid of the $reformat_datetime$ function.
7	Standard configurations	3	A template to reduce the effort of configuration can be saved.
8	Task administration	4	A scheduler is available to automate the execution of the queries.
9	Intelligent adjustment	0	This feature is not supported.
10	Protocol function	3	Information about the crawling process is written into log files.
11	Notifications	4	After executing a query at the WebQL server a diagnostic file is generated.
12	Notification administration	0	This feature is not supported.
13	Visual debugging	2	A visual debugging mode is not available. Queries can be debugged with the out_log function.
14	User-Agent	4	The user-agent name can be chosen arbitrary.
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	1	Sitemaps can be found due to the tags in the HTML code. There is no explicit function for this feature.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.

19	Recordable crawler navigation	0	This feature is not supported.
20	Log in	4	The form and HTTP log in operates without any problems.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	0	This feature is not supported.
23	Waiting periods	3	Waiting periods between executing queries can only be defined in the WebQL Server.
24	Multitasking	2	The WebQL studio can perform one query at the same time. A WebQL Server is needed to perform more queries at the same time.
25	Http standard codes	0	This feature is not supported.
26	Filling forms	4	Forms can be filled, options, checkboxes and radio buttons selected. The filled form can be submitted.
27	Pop up windows	4	WebQL chooses the main frame window.
28	Cookies	4	WebQL handles cookies automatically.
29	Secure connections	4	The crawler can handle HTTPS without any problems.
30	Next Page	4	The crawler handles the next page pattern smoothly.
31	Selection methods	4	There are three different methods to select content on a web page: regular expressions, tables and with HTML code before and after the selected text passage.
32	Tables	4	The selecting of information in tables works smoothly.
33	Technologies	2	Flash is not supported. HTML and XHTML are supported.
34	Capturing meta data	2	Meta-Information can be selected for the extraction. There is no explicit function for this feature.
35	Header Information	4	Header information can be easily retrieved.
36	Extraction condition	4	The condition of the extraction can be defined in the where clause.
37	Efficient extraction	4	It is possible to extract substrings of strings.
38	Transformation of formatting differences	3	Formatting differences can be transformed into one proper standard.
39	Validation	0	This feature is not supported.
40	Unique name	4	The filename can be chosen arbitrary.
41	Standard file formats	3	Microsoft Excel, TXT, HTML, XML, CSV

42	Customize standard formats	2	The output data can be changed by the select clause of a query. Nested XML structures can not be created.
43	Database connectivity	3	WebQL allows writing the extracted information into a database through ODBC.
44	Usability	2	The usability of the tool is quite good, but a user should bring knowledge in regular expressions and SQL.
45	Adaptability and expendability	3	QL2 server provides an API which allows the developer to embed the data extraction and transformation capabilities in custom applications.
46	Help functions and documentation	4	The tool provides a great documentation with a lot of simple and understandable examples.
47	Error handling	3	For an occurred error an error message is created which is quite clear.
48	Installation	4	The installation is quite simple and done by an installation wizard with few user interactions.
49	System assumption	4	The application needs a normal standard configured computer.
50	Platform independent	3	The program is for Windows and Linux available. Other platforms are not supported.
51	Support	3	QL2 provides a basic and advanced training for the users.
52	Effectiveness	3	The crawler is easy to configure if the user has experiences with SQL and regular expressions.
53	Proxy Server	4	The crawler supports proxy servers.
54	Crawler Traps	0	This feature is not supported.

Table 7.16: WebQL detailed rating

7.4.7 WebScraperPlus

The following tables show the rating (abbreviated by an "R") and description of how the WebScraperPlus supports this feature (feature number abbreviated by an "FNr").

FNr	Feature	R	Description
1	Creation/Modifying of data model	3	A data model is created during the scraping template. The data model is called dataset.
2	Importing/Exporting/ Reuse of data model	1	A separate exportation or importation is not possible because the data model is linked with the scraping template.
3	Creation/Modifying of scraping template	4	The creation of a scraping template is simple. The desired pattern can be selected by simply clicking on it.
4	Importing/Exporting/ Reuse of scraping template	2	A scraping template can not be imported or exported. Various links can be added to the scraping template to automate the process.
5	Creation/Modifying of URLs	3	Various URLs can be added to scraping templates.
6	Time zone	0	This feature is not supported.
7	Standard configurations	2	There are no standard configurations, but there is a wizard to configure only the main steps.
8	Task administration	2	Task can be configured with an automation wizard. They are displayed in an overview. They can be launched via a console.
9	Intelligent adjustment	2	Few changes of a website can be handled smoothly by the crawler, but broad changes lead to an automated error.
10	Protocol function	2	The tool creates a log file which protocols each process and event of the tool.
11	Notifications	1	Additional notifications can not be created.
12	Notification administration	1	Notifications can not be administrated.
13	Visual debugging	4	The visual debugging is a kind of step by step replay of the extraction. That is an easy way to find errors in the configuration.
14	User-Agent	4	WebScraperPlus can emulate any user agent by setting the user agent field. By default: Mozilla/4.0
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	0	This feature is not supported.

17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.
19	Recordable crawler navigation	3	A direct link to the desired pattern is used. So navigation does not have to be recorded.
20	Log in	3	The crawler is able to log in on various websites by the aid of a form submit task.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	4	Links can be crawled parameterized. Thereby limitations can be taken like hostname, filename, file type and others.
23	Waiting periods	0	This feature is not supported.
24	Multitasking	3	Task can be performed in an automated way.
25	Http standard codes	0	The user can not influence the HTTP request behavior.
26	Filling forms	4	A form submits task has to be created to fill out a form. This task contains the requested input information.
27	Pop up windows	4	The crawler handles pop up windows smoothly.
28	Cookies	3	Cookies can be stored and viewed by a Web Form Explorer.
29	Secure connections	4	Secure connections work smoothly.
30	Next Page	4	Next pages can be crawled automatically if a next page pattern is defined.
31	Selection methods	3	The selection of content works smoothly. A data record can be selected. After that the tool automatically finds further data records. The selection is done by the aid of text strings. The longer the string is the better the tool works.
32	Tables	4	Due to the structured architecture of tables in HTML the selection works smoothly.
33	Technologies	3	HTML and XHTML sites can be crawled. Flash sites could not be scraped, because the selection of the information works by the surroundings of HTML code.
34	Capturing meta data	0	This feature is not supported.
35	Header Information	3	The header information can be selected.
36	Extraction condition	0	This feature is not supported.
37	Efficient extraction	4	Just the defined information is extracted.
38	Transformation	0	Some formatting differences can be transformed to a standard.

39	Validation	0	This feature is not supported.
40	Unique name	4	The name of the file can be chosen arbitrary.
41	Standard file formats	2	Excel is supported as output format. No other formats are supported.
42	Customize standard formats	1	The structure of the file can not be affected because Excel is the only output format beside the normal database formats.
43	Database connectivity	4	The following databases are supported: SQL Server, MySQL, delimited text files, Access and others with ODBC drivers.
44	Usability	4	The usability is quite good. A wizard is available for the configuration.
45	Adaptability and expendability	0	This feature is not supported.
46	Help functions and documentation	4	Online help, video tutorials, faqs, forum, examples, additionally training packages can be bought.
47	Error handling	3	A notification is created if an error occurs but it is not possible to create additional notifications for the user.
48	Installation	3	Before install the crawler the Microsoft .NET Framework 2.0 has to be installed. The installation itself is done by a wizard. Synchronous Microsoft SQL Server 2005 is installed.
49	System assumption	3	The application runs smoothly on computer with standard configuration. If many tasks are executed at the same time, a fast computer is advisable.
50	Platform independent	2	Windows 2000, XP, Vista
51	Support	4	A support package can be purchased. Guaranteed Same Business Day Response, otherwise a forum for immediate questions is available.
52	Effectiveness	3	The learning time is short, because there is a lot of material. After the video tutorials it should be no problem to handle the basics of the tool.
53	Proxy Server	4	The proxy settings are inherited from the internet explorer settings.
54	Crawler Traps	0	This feature is not supported.

Table 7.17: WebScraperPlus detail rating

7.4.8 WebSundew

The following tables show the rating (abbreviated by an "R") and description of how the WebSundew supports this feature (feature number abbreviated by an "FNr").

FNr	Feature	R	Description
1	Creation/Modifying of data model	3	A data model is created during the selection process. This is done by clicking on content and assigned to a name of the data model.
2	Importing/Exporting/ Reuse of data model	0	This feature is not supported.
3	Creation/Modifying of scraping template	2	A scraping template is only created for one site.
4	Importing/Exporting/ Reuse of scraping template	0	This feature is not supported.
5	Creation/Modifying of URLs	3	Different links can be added to the Web macro.
6	Time zone	0	This feature is not supported.
7	Standard configurations	0	This feature is not supported.
8	Task administration	3	Tasks can be executed in certain time intervals or on defined days by the scheduler.
9	Intelligent adjustment	1	Few changes can lead to problems. This could be a forgotten closed tag.
10	Protocol function	0	This feature is not supported.
11	Notifications	2	A notification contains time, date, source and text of the message.
12	Notification administration	2	Notifications are viewed in a message log view. These are dived in navigation and extraction log.
13	Visual debugging	0	This feature is not supported.
14	User-Agent	0	There is no possibility to change or give the crawler a name.
15	Robot exclusion standard	0	This feature is not supported.
16	Robot exclusion non standard extensions	1	Sitemaps can be found due to the tags in the HTML code. There is no explicit function for this feature.
17	Robot exclusion extended standard	0	This feature is not supported.
18	Excluding with meta tags	0	This feature is not supported.

19	Recordable crawler navigation	4	The navigation can be recorded.
20	Log in	4	The form and HTTP log in operates without any problems.
21	Captcha	0	This feature is not supported.
22	Parameterized crawling	3	The tool provides the function to find sites on the basis of given variables.
23	Waiting periods	0	This feature is not supported.
24	Multitasking	2	Tasks are executed in a first in - first out process.
25	Http standard codes	0	This feature is not supported.
26	Filling forms	4	Forms can be filled, options, checkboxes and radio buttons selected. The filled form can be submitted.
27	Pop up windows	2	Pop up windows are displayed in an own frame and do not disturb.
28	Cookies	2	Cookies are administrated automatically. The user can not effect or change them.
29	Secure connections	4	Connection via HTTPS works smoothly.
30	Next Page	4	Iterating to the next page works smoothly.
31	Selection methods	2	Information can be selected by a data iterator pattern or a detail data pattern.
32	Tables	3	The selection of content line by line works fluently. Content which is spread over several rows can not be identified as context. This can lead to problems with empty cells of the data model.
33	Technologies	2	Flash pages are navigable but are not fully displayed. It is not possible to extracted content from flash sites.
34	Capturing meta data	2	For this feature there is no explicit function, but meta data can be selected.
35	Header Information	2	For this feature there is no explicit function, but header information can be selected.
36	Extraction condition	0	This feature is not supported.
37	Efficient extraction	2	Problems occur if part of Strings should be extracted.
38	Transformation of formatting differences	0	This feature is not supported.
39	Validation	0	This feature is not supported.
40	Unique name	4	The name of the file can be customized with variables and templates.

41	Standard file formats	3	These file formats are supported CSV, XML and XLS.
42	Customize standard formats	3	The file formats can be customized by templates.
43	Database connectivity	1	There should be a support for MS SQL, MySQL, Oracle and other databases if there is a corresponding JDBC driver according to the online help. But there is no way to use other output formats than CSV, XML or XLS.
44	Usability	3	The interface of the tools is quite clear.
45	Adaptability and expendability	0	Menu entries can not be adapted to the users need. There is no possibility to enlarge the tool.
46	Help functions and documentation	3	The help documentation is incomplete because some help pages are blank. There are flash tutorials available.
47	Error handling	2	Occurred errors are written into the log message.
48	Installation	3	The installation is simple and the configuration is done fast. The installation is done by a wizard.
49	System assumption	4	Operating System Requirement: Windows Vista/XP/2000, CPU: Pentium III/Athlon 500 MHz or above, RAM: 256 MB or above.
50	Platform independent	2	Windows Vista/XP/2000
51	Support	3	A forum is available to post questions. There is also a possibility to track problems.
52	Effectiveness	2	The learning time is quite short. The configuration could be faster by the use scraping templates.
53	Proxy Server	4	It is possible to configure a proxy server.There is a possibility to configure a proxy server.
54	Crawler Traps	0	There is no prevention algorithm against crawler traps.

Table 7.18: WebSundew detail rating

7.5 HTTP Status Codes

Each Status-Code is named below [FGM+99].

Informational 1xx

- 100 Continue
- 101 Switching Protocols

Successful 2xx

- 200 OK
- 201 Created
- 202 Accepted
- 203 Non-Authoritative Information
- 204 No Content
- 205 Reset Content
- 206 Partial Control

Redirection 3xx

- 300 Multiple Choices
- 301 Moved Permanently
- 302 Moved Temporarily

303 See Other

- 304 Not Modified
- 305 Use Proxy
- 306 [Unused]
- 307 Temporary Redirected

Client Error Definitions 4xx

- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 406 Not Acceptable
- 407 Proxy Authentication Required
- 408 Request Timeout
- 409 Conflict
- 410 Gone
- 411 Length Required
- 412 Precondition Failed
- 413 Request Entity Too Large
- 414 Request-URL Too Long

415 Unsupported Media Type

416 Request Range Not Satisfiable

417 Expectation Failed

Server Error Definitions 5xx

500 Internal Server Error

501 Not Implemented

502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

505 HTTP Version not supported

List of Figures

List of Tables

Bibliography

[AM07] Andrew A. Adams and Rachel McCrindle. *Pandora's Box: Social and Professional Issues of the Information Age*. Wiley and Sons, 1 edition, 2007.

[BCGMS00] Onn Brandman, Junghoo Cho, Hector Garcia-Molina, and Narayanan Shivakumar. Crawler-friendly web servers. *SIGMETRICS Perform. Eval. Rev.*, 28(2):9–14, 2000.

[Bec07] P. Becker. Gesucht und gefunden: Die funktionsweise einer suchmaschine. Technical report, Wissens- und Informationsmanagement, 2007.

[BFG07] Robert Baumgartner, Oliver Frölich, and Georg Gottlob. The lixto systems applications in business intelligence and semantic web. In *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*, pages 16–26, Berlin, Heidelberg, 2007. Springer-Verlag.

[Bra01] P. Bradley. Robots, spiders and your website. *Ariadne*, Issue 21, 2001. http://www.ariadne.ac.uk/issue27/search-engines/intro.html.

[Bre98] J. Bremer. Ein datenbankunterstütztes, hierarchisches sichtensystem für das world wide web. Master's thesis, Universität Hannover, 1998.

[CBT05] V. Poulopoulos C. Bouras and A. Thanou. Creating a polite, adaptive and selective incremental crawler. *IADIS International Conference WWW/INTERNET 2005*, Volume I:307 – 314, 2005. Lisbon, Portugal.

[Cho02] Junghoo Cho. *Crawling the web: discovery and maintenance of large-scale web data*. PhD thesis, Stanford, CA, USA, 2002. Adviser-Garcia-Molina,, Hector.

[CS07] Z. Covic and L. Szedmina. Security of web forms. *Intelligent Systems and Informatics, 2007. SISY 2007. 5th International Symposium on*, pages 197–200, Aug. 2007.

[FGM+99] R. Fieldin, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol. `http://www.ietf.org/rfc/rfc2616.txt`, 1999.

[HC03] K. Hemenway and T. Calishain. *Spidering hacks, 100 Industrial-Strength Tips & Tools*. O'Reilly Media, 2003.

[HKB08] Wolfgang Holzinger, Bernhard Kruepl, and Robert Baumgartner. Exploiting semantic web technologies to model web form interactions. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1145–1146, New York, NY, USA, 2008. ACM.

[HKKR05] Martin Hitz, Gerti Kappel, Elisabeth Kapsammer, and Werner Retschitzegger. *UML@Work, Objektorientierte Modellierung mit UML 2*. dpunkt.verlag, Heidelberg, 2005. ISBN: 3-89864-261-5.

[Kos07] M. Koster. The web robots pages. `http://www.robotstxt.org/robotstxt.html`, 2007.

[Kre06] B. Kretzel. Evaluation of requirements engineering tools and a value-based tool selection process. Master's thesis, TU Vienna, 2006.

[LB06] Anália G. Lourenço and Orlando O. Belo. Catching web crawlers in the act. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, pages 265–272, New York, NY, USA, 2006. ACM.

[LJF+07] Qiong Li, Tao Jin, Yuchen Fu, Quan Liu, and Zhiming Cui. The design and implementation of a topic-driven crawler. In *IITA '07: Proceedings of the Workshop on Intelligent Information Technology Application*, pages 153–156, Washington, DC, USA, 2007. IEEE Computer Society.

[LRS+06] G. Lietz, K. Rapke, I. Schicktanz, H. Stautmeister, and W. Henning. Optimierung der auffindbarkeit von webinhalten, 2006.

[Mac01] Leszek A. Maciaszek. *Requirements analysis and system design: developing information systems with UML.* Addison-Wesley Longman Ltd., Essex, UK, UK, 2001.

[MN01] S. Münz and W. Nefzger. *HTML 4.0 Handbuch. HTML, JavaScript, DHTML, Perl.* Franzis Verlag, 2001.

[Nai82] J. Naisbitt. *Megatrends.* Warner Books, 1982.

[Nie00] J. Nielsen. *Usability Engineering.* Academic Press Inc, 2000.

[PLL03] Eng-Huan Pek, Xue Li, and Yaozong Liu. Web wrapper validation. In *Web Technologies and Applications*, page 597. Springer Berlin / Heidelberg, 2003.

[Pop07] Ana-Maria Popescu. *Information extraction from unstructured web text.* PhD thesis, University of Washington, Seattle, WA, USA, 2007.

[RA05] A. Rungsawang and N. Angkawattanawit. Learnable topic-specific web crawler. *J. Netw. Comput. Appl.*, 28(2):97–114, 2005.

[SCG08] Yang Sun, I.G. Councill, and C.L. Giles. Botseer: An automated information system for analyzing web robots. *Web Engineering, 2008. ICWE '08. Eighth International Conference on*, pages 108–114, July 2008.

[Sch07] Michael Schrenk. *Webbots, Spiders, and Screen Scrapers: A Guide to Developing Internet Agents with PHP/CURL.* No Starch Press, March 2007.

[Tru06] R. Trujillo. Simulation tool to study focused web crawling strategies. Master's thesis, Lund University, 3 2006.

[VCR+07] Om Vikas, Nitin J. Chiluka, Purushottam K. Ray, Girraj Meena, Akhil K. Meshram, Amit Gupta, and Abhishek Sisodia. Webminer–anatomy of super peer based incremental topic-specific web crawler. In *ICN '07: Proceedings of the Sixth International Conference on Networking*, page 32, Washington, DC, USA, 2007. IEEE Computer Society.

[ZXL08] Weifeng Zhang, Baowen Xu, and Hong Lu. Web page's blocks based topical crawler. In *SOSE '08: Proceedings of the 2008 IEEE*

International Symposium on Service-Oriented System Engineering, pages 44–49, Washington, DC, USA, 2008. IEEE Computer Society.