Micro-systemic Linguistic Analysis and Software Engineering: a synthesis

CARDEY Sylviane
GREENFIELD Peter
Centre Lucien TESNIERE
Université de Franche-Comté

Résumé :
Dans cet article nous essayons de montrer que lôanalyse linguistique micro-systémique comporte des propriétés inhérentes qui satisfont certaines notions pertinentes du génie logiciel et de plus, que ses fondations mathématiques apportent les méta-descriptions dont a besoin le processus de développement du logiciel.

1. Introduction

In this paper we endeavour to show that micro-systemic linguistic analysis has certain inherent features which satisfy notions pertinent to software engineering and furthermore that its mathematical underpinnings provide the meta-descriptions required for the software development process.
To this end, the methodological approach that we adopt is to view the potential synthesis from two points of view, one being that of micro-systemic linguistic analysis and the other engineering. The notions that are pertinent for each are summarised in the following diagram:

Micro-Systemic Linguistic Analysis	**Engineering**
Capturing micro-systemic linguistic analysis	Application of engineering principles to micro-systemic linguistic analysis
Examples	**Examples**
Generic model mathematically based Compositionality Explicit constraints: Global: correct (no falsehoods) complete (no more, no less) Local: nested Inherent systemic traceability	Engineering: Project management Quality management Linguistic Engineering: Corpus analysis Performance & Competence Software Engineering Lifecycle Configuration management Linguist as user **AND** programmer Formal Specification Performance

It will be observed that with œEngineeringô we have taken a wider perspective than the title of the paper would suggest; this is because we wish to show that software engineering has a contribution too to linguistic engineering in the context of micro-systemic linguistic analysis.

2. Micro-Systemic Linguistic Analysis

The notion of micro-system is due to Yves Gentilhomme (GENTILHOMME 1985). Micro-systemic linguistic analysis (CARDEY, S. 1987) is based on the postulate that a language can be segmented into individual systems based on the observation that such systems influence each other.

Micro-systemic linguistic analysis proposes that to be processed safely languages have to be decomposed into systems which can be analysed by a human being and by machine because they are small enough but also complete so as to be able to work together as a unified system. As well as this, the systems so delimited can interact with other such systems, and this interaction is a property of language. Nothing is independent; lexis, morphology, syntax are linked.

Applications of micro-systemic linguistic analysis are very varied (CARDEY & GREENFIELD 2006) ranging over:

Grammar teaching: the Studygram system

Disambiguated parts of speech tagging: the Labelgram system

Machine translation of 'far' language couples (including anaphoric reference and zero anaphor processing)

Grammar checking and correcting (including noun phrase identification)

Sense mining: the Classificatim system

Safety critical applications where evaluation ability is required in the form of validation and traceability as in controlled languages, for example:

aircraft cockpit alarm message vocabulary

machine translation of medical protocols.

3. The Methodology

Micro-systemic linguistics methodology consists in analysing a linguistic system in component systems as follows:

Sc: a system which is recognisably canonical;
Sv: another system representing the variants;
Ss: a 'super' system which puts the two systems **Sc** and **Sv** in relation with each other.

3.1 Establishing a Micro-systemic Linguistics Analysis

This requires modelling system **Ss**. To do so for some application, the linguist establishes two categorisations:
Firstly a 'non-contextual' (nc) categorisation of the canonical forms in relation with the variant forms in isolation, the context being limited to just the canonical and variant forms themselves.
Secondly an 'in-context' (ic) categorisation of the canonical forms in relation with the variant forms in terms of the linguistic contexts of the variant forms. The systemic analysis reveals precisely what other internally related linguistic systems are involved.
As to which categorisation (nc or ic) to start with and even whether it is possible or feasible to sequence the establishment of the two categorisation depends on various factors such as:
What prior knowledge is available? For example existing classifications as for example a parts of speech tag set; the simplicity or otherwise of organising observations including their extraction. For example in machine translation and in concept mining, concepts (semes) which will constitute the canonical forms are themselves often revealed during the analysis process at the same time as the contexts indicating their presence as variants in the language.

3.2 Mathematical Modelling

The mathematical modelling of the core structure in micro-systemic linguistic analysis is reported in detail in (CARDEY & GREENFIELD 2005). Being ócoreô we will not address the algebraic composition of micro-systems. In this paper our interest is to present those results which are conducive to engineering in general and software engineering in particular knowing that this mathematical model leads to a computational model. In this respect, the mathematical model represents the pivot between the formal linguist and the software engineer.

For systems **Ss**, **Sc** and **Sv**, let ***S*** be a set structure modelling super system **Ss**, let ***C*** the set of canonical forms, let ***V*** be the set of variant forms, and let **C*V*** be the binary ordered relation between ***C*** and ***V*** corresponding to system **Ss**.

Each of the above two categorisations, 'nc' and 'ic', can be modelled by a partition on ***CV***; we have ***P*nc** and ***P*ic**. Given that we have partitions, from the fundamental theorem on equivalence relations, it follows that there exist two corresponding equivalence relations ***E*nc** and ***E*ic** on ***CV***. Each equivalence class in respectively ***E*nc** and ***E*ic** corresponds to a distinct categorisation or case. We model system **Ss**, the super system relating systems **Sc** and **Sv**, by means of the ordered binary relation ***S*** between the equivalence relations ***E*nc** and ***E*ic**, and similarly $\boldsymbol{S}^{-1}$ between ***E*ic** and ***E*nc**.

We can subsequently model functions for finding the canonical element(s) corresponding to a variant element and vice-versa or others such as finding the (name of the) canonical equivalence class for a variant element as for example in parts of speech tagging. Furthermore, because we have a precise structure for ***S***, we can verify that a given linguistic analysis representation is well formed. In respect of equivalence relations and when the linguistic domain consists of strings, we note that every finite automaton induces a right invariant equivalence relation on its input strings (HOPCRAFT & ULLMAN 1969, pp. 28-30).

4. An Example Linguistic System

We use as an example ‘the doubling or not of the final consonant in English words before the endings -ed, -ing, -er, -est, -en’.

This system, *Doubling_or_not*, comprises the following component systems:

Sc*Doubling_or_not*, the words concerned in their basic form, that is their canonical form; e.g. 'model', 'frolic'

Sv*Doubling_or_not*, the words concerned in their derived or inflected form, the variants; e.g.
'model-ing', 'modell-ing', 'frolick-ed'

Ss*Doubling_or_not*, the super system relating systems **Sc***Doubling_or_not* and **Sv***Doubling_or_not*.

In respect of ***C*** (the set of canonical forms), ***V*** (the set of variant forms), and ***CV*** (the binary ordered relation between ***C*** and ***V*** corresponding to system **Ss**), for *Doubling_or_not* the linguist observes:

C_{Doubling_or_not} =
{é , model, frolick, é }

V_{Doubling_or_not} =
{é , modeling, modelling, frolicked, é }

CV_{Doubling_or_not} =
{é , (model, modeling), (model, modelling), (frolick, frolicked), é }

4.1 The Non-contextual Corpus Analysis

The non-contextual (nc) corpus analysis results in:

Non-contextual (nc) corpus analysis *Doubling_or_not*		
Canonical	**Variant**	**Attestation**
model	Modelling	WNCD 1981
model	Modelling	CEOED 1971
frolic	Frolicked	CEOED 1971

where the attestations are in:
WNCD = Webster's New Collegiate Dictionary
CEOED = Concise Edition of the Oxford English Dictionary
The 'non-contextual' (nc) categorisation leads to the following representation of system **Snc**$_{Doubling\_or\_not}$ (which can be stored as a spread-sheet):

System Snc *Doubling_or_not* ï Non-contextual (nc) analysis						
Conditions						
Id	**Condition text**					
cv	Word with final consonant in English taking -ed, -ing, -er, -est, -en					
cvd	Doubling of the final consonant					
k	The words terminating in -ic take ï ck					
Operators						
Id	**Operator text**					
N	No doubling of the consonant					
D	Doubling of the consonant					
K	The words terminating in ï ic take ï ck					
Algorithm with case justifications in organigramme form						
Line #	**Level**	**Condition**	**Canonical**	**Operator**	**Variant**	**Attestation**
0	0	cv	model	N	modeling	WNCD 1981
1	1	cvd	model	D	modelling	CEOED 1971
2	2	k	frolic	K	frolicked	CEOED 1971

4.2 Some Observations Concerning the Formalisation

A case based algorithmic analysis approach is used. The conventional representation of algorithm (nc) which above is in óorganigrammeô form is as follows:

```
if condition cv is true
    then    if condition cvd is true
            then    if condition k is true
                    then operator K
                    else operator D
                    fi
            else    operator N
    fi
fi
```

The analysis approach is by nature intuitionistic and constructive; conditions in the algorithm are by nature positive and not negative; in the logical representation of the algorithm no use is made of the excluded middle. In other words our mathematical model of an analysis is based on classical logic and it is constructive. Being constructive it lends itself to calculable operations. The model theoretic model of $\mathrm{Snc}_{\mathit{Doubling\_or\_not}}$ is as follows:

$$\begin{array}{llllll}
cv \wedge \neg\, cvd \wedge & & N \wedge \neg\, D \wedge \neg\, K & \vee \\
cv \wedge & cvd \wedge \neg\, k \wedge & \neg\, N \wedge \quad D \wedge \neg\, K & \vee \\
cv \wedge & cvd \wedge \quad k \wedge & \neg\, N \wedge \neg\, D \wedge \quad K &
\end{array}$$

In the model, each line contains a conjunction which corresponds to an interpretation which itself can be interpreted as a proof justification of the case. The set formulation of conditions components (nc) where set ***CV*** is the set engendered by proposition (condition) cv etc. is as follows:

Non-contextual (nc) analysis			
Algorithm		**Set name**	**Set formulation**
Line #	**Level**		
0	0	***CVnc*.0**	***CV* \ *CVD***
1	1	***CVnc*.0.0**	***CV* ž *CVD* \ *K***
2	2	***CVnc*.0.0.0**	***CV* ž *CVD* ž *K***

4.2.1 Partitioning (non-contextual) the Set CV

The sets ***CVnc*.0**, ***CVnc*.0.0** and ***CVnc*.0.0.0 partition** the set ***CV***:

The intersection of the sets ***CVnc*.0**, ***CVnc*.0.0** and ***CVnc*.0.0.0** is the empty set:

$\cap\{$ ***CVnc*.0**, ***CVnc*.0.0**, ***CVnc*.0.0.0**$\} = \varnothing$

The union of the sets ***CVnc*.0**, ***CVnc*.0.0** and ***CVnc*.0.0.0** is the set ***CV***:

$\cup\{$ ***CVnc*.0**, ***CVnc*.0.0**, ***CVnc*.0.0.0**$\}$ = ***CV***

4.2.2 Equivalence relations over CV

Non-contextual (nc)

The sets ***CVnc*.0**, ***CVnc*.0.0** and ***CVnc*.0.0.0** partition the set ***CV***. Being a partition, the algorithm has determined an equivalence relation ***E*nc** over ***CV*,** each of the sets ***CVnc*.0, *CVnc*.0.0 and *CVnc*.0.0.0** is an equivalence class., The number of equivalence classes, that is the index of the equivalence relation ***E*nc, inc,** is 3 (the number of lines in the algorithm).

In-context (ic)

A similar argument applies.

The index of the equivalence relation ***E*ic, iic,** is 13.

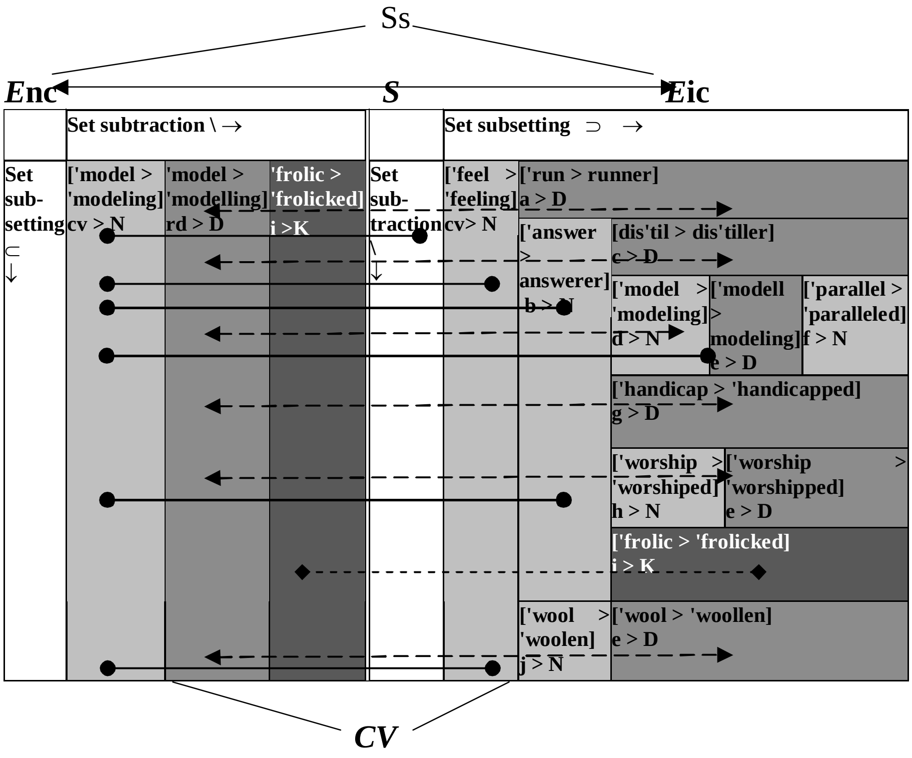
4.3 The In-context Analysis

The ǿin-context' (ic) categorisation leads to the following representation of system **Sic**$_{\mathit{Doubling\_or\_not}}$ (which can be stored as a spread-sheet):

Sic *Doubling_or_not* - In-context (ic) analysis						
Conditions						
<u>Id</u>	Condition text					
cv	word with final consonant in English taking ï ed, -ing, -er, -est, -en					
a	word of a syllable of the form C-V-C					
b	terminated by C-V-C or by C-V(pronounced)-V(pronounced)-C					
c	last syllable accented					
d	terminated by ï l or ï m					
e	used in England					
f	"(un)parallel"					
g	"handicap, humbug"					
h	"worship, kidnap"					
i	terminated par ï ic					
j	"wool"					
Operators						
Id	Operator text					
N	No doubling of the consonant					
D	Doubling of the consonant					
K	The words terminating in ï ic take ï ck					
Algorithm with case justifications in organigramme form						
Line #	Level	Condition	Canonical	Operator	Variant	Attestation
0	0	cv	feel	N	feeling	CEOED 1971
1	1	a	run	D	runner	CEOED 1971
2	1	b	answer	N	answerer	CEOED 1971
3	2	c	dis'til	D	dis'tiller	CEOED 1971
4	2	d	model	N	modeling	WNCD 1981
5	3	e	model	D	modelling	CEOED 1971
6	4	f	(un)parallel	N	(un)paralleled	CEOED 1971
7	2	g	handicap	D	handicapped	CEOED 1971
8	2	h	worship	N	worshiped	WNCD 1981
9	3	e	worship	D	worshipped	CEOED 1971
10	2	i	frolic	K	frolicked	CEOED 1971
11	1	j	wool	N	woolen	CEOED 1971
12	2	e	wool	D	woollen	CEOED 1971

4.4 Formulation of the Super System

The super system **Ss*Doubling_or_not*** is formulated as the ordered binary relation ***S*** between the equivalence relations ***E*nc** and ***E*ic,** each over ***CV*,** shown here with the materialisation of its associated graph.


Ss
Enc
S
Eic
Set subtraction \ →
Set subsetting ⊃ →
Set sub-setting ⊂ ↓
['model > 'modeling] cv > N
'model > 'modelling] rd > D
'frolic > 'frolicked] i >K
Set sub-traction \ ↓
['feel > 'feeling] cv> N
['run > runner] a > D
['answer > answerer] b > N
[dis'til > dis'tiller] c > D
['model > 'modeling] d > N
['modell > modeling] e > D
['parallel > 'paralleled] f > N
['handicap > 'handicapped] g > D
['worship > 'worshiped] h > N
['worship > 'worshipped] e > D
['frolic > 'frolicked] i > K
['wool > 'woolen] j > N
['wool > 'woollen] e > D
CV

5. Exploiting the Mathematical Modelling

The examples are illustrated using the system *Doubling_or_not.*

5. 1 Computerised Source Representations

5.1.1 Use of Spreadsheets

Spread-sheets (idem) are convenient for capturing the development of the analysis and for storing source forms.

5.1.2 Liapunof of Shestopal Algorithmic Representation

The in-context algorithm:

algorithm("a ^ 1 D. v 1 b ^ 2 c ^ 3 D. v 3 d ^ 4 e ^ 5 f ^ 6 N. v 6 D. v 5 N.v 4 g ^ 7 D. v 7 h ^ 8 e ^ 9 D. v 9 N. v 8 i ^ 10 K. v 10 N.v 2 j ^ 11 e ^ 12 D. v 12 N. v 11 N.").

5.1.3 Binary Decision Tree Algorithmic Representation

The in-context algorithm:

```
condition(a) ->
   operateur('D',0) ;
   condition(b) ->
      condition(c) ->
         operateur('D',1) ;
         condition(d) ->
            condition(e) ->
               condition(f) ->
                  operateur(N,2) ;
                  operateur('D',3) ;
               operateur('N',4) ;
            condition(g) ->
               operateur('D',5) ;
               condition(h) ->
                  condition(e) ->
                     operateur('D',6) ;
                     operateur('N',7) ;
                  condition(i) ->
                     operateur('K',8) ;
                     operateur('N',9) ;
      condition(j) ->
         condition(e) ->
            operateur('D',10) ;
            operateur('N',11) ;
         operateur('N',12).
```

5.2 Algorithm Execution

The trace of an interactive execution of the in-context algorithm for the word ómodelô is as follows:

```
Microsystem: 'System Sic Doubling_or_not - In-context
(ic) analysis'
Version: '30:11:2006'
0:    cv: Word with final consonant in English taking -ed,
-ing, -er, -est, -en ? y/n : y
1:       a: Word of a syllable of the form C-V-C ? y/n : n
2:       b: Terminated by  C-V-C or by C-V(pronounced)-
V(pronounced)-C ? y/n : y
3:       c: Last syllable accented ? y/n : n
4:       d: Terminated by -l or -m ? y/n : y
5:       e: Used in England ? y/n : y
6:       f: (un)parallel ? y/n : n
5         Last true condition: e Canonical: model
5         Operator_Id: D : 'Doubling of the consonant'
Variant: modelling Justification: 'CEOED 1971'
Trace = 5 - [cv,\+ a,b,\+ c,d,e,\+
f,covj(model,[ovj('D',modelling,'CEOED 1971')])]
```

5.2 The Model-theoretic Model and Algorithms

The model-theoretic model of the micro-system can be generated from the algorithm:

```
Microsystem: 'System Sic Doubling_or_not - In-context
(ic) analysis'
Version: '30:11:2006'
Model:
0 - [cv,\+ a,\+ b,\+ j,covj(feel,[ovj('N',feeling,'CEOED
1971')])]
```

```
1 - [cv,a,covj(run,[ovj('D',runner,'CEOED 1971')])]
2 - [cv,\+ a,b,\+ c,\+ d,\+ g,\+ h,\+
i,covj(answer,[ovj('N',answerer,'CEOED 1971')])]
3 - [cv,\+ a,b,c,covj('dis''til',[ovj('D','dis''tiller','CEOED
1971')])]
4 - [cv,\+ a,b,\+ c,d,\+
e,covj(model,[ovj('N',modeling,'WNCD 1981')])]
5 - [cv,\+ a,b,\+ c,d,e,\+
f,covj(model,[ovj('D',modelling,'CEOED 1971')])]
6 - [cv,\+ a,b,\+
c,d,e,f,covj('(un)parallel',[ovj('N','(un)paralleled','CEOED
1971')])]
7 - [cv,\+ a,b,\+ c,\+
d,g,covj(handicap,[ovj('D',handicapped,'CEOED
1971')])]
8 - [cv,\+ a,b,\+ c,\+ d,\+ g,h,\+
e,covj(worship,[ovj('N',worshiped,'WNCD 1981')])]
9 - [cv,\+ a,b,\+ c,\+ d,\+
g,h,e,covj(worship,[ovj('D',worshipped,'CEOED
1971')])]
10 - [cv,\+ a,b,\+ c,\+ d,\+ g,\+
h,i,covj(frolick,[ovj('K',frolicked,'CEOED 1971')])]
11 - [cv,\+ a,\+ b,j,\+
e,covj(wool,[ovj('N',woolen,'CEOED 1971')])]
12 - [cv,\+ a,\+ b,j,e,covj(wool,[ovj('D',woollen,'CEOED
1971')])]
#Model = 13
```

In general there are many functionally identical algorithms that can be generated from a model theoretic model (the conditions and operators resting unchanged). Let P_N be the number of algorithms that can be generated from a model with N conditions, where all the 2^N possible interpretations are present. We have:

$$P_N = N \times (P_{N-1})^2$$

where $P_2 = 2$ (HUMBY 1973, pp. 32-34).

In consequence alternative functionally identical algorithms can be generated to meet specific needs such as speed optimisation in automated applications.

5.3 Sets Defined during the Algorithm Execution

The sets defined during the execution of the non-contextual algorithm are as follows (the same arguments apply to the in-context algorithm):

Non-contextual (nc) analysis			
Algorithm		**Set name**	**Set formulation**
Line #	**Level**		
0.	0.	***CV'nc*.0**	***CV***
1.	1.	***CV'nc*.0.0**	*CV* ž ***CVD***
2.	2.	***CV'nc*.0.0.0**	*CV* \ *CVD* ž ***K***

The sets so defined form a collection of proper sub-sets (the same arguments apply to the in-context algorithm):

Non-contextual (nc) analysis	
Algorithm line #	**Parent set ⊃ Line set**
	***CV'nc*.0 ⊃ *CV'nc*.0.0**
	***CV'nc*.0.0 ⊃ *CV'nc*.0.0.0**

The sets defined during the execution of the in-context algorithm at the same (nesting) level and with common parent set are mutually disjoint:

In-context (ic) analysis			
Algorithm		**Set name**	**Set formulation**
Line #	**Level**		
0.	0	***CV'ic*.0**	***CV***
1.	1	***CV'ic*.0.0**	*CV* ž ***A***
2.	1	***CV'ic*.0.1**	*CV* \ *A* ž ***B***
3.	2	***CV'ic*.0.1.0**	*CV* \ *A* ž *B* ž ***C***
4.	2	***CV'ic*.0.1.1**	*CV* \ *A* ž *B* \ *C* ž ***D***
5.	3	***CV'ic*.0.1.1.0**	*CV* \ *A* ž *B* \ *C* ž *D* ž ***E***
6.	4	***CV'ic*.0.1.1.0.0**	*CV* \ *A* ž *B* \ *C* ž *D* ž *E* ž ***F***
7.	2	***CV'ic*.0.1.2**	*CV* \ *A* ž *B* \ *C* \ *D* ž ***G***
8.	2	***CV'ic*.0.1.3**	*CV* \ *A* ž *B* \ *C* \ *D* \ *G* ž ***H***
9.	3	***CV'ic*.0.1.3.0**	*CV* \ *A* ž *B* \ *C* \ *D* \ *G* ž *H* ž ***E***
10.	2	***CV'ic*.0.1.4**	*CV* \ *A* ž *B* \ *C* \ *D* \ *G* \ *H* ž ***I***
11.	1	***CV'ic*.0.2**	*CV* \ *A* \ *B* ž ***J***
12.	2	***CV'ic*.0.2.0**	*CV* \ *A* \ *B* ž *J* ž ***E***

5.4 Well Formed Representations

For a micro-systemic linguistics analysis representation to be well formed, two constraints must be met:

1. Proper sub-setting
2. Disjunction.

For automatic applications, as for example the Labelgram disambiguating tagger system (CARDEY & GREENFIELD 2003), both constraints can be processed

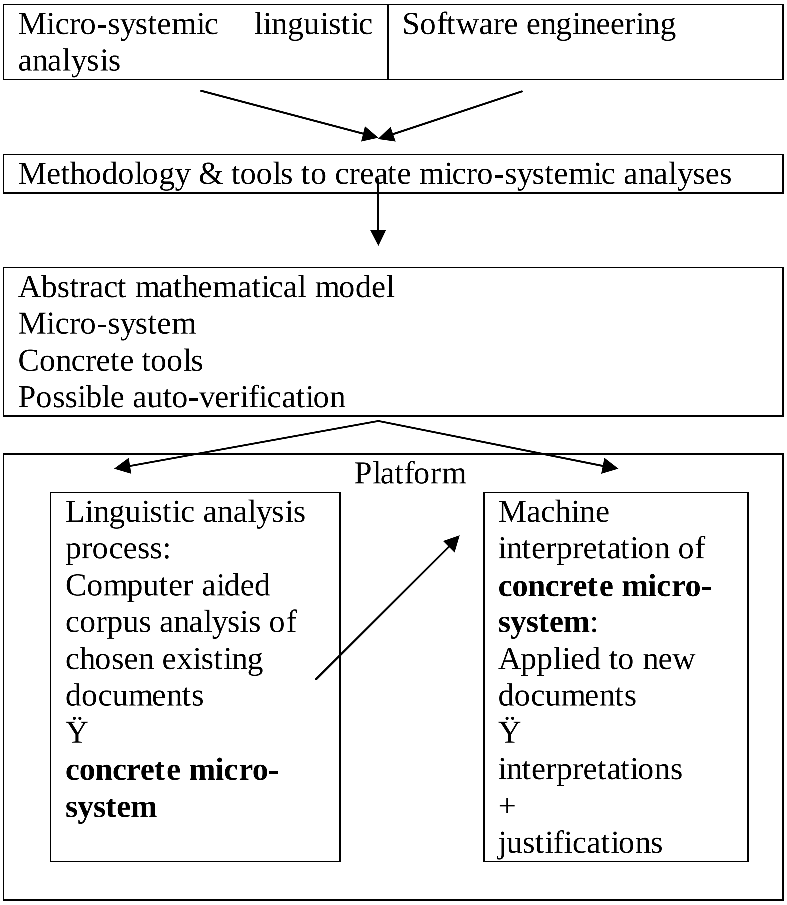
automatically using abstract interpretation techniques (ROBARDET 2003). As a result this enables:
automated verification
speed optimisation:
proper sub-setting : removal of redundant context constraints
disjunction re-ordering

5.5 Case Justifications

Being a partition with each equivalence class being associated directly with a line in the algorithm allows us to include a justification for each class. Each class corresponds to a distinct case. Case justifications can be:
competence (the linguist's) based
performance based in including an attestation; for example: observed in a corpus and corroborated by the linguist entry in a relevant dictionary (in particular for the lexis). Thus, absence of a case attestation would imply competence; it should be noted that scientific method requires that a theory be demonstrated by experimental evidence. Thus linguistic practice normally involves substituting competence based justifications (no attestation) by performance based justifications (attestation present).
Case justifications assist evaluation processes such as validation and regression testing and are essential in safety critical applications (CARDEY *et al.* 2006).
Case justifications, precisely because they are case based, can serve as the basis for evaluation benchmarks.

6. Conclusion

The synthesis of micro-systemic linguistic analysis and software engineering can be illustrated by the following diagram:


Micro-systemic linguistic analysis
Software engineering
Methodology & tools to create micro-systemic analyses
Abstract mathematical model
Micro-system
Concrete tools
Possible auto-verification
Platform
Linguistic analysis process:
Computer aided corpus analysis of chosen existing documents
Ÿ
concrete micro-system
Machine interpretation of **concrete micro-system**:
Applied to new documents
Ÿ
interpretations
+
justifications

References

- CARDEY, S. 1987. *Traitement algorithmique de la grammaire normative du français pour une utilisation automatique et didactique*, Doctorat d'état, Université de Franche-Comté, France.

- CARDEY S., GREENFIELD P. 2003. Disambiguating and Tagging Using Systemic Grammar. *In Proceedings of the 8th International Symposium on Social Communication*, Santiago de Cuba, Actas I, pp.559-564

- CARDEY, S., GREENFIELD, P. 2005. A Core Model of Systemic Linguistic Analysis. *In Proceedings of the International Conference RANLP-2005 Recent Advances in Natural Language Processing*, Borovets, Bulgaria, 21-23 September 2005, pp. 134-138.

- CARDEY, S., GREENFIELD, P. 2006. Systemic Linguistics with Applications. *In Linguistics in the Twenty First Century*. Cambridge Scholars Press, United Kingdom, ISBN 1904303862, pp. 261-271.

- CARDEY S., GREENFIELD P., BIOUD M., DZIADKIEWICZ H., KURODA K., MARCELINO I., MELIAN C., MORGADINHO H., ROBARDET G., VIENNEY S. 2006. The Classificatim Sense-Mining System. *In Advances in Natural Language Processing*, Springer-Verlag ï LNAI 4139, ISBN 3-540-37334-9, pp. 674-684.

- GENTILHOMME Y. 1985. *Essai dœapproche microsystémique, Théorie et pratique, Application dans le domaine des sciences du langage*, Peter Lang, Berne.

- HOPCROFT, J.E., ULLMAN, J.D. 1969. *Formal languages and their relation to automata*, Addison-Wesley Publishing Company.

- HUMBY, E. 1973. *Programs from decision tables*, Macdonald/American Elsevier, ISBN 0 356 04126 3/ISBN 0 444 19569 6.

- ROBARDET, G. 2003. *Vérification automatisée des règles de contexte du logiciel Labelgram*, Mémoire de maîtrise, Sciences du langage mention TAL, Besançon, Université de Franche-Comté, France.