Temporal Knowledge Acquisition and Modeling

Cyril Faucher[1], Charles Teissèdre[2,3], Jean-Yves Lafaye[1], Frédéric Bertrand[1]

[1] L3i – University of La Rochelle, France
{cyril.faucher, frederic.bertrand, jean-yves.lafaye}@univ-lr.fr
[2] MoDyCo – UMR 7114 - Paris Ouest Nanterre La Défense University – CNRS, France
[3] Mondeca – 3, cité Nollez, Paris, France
charles.teissedre@mondeca.com

Abstract. The objectives of this paper are to present, describe, and explain the foundations and the functionalities of a temporal knowledge acquisition and modeling solution workflow, which aims at acquiring temporal knowledge from texts in order to populate a constrained object model. We are using several models for temporal data, one of which is generic and employed as a pivot model between a linguistic representation and a calendar representation. The approach we propose is generic and has been tested against a real use case, in which input data is made of temporal properties defining when a given location (a theater, a restaurant, a shopping center, etc.) is open or closed. Most expressions entered are expressed in intension. Our models provide a core support to the system that linguistically analyses data entries, transforms them into extensive calendar information and allow users to control the quality of the system's interpretation.

Keywords: Temporal Knowledge Acquisition, Temporal Data Modeling, Linguistic Annotation, Model Driven Engineering.

1 Introduction

With the surge of Semantic Web applications, many fields of knowledge are now subject to semantic browsing and querying. Time is a common feature that appears in many pieces of information, whatever the domain. Hence, the need of models and standards to deal with temporal issues; either for querying, visualizing, updating or reasoning, i.e.: processing temporal data. Useful standard specifications already exist: ISO19108 [1], OWL-Time [2], iCalendar [3]. Leaving apart the heterogeneity of time reference systems which is extensively and rather satisfactorily addressed in the literature, processing temporal information remains a challenge, and open practical questions are still key issues. Temporal expressions enounced in natural language can be complex. In fact, a huge amount of knowledge about time and calendar shared by people is still out of the scope of concrete time related information systems because information is neither accessible nor exploitable.

This paper describes an integrated approach with corresponding models and software components that fulfill a part of users' needs. The comprehensive applicative context concerns a knowledge base about event's occurrences. In the simplest case, a

user queries the system to get the dates when an event occurs, or alternately, he queries for the event set that happen during a given period.

These typical use cases assume that prior functionalities are to be offered, for instance: extracting temporal information from original information sources, recording significant issues in appropriate formats, recording metadata as well, and providing means for data management, including data consistency checking.

Our proposal aims at providing constrained generic models and some tools that can be reused and adapted to various applicative contexts and domains. Nevertheless, we shall here focus on some examples stemming from tourism leisure. The main questions in this specific context concern the opening or closing hours for registration, the schedule of a show, and so on. The set of time assertions below**,** are instances of the kind of information we intend to process.

(1) The museum is open every day except Tuesday and the following French holidays: December 25, January 1, May 1, and August 15. Opening hours: Monday, Thursday, Saturday, Sunday: from 9 a.m. to 6 p.m. Wednesday, Friday: from 9 a.m. to 10 p.m.
(2) Opening times: Monday to Friday: Lunch (12 noon to 2 p.m.) Dinner (6.30 p.m. to 11 p.m.). Saturday: Dinner (6.30 p.m. to 11 p.m.)
(3) Opened every day from 10:00 to 18:00, except Tuesdays.

We shall pay a special attention to first: the extraction process of temporal information from English texts, second: the specification of an Object Model that stands as a pivot representation for temporal information being either intensional [4] or extensional, and third: the connection between the two. Indications will be given about some ways to check data consistency and achieve data visualization. The entire workflow interoperates between technical spaces, i.e. texts, linguistic models, object models and user interfaces. Our contribution pertains to Model Driven Engineering (MDE, [5]) which explicitly references common metamodels and standards.

The next section gives an overview of related work. Section 3 and 4 respectively describes the architecture of our proposal and provides excerpts of the Object Model. Section 5 is dedicated to temporal knowledge acquisition and presents the steps going from an English text up to a set of structured well identified temporal expressions. We also present a formal grammar which can be used as a controlled language to edit the model instances. We conclude by listing the key issues of our work and outline the future developments.

2 Related work

One purpose of developing Natural Language Processing (NLP) resources and services for temporal data capture is to ease the process of populating a knowledge base. Within the Semantic Web community and in the context of knowledge acquisition, an important issue is to automate (partly or completely) the process that captures information provided in texts, in order to make it computable for software components [6]. Many research projects take more specifically interest in temporal information expressed in texts, designing annotation tagsets that describe their semantics, such as the TimeML project [7]. A challenging task tackled by many

research projects is to transform calendar expressions found in texts into a structured and computable format, such as iCalendar or ISO 8601 standards: for instance, they aim at anchoring the situations described in texts on a timeline [8], [9], [10].

Our process explicitly distinguishes the linguistic representation of time from the calendar representation of temporal data. Indeed, the linguistic representation of time is external to its social representation [11], which, in a rationalizing effort, progressively stabilized itself in the standard calendar representation that uses concrete dates and describes temporal intervals. Natural language can define periodic references ("*on Mondays"*) or vague references ("*somewhere around mid March"*). Moreover, expressions like complex aggregates can build a new temporal reference on top of another (e.g.: "*on the 1st Wednesday of every Month", "two days later"*) or in link with the enunciation process (e.g.: "*today"*, "*this week-end"*). OWL-Time [2] took interest in modeling calendar references, without reducing them only to their numeric representation. This formal ontology can model complex temporal aggregates and periodic references. A significant aspect of our approach, in front of existing standards [2] and models, is that it clearly distinguishes different modeling layers, each addressing different users and roles: a linguistic modeling of temporal references for natural language processing, business-case models for end-users applications, and a generic pivot model to map them all [12].

The calendar expressions' linguistic model reused here is presented in [13] and [14]. It describes the semantics of temporal adverbs as decomposable units calling upon a compositional interpretation of their significance. Calendar expressions are considered here as a conjunction of semantic operators (*since*, *by the end of*, *during*, *before*, etc) interacting with calendar base references.

3 From texts to structured data

This section presents the global architecture (Fig. 1) of our proposal and describes the main workflow. The three major processes correspond to the population of the three different models that are manipulated: the linguistic model of calendar expressions, the calendar model (extension of iCalendar) and, in between, the pivot model (PivotObjectModel). The temporal knowledge acquisition system processes original data which consist of natural language texts. A first step is performed by TextFiltering and TextAnnotation modules which identify, extract and decompose the semantics of calendar expressions found in texts, thanks to lexicon lists and local grammars implemented in the form of Unitex[1] transducers. The output is post-treated to lift ambiguities and to instantiate the linguistic model.

A model transformation is applied to populate the PivotObjectModel with the instances of the linguistic model. A second model transformation is used to translate intensional expressions stored in the PivotObjectModel into their extensional equivalent counterpart (e.g.: "*on Mondays*" is transformed in "*on Monday, Jan 2", "on Monday, Jan 9*", etc). The output is depicted on a calendar widget that is used to correct possible misinterpretations, remove ambiguities or complete incomplete statements. A

[1] Unitex: http://www-igm.univ-mlv.fr/~unitex

third model transformation can be used to translate PivotObjectModel instances into the iCalendar standard (e.g.: for an export towards search engines). The PivotObjectModel can capture temporal information either from the output of the TextAnnotation process or from the CalendarEdition module. Data from any other source could be integrated thanks to a plain mapping of concepts between the sources and the PivotObjectModel.

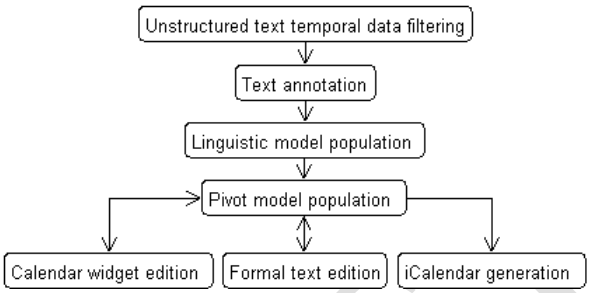

Fig. 1. General Workflow of the system.

The PivotObjectModel extends the ISO19108 standard. Additional features are provided so that more semantics could be handled: e.g. intensional periodical occurrences, relative time positions between events ("*opening at lunchtime"*, lunchtime temporal properties being specified separately). The PivotObjectModel can readily bind temporal properties to these events. It provides an invariant structure that can easily be interfaced with other modules, either for populating the model or to retrieve temporal information.

Thanks to a formal grammar, all model instances can automatically be translated into a textual representation - a controlled language - and hence can without ambiguity be understood by a human operator as well as deterministically processed by a computer. In the wake of Natural Language Generation techniques, which produce natural language text from structured knowledge [15], it provides a controlled textual rendering of the model, so that user can easily understand/modify model instances. It enables an interaction between linguistic data and knowledge models, providing a manual edition of complex model instances.

4 Temporal Modeling

In this section we describe the pivot generic temporal model (PivotObjectModel) which centralizes the information from the source providers. We propose a general UML [16] object model for temporal events properties.

We selected the ISO19108 standard as a reference for modeling the basic concepts: Instant and Period. The main reasons for this choice are the following: (i) the object representation of the ISO19108 proves to be well fitted for being used in MDE as a pivot representation; (ii) the ISO191000 series treats of geographical information issues which are very commonly associated with temporal features.

4.1 Periodic Rule Model

For sake of brevity, we only discuss selected excerpts of our model[2] and exclude the part dedicated to exceptions. Let's first focus on the central concept in Fig. 2. The PeriodicRule class is the root element for defining periodicity issues about a PeriodicTemporalOccurrence. A PeriodicTemporalOccurrence is an aggregation of PeriodicRules. Each aggregated element indicates a simple periodic phenomenon (i.e.: only one Frequency). The composition of all elements in the set, results in the sum of the simple components. Consequently, the first property of a PeriodicRule is its Frequency. According to a common definition, a Frequency is a pair of values respectively indicating the number of occurrences (times attribute) that happen during a special time span (referenceDuration role).

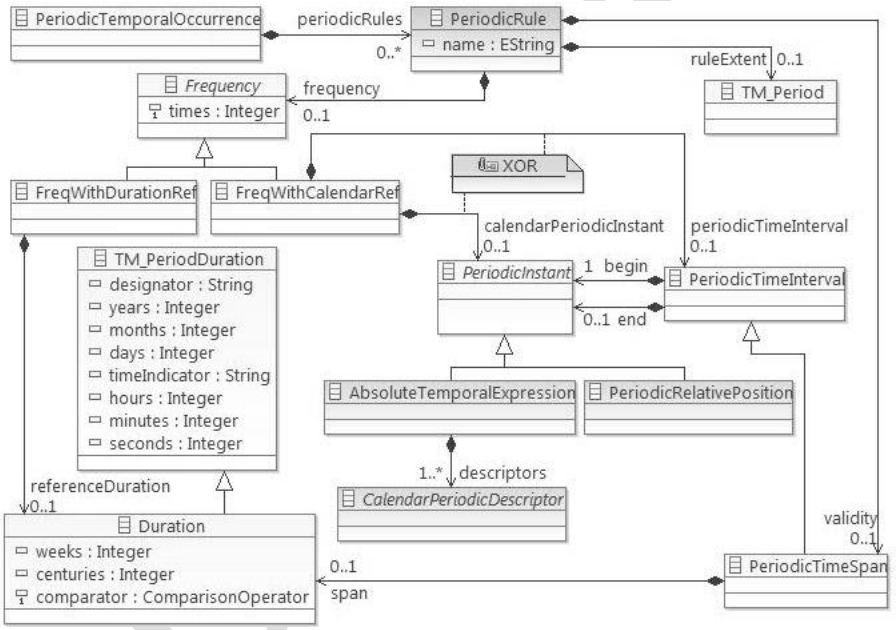

Fig. 2. Excerpt of the PeriodicTemporalOccurrence model.

As shown in Fig. 2, referenceDuration ends in a Duration datatype. This might be too restrictive in practice, since only durations could then be referenced. Thus, we give access to the whole set of AbsoluteTemporalExpressions for specifying the beginning and the end of intervals *via* the role periodicTimeInterval. Intrinsic periodic CalendarPeriodicDescriptors can be specified (e.g.: "*each Monday*", "*each first Wednesday*"), providing means to specify the major calendar units. Instants may also be specified by adding a NumericRank to a calendar unit: e.g.: "*3rd Sunday, 28th week*".

The optional startTime attribute is specified for frequency, in order to anchor the first periodic phenomenon occurrence on a concrete calendar, i.e.: to define its phase

[2] All the classes prefixed by TM_ come from the ISO19108 standard.

once its frequency is known. As mentioned above, if no referenceDuration is given for a PeriodicRule, then a PeriodicTimeInterval must be specified with two properties, namely begin and end, which are AbsoluteTemporalExpression. Of course, constraints are to be checked, e.g.: begin precedes end for all occurrences, and both begin and end should have the same frequency), but begin and end occurrences may present a phase difference. This means that the length of PeriodicTimeInterval occurrences is not necessarily equal (e.g.: PeriodicTimeInterval occurring *"from the first Tuesday to the last Monday of each month"*). The class PeriodicRelativePosition is used to specify a PeriodicInstant in relation with another previously defined.

4.2 Rule Extent and Periodic Time Span

A periodic phenomenon is infinite. Time boundaries can however be provided, for instance to identify a starting point. The optional ruleExtent role specifies the period during which the PeriodicRule applies. The association end is a TM_Period with a beginning and an optional end. The semantics of ruleExtent is that all occurrences are valid inside the extent and invalid otherwise. A fixed time extent may prove insufficient to capture some situations which are not scarce among periodic events. As a matter of fact, the extent should itself often be periodic. This is the case in the following assertion: "*the event occurs each first week of the month from March to September"*. Therefore, a PeriodicTimeSpan is defined to specify the periodic time window: "*from March to September*".

5 Capturing Temporal Knowledge of Access Periods

The generic workflow for temporal knowledge acquisition (Section 3) can be specialized in different domain specific cases. We studied a use case capturing access period information, i.e. the temporal properties that define the opening and closing hours of a given location. Such information is valuable for applications that intend to answer queries such as "*Which restaurants are open tonight after 10 p.m.?*" or "*Is there a supermarket opened on Sunday morning?*". In accordance with the generic workflow, different strategies are implemented to ease acquisition: text filtering and annotation, calendar edition, formal text edition, to provide a controlled language view.

5.1 Access period information

Considering examples that define accessibility, it appears that the semantic decomposition of access periods is complex. They are composed of various temporal references: periodic references (*"opened every Monday"*), temporal intervals (*"from January 1st to March 11th"*), exceptions (*"except Tuesday"*), and specification of temporal units with different granularities ("*on Monday*, *from 8am to 8pm"*).

All these information are reflected in the linguistic model we propose (Fig. 3), which aims at being consistent with the way access periods are linguistically built up.

An AccessPeriod is defined by one or several AbstractCalendarExpressions, which stand for CalendarExpressions or CalendarExpressionIntervals. Over a base or a set of base AbstractCalendarExpression, the definition of an AccessPeriod can aggregate exception or specification by using AbstractCalendarExpression. A CalendarExpressionInterval is linked to two CalendarExpressions, one of which is the start, the other being its end. A CalendarExpression (the core object in this model) is composed of the following main attributes: (i) the different time units that might enter in their composition, (ii) parts of day, parts of month and parts of year. A more detailed description of the linguistic model of calendar expressions is presented in [13].

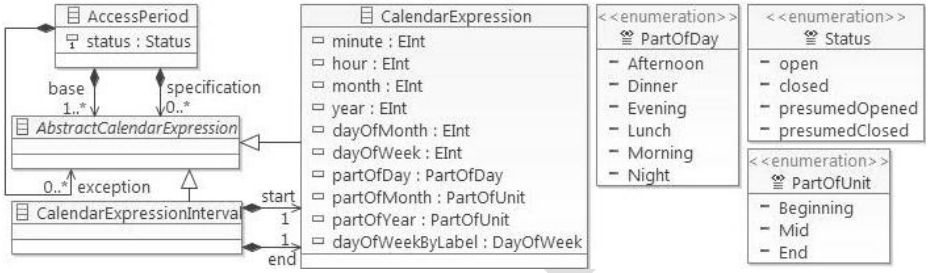

Fig. 3. Proposition for a linguistic modeling of access periods.

5.2 Text Annotation

Natural language analysis is very convenient to enter periodic and complex temporal references defining periods of accessibility. Instead of filling complex forms, users can simply define access periods in natural language. Expressions denoting access periods are analyzed by a set of transducers (Finite State Machines) defined thanks to the NLP platform Unitex. The output breaks down the semantics of access period and describes the way they are linked to one another. The annotation tagset contains metadata for the temporal references and the relation between calendar expressions such as exception ("*Opened every day except on Wednesday"*) or temporal granularity specification ("*on Sunday, from 9 a.m. to 12 p.m."*).

The annotation module is coupled with a normalization tool which completes and converts text annotation metadata into a structured format that corresponds to the linguistic model described in Subsection 5.1. After the annotation process, an automated post-treatment is necessary to lift ambiguities, in particular about the range of *exceptions* or *specifications*. For instance, in the expression "*Opened every day except on Tuesday, from 9 a.m. to 10 p.m.*", the specification (*from 9 a.m. to 10 p.m.*) should not be attached to the exception (*except on Tuesday*), but to the base expression (*every day*). This post-treatment also unfolds elliptic phrases (*closed on Tuesday and December 25*) and rebuilds their semantics in an explicit form (*closed on Tuesday and closed on December 25*). In its current state, the annotation process, nevertheless, shows limits. A simple rewriting process is sometimes required in order to filter external knowledge (such as school holiday, undefined bank holiday, or any lexicon that is external to the definition of temporal properties). For instance, for

being fully interpreted by the annotation process, the *e.g. 2* listed in the introduction can be rewritten and simplified in the following way:

> **e.g. 2:** *Opening times: Monday to Friday: Lunch (12 noon to 2pm) Dinner (6.30pm to 11pm). Saturday: Dinner (6.30pm to 11pm)*
> **e.g. 2 rewritten:** *Opening times: Monday to Friday, from 12 noon to 2pm and from 6.30pm to 11pm. Saturday, from 6.30pm to 11pm*.

For now, the annotation process resources only cover the lexicon and structure of calendar expressions commonly used when defining access period. It could though progressively be extended to cover a larger lexicon.

5.3 Annotation process: elements of evaluation

As a first evaluation of the annotation process, we submitted a corpus of 400 access period expressions to the text annotation module. These expressions were manually collected on various Web sites (of restaurants, theaters, etc). The *precision rate*, which evaluates the quality of the annotation output compared to the one of a human operator, is 82.25%. The scoring methodology was classic and straight: a well annotated expression receives a score of 1, any incorrect/incomplete annotation receives a score of 0. 71 expressions were not correctly interpreted and needed a simple rewriting process.

The interpretation process most commonly fails (1) when some external knowledge is required (such as the period covered by school holiday in the expression "*closed during school holidays*") and (2) when the system faces ambiguous definitions. For instance, the expression "*Opened Saturday and Sunday morning"* is ambiguous since it could mean that the location is open all Saturday or only Saturday morning. The system is deterministic. Then only one interpretation must be retained (in this case, only the first interpretation is given). These limits in the annotation process are not truly troublesome in the actual workflow, since the system is conceived to assist human operators who can modify the input, in case of unsatisfactory analysis.

5.4 Pivot model population, Temporal reasoning and Edition

The translation of linguistic model instances into pivot model instances is achieved by model transformations written in Kermeta[3], which convert CalendarExpressions from the linguistic model into PeriodicRules conforming to the pivot model.

During this process, the module faces two main difficulties. (1) False contradictions management. For example, when dealing with the following expression "*Opening days: every day from 9am to 10pm. Closed on Tuesday*" the module raises a conflict: while a reader understands that "*Closed on Tuesday"* prevails over the former access period. The system wrongly detects contradictory information when it deals with Tuesdays, which are considered being opened and closed at the same time. (2) Symbolic or vague temporal unit management. The transformation from linguistic

[3] Kermeta: http://www.kermeta.org

representation to concrete calendar representation can not be straightforward for symbolic or vague temporal units, such as: *"morning", "end of November"*, etc. In such cases, the system relies on a set of mapping rules that can be parameterized (e.g. convert "*morning*" into a time interval e.g.: "*from 8am to 12 am*").

The calendar widget (Fig. 4) is the user interface for editing the annotation: it offers a convenient way for checking consistency of the text input's analysis.

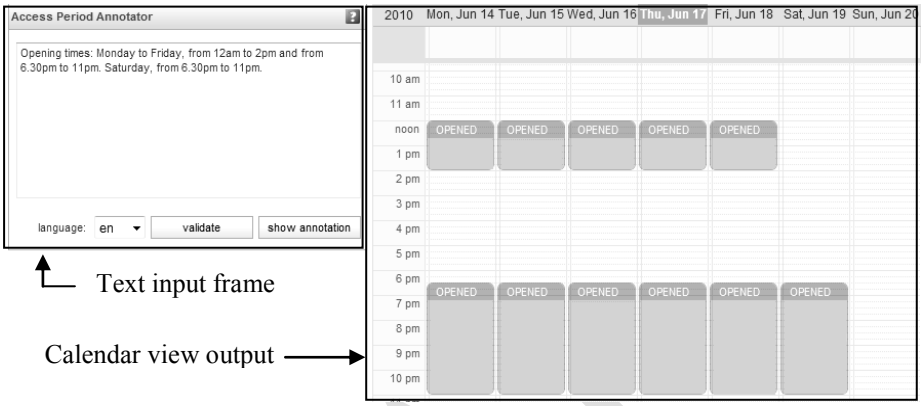

Fig. 4. Concrete calendar view of abstract access information[4] for e.g. 2.

In connection with the PivotObjectModel, a textual grammar is specified and offers another convenient way to read the content of the data stored as instances of the pivot model. It allows an automated translation of any rule from the model into an equivalent counterpart expressed in a readable controlled language thus allowing the user to check the modeled data semantics. The translation is bi-directional and implemented with xText[5]. Fig. 5 shows an example of the grammar translator output with respect to the opening specification of a shop as: "*from Monday to Friday, 10 a.m. to 8 p.m. except Thursday"*. The generated text is using one periodic rule *"10 a.m. to 8 p.m."* and a time span which is also periodic "*from Monday to Friday"*.

```
// rule 1: 10 a.m. to 8 p.m. - rule: from each 10th hour to each 20th hour
// from Monday to Friday       using a time span as from each Monday to each Friday
// except Thursday             except each Thursday
```

Fig. 5. Excerpt of text generated from pivot model instances with the grammar translator.

6 Conclusion and future work

The generic workflow presented in this paper considers the temporal knowledge acquisition as a process which goes along from an unstructured text analysis to a

[4] TKA (Temporal Knowledge Acquisition): http://client2.mondeca.com/AccessPeriodEditor/
[5] xText: http://www.eclipse.org/Xtext/

structured and computable model of temporal data, which can then be presented through a controlled language representation. This controlled language can stand for a surrogate of the pivot model instances. In this workflow, the quality of the information analysis can be controlled by users who can interact with the calendar widget or with the controlled language representation, in order to create, correct or delete information. The model transformation component bridges both the linguistic model of temporal expressions and an iCalendar compliant model to the generic pivot model. In the specific use case presented to test this workflow, human operators define complex temporal information in a simple way, enter access period definition in a textual form and visualize the system's interpretation on a calendar widget.

Further work will consider integrating the system with a query engine, in order to build a Semantic Portal in which temporal filters could be used.

Acknowledgments This project is partially granted by the ANR RMM2 project.

References

1. International Organization for Standardization: Text of 19108 Geographic information - Temporal schema, 55 p. (2002)
2. Hobbs, J. R. & Pan, F.: An Ontology of Time for the Semantic Web. In: ACM TALIP. Special issue on Temporal Information Processing, Vol. 3, No. 1, p. 66-85 (2004)
3. Dawson F., Stenerson, D.: Internet Calendaring and Scheduling Core Object Specification (iCalendar) - RFC2445, RFC Editor (1998)
4. Carnap, R.: Meaning and Necessity. University of Chicago Press (1947)
5. Schmidt, Douglas C.: Model-Driven Engineering. IEEE Computer Society (2006)
6. Bontcheva, K., Cunningham, H.: The Semantic Web: A New Opportunity and Challenge for Human Language Technology. In: 2nd ISWC Proceedings, Workshop on HLT for The Semantic Web and Web Services. 20-23 October 2003. p. 89-96. Florida (2003)
7. Pustejovsky, J., Castano, J., Ingria, R., Sauri, R., Gaizauskas, R., Setzer, A. & Katz, G.: TimeML: Robust Specification of Event and Temporal Expressions in Text. In: IWCS Proceedings. Florida (2003)
8. Mani, I. & Wilson, G.: Robust temporal processing of news. In: Proceedings of the 38th ACL, p. 69–76. Hong Kong (2000)
9. Setzer, A., Gaizauskas, R.: Annotating Events and Temporal Information in Newswire Texts. In: 2nd LREC Proceedings, p. 64-66. Athens (2000)
10. Schilder, F., Habel, C.: From Temporal Expressions to Temporal Information: Semantic Tagging of News Messages. In: ACL'01 Proceedings, Workshop on temporal and spatial information processing, p. 65-72. Toulouse (2001)
11. Benveniste, E.: Problèmes de linguistique générale. Vol. 2. Gallimard (eds). Paris (1974)
12. Bézivin, J.: On The Unification Power of Models. Software and System Modeling 4(2):171–188 (2005)
13. Battistelli, D., Couto, J., Minel, J-L. & Schwer, S.: Representing and Visualizing calendar expressions in texts. In: STEP'08. Venise (2008)
14. Teissèdre, C., Battistelli, D. & Minel, J-L.: Resources for Calendar Expressions Semantic Tagging and Temporal Navigation through Texts. In: 7th LREC, 19-21 May 2010. Valletta (2010)
15. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Journal of Natural Language Engineering, Vol. 3 Part 1 (1999)
16. OMG, Unified Modeling Language (UML): Superstructure. Version 2.2 (2009)