Balisage: The Markup Conference 2010
Proceedings

Freestyle Markup Language

Specification of an intuitive, powerful, polyhierarchical new extensible markup language

Denis Pondorf
IT-Freelancer and PhD student
Hamburg, Germany
<contact@freestyle-markup.org>

Andreas Witt
Senior Researcher
Institute for the German Language (IDS), Mannheim, Germany
<witt@ids-mannheim.de>

Balisage: The Markup Conference 2010
August 3 - 6, 2010

How to cite this paper

Pondorf, Denis, and Andreas Witt. “Freestyle Markup Language: Specification of an intuitive, powerful, polyhierarchical new extensible markup language.” Presented at Balisage: The Markup Conference 2010, Montréal, Canada, August 3 - 6, 2010. In *Proceedings of Balisage: The Markup Conference 2010*. Balisage Series on Markup Technologies, vol. 5 (2010). DOI: 10.4242/BalisageVol5.Pondorf01.

Abstract

This paper provides a new generation of a markup language by introducing the *Freestyle Markup Language* (*FML*). Demands placed on the language are elaborated, considering current standards and discussions. Conception, a grammatical definition, a corresponding object graph and the bi-directional unambiguous transformation between these two congruent representation forms are set up. The result of this paper is a fundamental definition of a completely new markup language, consolidating many deficiency-discourses and experiences into one particular implementation concept, encouraging the evolution of markup.

Table of Contents

Introduction

Nowadays, the Extensible Markup Language (*XML*) is broadly accepted as a standardized serialization format and universal transfer syntax. According to the well-formedness constraints ("properly nested"), *XML* allows the markup of content only in strict monohierarchical structures – markup in non-hierarchical or multi-

hierarchical structures is not inherently provided in the language: This deficit is sufficiently discussed in the literature and represents a problem for many application scenarios.

> Unfortunately there is a large gap between the *XML* tree model and the models traditionally used in software engineering[...] Markup-based models revolve around order and hierarchy whereas the more popular graph-based data models revolve around linking relationships and roles. A linked data structure (directed graph) can more directly express sophisticated information than can a simple tree.
>
> — Prescod 2000

Moreover, further restrictions exist that complicate an unlimited use of *XML* in practice and prohibit an intuitive *freestyle* markup. Markup languages are not only used in the typographic field as originally intended. Mostly and increasingly markup applies to data structures in general. This fact confirms the necessity to further develop present markup standards, just as the data structures have evolved [Ernst 2009] starting from lists via table relations and trees to graphs.

The descriptive markup language, *FML*, provides an unrestricted *freestyle* markup that addresses problems and deficiencies with monohierarchical structure constraints. The term "*freestyle*" has been chosen to reflect the languages requirement to allow an intuitive-freehand markup.

According to the language taxonomy from Coombs 1987, *FML* is a generalized descriptive markup language. It is not restricted to any particular application scenario; it is a metalanguage, like *SGML*/*XML*, but more powerful since it supports more structuring features.

Requirement Analysis

> [...] it is time to determine what additional features are required from markup systems to make the formal description of such non-hierarchical phenomena straightforward.
>
> — Durand 1996

As a preparation 17 requirements for the project *FML* have been carefully identified and evaluated. The development of the new data-centric markup language demands the consideration of the following distinctive aspects:

1. **Grammar**
   For the syntax of the language a precise system of rules has to be defined. Only a clear grammar allows an automatic analysis of language instances. Each *FML*-document has to be syntactically constructed by deduction sequences and recognized as being wellformed by an automaton. In comparison with to the 107 production rules of *XML* we have to make the notation rules of *FML* clearer and more understandable, according to the following statement:

   > By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and in effect increases the mental power of the race.
   >
   > — Cajori 1928

   For the benefit of the requirements entropy, ergonomics and only restricted necessary *XML*-compatibility many obsolete *SGML*-constructs, redundancies and practical irrelevant relicts may be segregated. *FML* has to concentrate on the essential: polyhierarchical markup of texts and data.
   For *FML*-documents a Chomsky-type-2-grammar consisting of a minimal set of production rules

$$P \subset \{\ \varphi \longrightarrow \psi \mid \varphi \in V_N,\ \psi \in (V_N \cup V_T)^*\ \}$$

has to be provided in EBNF notation. Thus *FML* may be produced by a context-free grammar and recognized by a nondeterministic pushdown automaton.

2. **Compatibility**

   Nowadays *XML* is the lingua franca of the Internet and EDI-systems and probably the most common serialization format and transfer syntax.

   > What started out as a simple standard for electronic publishing has matured into one of the most important and widely used paradigms in distributed computing. The impact and influence of the standard and the conceptual base are visible in every area of computing today.
   >
   > — Adler 2006

   This fact deserves special attention. First of all *FML* syntactically will not be compatible with *XML* since it introduces new concepts and structures that *XML* could not prepare. Additionally critical constructs like *CDATA Sections* [Walsh 2003] and non-transferrable concepts like *DTD* will be excluded. Past studies often attach great importance to compatibility. This leads occasionally to quite complicated results and crude workarounds. *FML* defies all concerns regarding strict compatibility. That approach of course has advantages and disadvantages. Moreover *FML* will adopt as many accepted and well-proven *XML*-syntaxes and interpretations as possible without interfering other requirements. Thus an *FML*-document might be a wellformed *XML*-document but does not have to be. Anyway, there will be a high recognition factor.

   For supporting the compatibility issue two further steps are taken: Developing migration-guidelines *XML→FML* and setting up a *XML* representation of *FML* (milestone approach) (see section “XML Representation”).

3. **Monohierarchy**

   SGML-compatibility and the established OHCO-view

   > [...] we can describe a text as an "ordered hierarchy of content objects"
   >
   > — DeRose 1990

   lead to an inseparable bond of markup and monohierarchical structures. Generally *FML* follows the appraisal

   > [...]we now know that the breaking of strict hierarchies is the rule, rather than the exception
   >
   > — Durand 1996

   but still has to pay tribute to the broadly accepted monohierarchical understanding: *FML* needs a nested property for deciding structural compliance. Furthermore monohierarchies shall be marked up exactly the same way as in *XML*.

4. **Interference**

   Examinations about complex relations far beyond monohierarchies are coltishly made in an experimental novel (with more information in footnote-annotations than in the main content):

   > These pieces won’t halt: the boundary of a book is less than air to them. These pieces wink at each other, they shnoogle sighingly, they meet to confer, they part, they wave adieu and zip toward different mental planet zones, they reproduce, they tease us with coherence, they grimace and coil about and finagle, they repeat one another, they flaunt, they taunt, they sail away. Maybe only a deity – if deities exist – explains (or is) these splinters’ unity.
   >
   > — Goldbarth 2003

   Texts (and data in general as well) may not be interpreted as a hierarchy of content-components, as Schmidt 2009 concludes:

   > Indeed, it is now generally recognised that literary and linguistic texts frequently or even predominantly exhibit overlapping structures.

The challenge of interference structures (crossover-markup, concurring-markup, overlapping-markup) and *XML´s* inability to represent them natively

> Markup under any *SGML*/*XML* convention is unable to easily represent arbitrary structures or to represent overlapping or concurrent structures.
>
> — Durusau 2002

have been intensely discussed for many years [Sperberg-McQueen 2007] [Witt 2002] [Renear 1993].

FML is requested to handle this issue naturally and without applying any cumbersome workarounds like milestone, stand-off, fragmentation, virtual joins, or redundant encoding.

If the content of a sample document is the symbol-sequence

```
A man, a plan, a canal: Panama!
```

and gets augmented with the two typographic properties *italic* and **blue**

```
A man, a plan, a canal: Panama!
```

we will get the following document structure:

```
A man, a plan, a canal: Panama!
  italic
                  blue
```

FML must encode this simple issue not fragmented (as it would be done in *HTML*)

```
<i>A man, a plan, a <b>canal</b></i><b>: Panama</b>!
```

but native and intuitive:

```
<i>A man, a plan, a <b>canal</i>: Panama</b>!
```

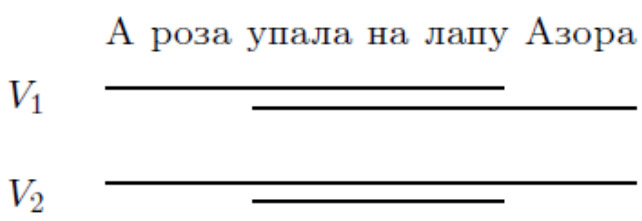
5. **Identification**

Start- and end-tags enclose the elements content. In *XML* corresponding start- and end-tags get identified by an equal tag-name. If more than one tag-pair with the same tag-name is involved, then identifying correspondence is ambiguous when allowing interference.

To illustrate the dilemma lets draw another typographic example and introduce the tag **u** with its semantic underline. The markup scenario

```
<u>А роза <u>упала на лапу</u> Азора.</u>
```

might be interpreted either as *V*1 or *V*2:



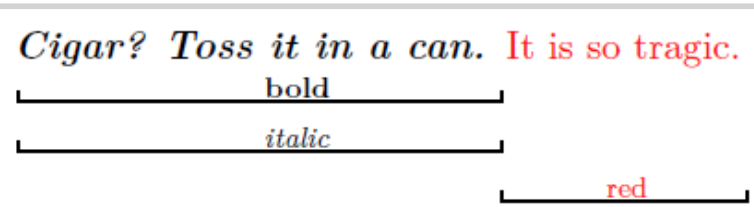
If a document contains *n* start-tags *Ts* with the same tag-name (and *n* end-tags *Te* with that tag-name), then the possible document interpretation *I* ranges between 1 and $n! = |Ts|! = |Te|!$. Thus, self-overlapping-markup is a special case of interference and may not be clearly interpreted without introducing a new concept:

FML must provide an mechanism (similar to the *co-indexing scheme* introduced in Huitfeldt 2001) for assigning an unique ID to a tag-pair. With ID´s, one start-tag gets unambiguously joined to a corresponding end-tag with the same name. Each assignment of an unique ID to a tag-pair decrements *I*. Tag-ID´s shall be optional. Due to compatibility, the Matrjoschka-doll-principle applies where the ID-mechanism is not in use.

6. **Congruence**

The principle of pluralism also applies to markup structures. In *FML* parallel markup must be possible. Parent-child-relationships between tags do not have to be enforced where there is no hierarchical relation.

Lets mark up content with the typographic properties bold (tag-name: **b**), italic (tag-name: **i**) and red (tag-name: **r**):



Without using any workaround constructs *XML* may mark up this structure inherently only by one of the two alternatives

Cigar? Toss it in a can. <r>It is so tragic.</r>

Cigar? Toss it in a can. <r>It is so tragic.</r>

But that is an overspecification. Neither is **b** a child of **i** nor is **b** parent of **i**.The content Cigar? Toss it in a can. is just **b** AND **i**. If an author would insert text between </**i**> and </**b**> then content would be produced that is **b** but not **i**. Insertion between </**b**> and <**r**> would produce detached content without any typographic properties. This lack of representing congruent markup may lead to inconsistencies during document maintenance. Also the original congruent structure may not be reconstructed from the markup representation.

Therefore *FML* has to provide a native tagging-mechanism for the treatment of congruent semantics and secure document maintenance.

7. **Independence**

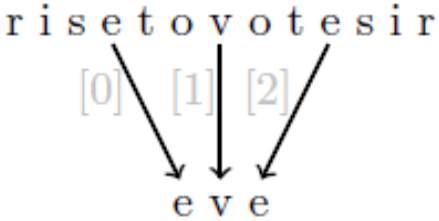
Heterogeneous structures have to be marked up in one redundant-free document. One semantic interpretation may exist besides other ones in the same document. Implied interference does not have to be subject to any restrictive structure constraints. Each markup-component in an FML document is non-ambiguously assigned to exactly one perspective (respectively default-perspective). During the transformation of an FML document to an FML graph particular perspectives may be optionally included or excluded. For an FML graph an unlimited number of perspective-nodes - sharing the same document content-nodes - will directly follow

the document-root-node.

This important topic has been discussed for a long time [Stührenberg 2006] [Witt 2007] and there are plenty of scenarios that would benefit from markup ability to handle independent structures. For instance, the *American National Corpus*[1] would not have to redundantly administrate their linguistic inventory seven times stand-off [Ide 2006]. They could administrate, search, edit, and analyze securely in one consolidated document.

8. **Segmentation**

   FML-elements may be composed via distributed ordered segments. By this means, the linear sequential ordering of content-symbols will be totally broken up. This is a rather terrific feature that takes advantage of the reusability-paradigm and facilitates arbitrary creations within a profane content.



   A new element may use content symbols from any position and in any order. For this purpose, segment-ID´s and order-ID´s have to be allocated to participating segment-tags.

   For instance this feature would allow to compose all words of the German language (120.000) in one redundant-free document having only the german alphabet as its content:

aAäÄbBcCdDeEfFgGhHiIjJkKlLmMnNoOöÖpPqQrRsSßtTuUüÜvVwWxXyYzZ

   Under the term *Discontinuous Structures* a comparable approach has been discussed in Sperberg-McQueen 2008.

9. **Fragmentation**

   Each fragment[2] of a wellformed *FML*-document must as well be a wellformed document. The cut-off points may be anywhere in the content and outside of embedded markup-components. One implication of that will be, that tag-pairs might break apart. Therefore, a mechanism of completion has to apply during transformation. The location of missing start- or end-tags can be assumed to be before or after the document borders.

   In the document prolog a document may optionally be declared as being a fragment. This way external processors get supported in identifying and composing distributed document-fragments. XInclude confirms the usefulness of this concept, that prepares further modularization and inclusion mechanisms:

> Many programming languages provide an inclusion mechanism to facilitate modularity. Markup languages also often have need of such a mechanism.

10. **Wildcard**

    Sometimes semantic objects, structural interruptions or relevant positions can be identified in a text- or data-sequence before adequate markup-tags are known for sure. *FML* shall offer a solution for those scenarios: A placeholder will be introduced. Embedding wildcards must be allowed everywhere in the content of a document. In due course the placeholder might be substituted for an empty-, start-, end-, multi-tag, processing instruction, comment, or content. The three states of validity (wellformed, nested, valid) are not influenced by wildcards.

11. **Inheritance**

    The attributes of elements shall be passed to their children. True to the statement of J. W. v. Goethe:

That which you've inherited from your fathers, earn it in order to possess it.

A typographic scenario helps to illustrate this concept:

able *was i ere i saw* elba

If the properties *italic* an **bold** get marked up

```
01          <palindrome>
02              able
03              <format italic="true">
04                  was i
05                  <format bold="true">
06                      ere
07                  </format>
08                  i saw
09              </format>
10              elba
11          </palindrome>
```

then it is obvious, that the inner format element (row 6) implicitly owns the property `italic="true"`.

This mechanism equates to the object-oriented paradigm inheritance [Firesmith1995] and observations about human knowledge structure and semantic networks [Collins 1969].

For *FML* the following inheritance rules are set:

- An elements attributes also apply for its child elements.
- Child attributes dominate parent attributes.
- Multiple inheritance leads to lossless consolidation in an ordered attribute-value list.
- Queries proceed in the direction child→parent.

Therefore an element might have far more attributes than it declares itself.

Neither does this feature affect the syntax of an FML document (except introducing an ordered attribute-value list) nor the properties of an FML graph. But the concept must apply when attributes are retrieved within an implementation of accessors.

12. **Graph Representation**

There are two representations of an *FML*-instance:

1. FML document
2. FML graph

Each wellformed *FML*-document must have an unambiguously corresponding graph. The graph will interpret and visualize the document and is the basis for graph

operations and for a data model within an implementation.

FML follows the fundamental statement “The computer representation of a document should reflect what really is” [DeRose 1990]. The FML graph must be a consistent polyhierarchical graph following the evolution of data structures:

> Long, long ago, people stored data in lists, because that was all that was available. Then, someone came up with the idea of storing data in tables. So relational databases came along and people moved up the ladder to tables. A few years ago, XML came along so data moved up again to trees. Can you guess what will happen next? The Semantic Web folks want us to move to using graphs. Should we move to graphs? Seems to be the next logical step in information evolution. What’s holding us back? Well, it’s probably too soon. The world is still in the tree phase.
>
> — Idehen 2003

13. **Transformation**

    Any derivation sequence of the context-free grammar of an FML document will lead to a monohierarchy. But the FML graph is polyhierarchical. Constructions have to apply, since document grammar is different to document interpretation. A system of rules for the bidirectional transformation $FML\ document \rightleftharpoons FML\ graph$ has to be defined. For both translation directions it must be possible to include relevant or to exclude irrelevant perspectives.

14. **Unambiguousness**

    An *FML* instance has to be unique. As well the interpretation of an FML document by an FML graph and the transformation between grammar and graph. In particular the following requirements have to be considered:

    - All syntactically identifiable function units (components) of an FML document must relate to corresponding nodes in an FML graph.
    - All nodes of an FML graph must relate to corresponding components.
    - All encapsulated informations in the nodes of an FML graph must relate to grammar-units.
    - The bidirectional transformation must be unambiguous.
    - No loss of information during transformation. Each change in one of the *FML* instance representation forms leads to a predictable change in the other one.

    For *SGML*/*XML* that requirement gets discussed with the terms *unambiguous markup* and *deterministic content model* [Brüggemann 1993]. Unambiguousness guarantees integrity and consistency of *FML* instances, secure interpretation and reliability during information exchange with *FML*.

15. **Entropy**

    A markup meta-language has no influence on the structured content. *FML* may not regulate content redundancies, but controls the syntactic rules for markup. Intensive use of markup increases document volume. The real content may shrink to only a fraction of the markup document.

    Compared to *XML* the markup coding expenses have to be reduced according to the principle of syntactical minimalism. *XML* deals very lavishly with markup characters. An example is the syntax for comments:

    ```
    <!--comment-->
    ```

    Why 'waste' 7 symbols? That style of encoding is not primarily subject to the requirement high entropy or minimal redundance.

    > The utility of a language as a tool of thought increases with the range of topics it can treat, but decreases with the amount of vocabulary and the complexity of grammatical rules which the user must keep in mind. Economy of notation is therefore important. Economy requires that a large number of ideas be expressible in terms of a relatively small vocabulary.
    >
    > — Iverson 1980

An FML document shall use only a minimal and necessary amount of information content for markup control characters. On the other hand notation minimalism must avoid an extensive overload of symbols. Therefore *FML* will use a psychological optimum considering the requirements compatibility and ergonomics.

16. **Ergonomics**

Historically the most important processor of documents is the reading and writing human [Fischer 2004].

Several kinds of markup (punctuation, decorative, orienting, structuring, informative, commenting and print markup) apply to documents for supporting the natural reading process. For the data-centric approach “markup has nothing to do with publishing”, Brian Reid, the developer of *Scribe*, pointed out in the conclusion of Reid 1998. Now *FML* is a generalized and not a typographic markup language and respects the data-centric approach. Still the established document-centric ergonomic demands are transferable to the human-*FML* interaction:

- Convenience: providing necessary and minimize dispensable functionality,
- Usability: high quality, capacity and performance,
- Perceivability: technological transparency, clarity and unambiguousness,
- Readability: comprehensible documents and models,
- Learnability: minimal training period,
- Documentation: covering all information needs required by users.

Additional criteria apply for an implementation.

Generally, FML documents may be much more complex (more difficult to read and to process) than *XML*-documents, since for a human mind they are not as easy to capture, as monohierarchies are. In spite of that, authors have a higher degree of freedom: Documents may be marked up natively and without any synthetic constructs. Structure breaks do not have to be restored. Insertion tasks are less critical. Document-interpretation with an FML graph is transparent.

Another ergonomic requirement is proposed: the *freestyle*-criteria. The term *freestyle* indicates that the performer of an activity may determine the way of technical execution to a certain extent by himself. Also the performer is largely independent from restricting rules. This general definition [Wikipedia 2009] shall also apply when the performer is the author of an FML document and the activity is modelling and writing a document.

17. **Internationalization**

FML must support multilingual content and identifiers and therefore strictly use the *Unicode* character encoding *UTF-8*. Secure interchange and global encoding potentials will benefit from that stringent document condition. This way *FML* follows the distinct encoding tendency[3] and the general encoding

recommendation of the *W3C*:

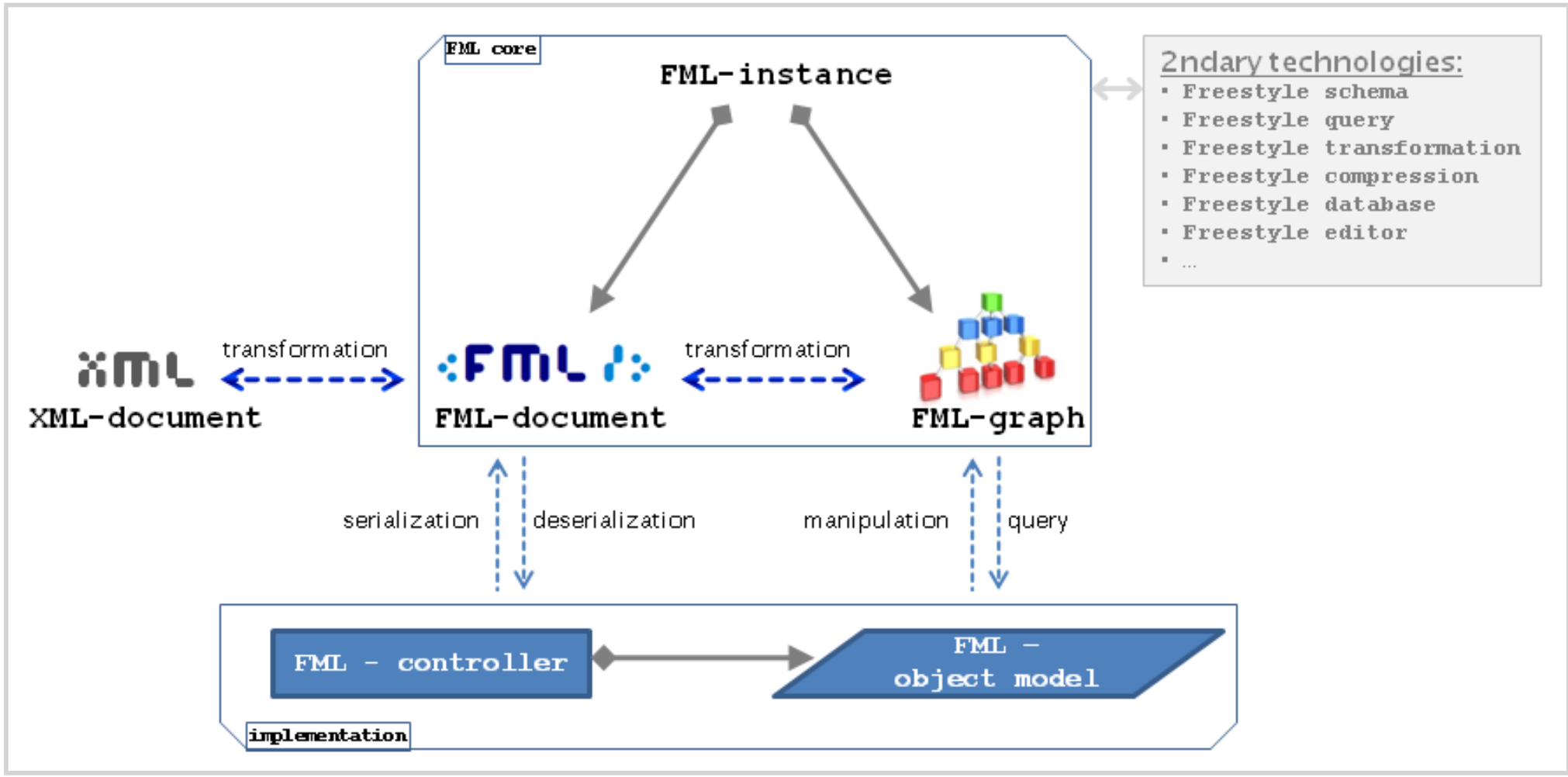
> [. . .]to provide an unambiguous encoding of the content of plain text, ultimately covering all languages in the world, but also major text-based notational systems for science, technology, music, and scholarship.
>
> — Freytag 2007

The secondary writing direction will be inalterable vertical, from top to bottom. The primary writing direction is horizontal, distinguishing dextrograde and sinistrograde writing. This way also semitic languages are supported.

Furthermore *FML* has to provide a stable reference implementation and a concise specification.

Architecture

The technological architecture of FML is outlined, interaction of the function units is illustrated:

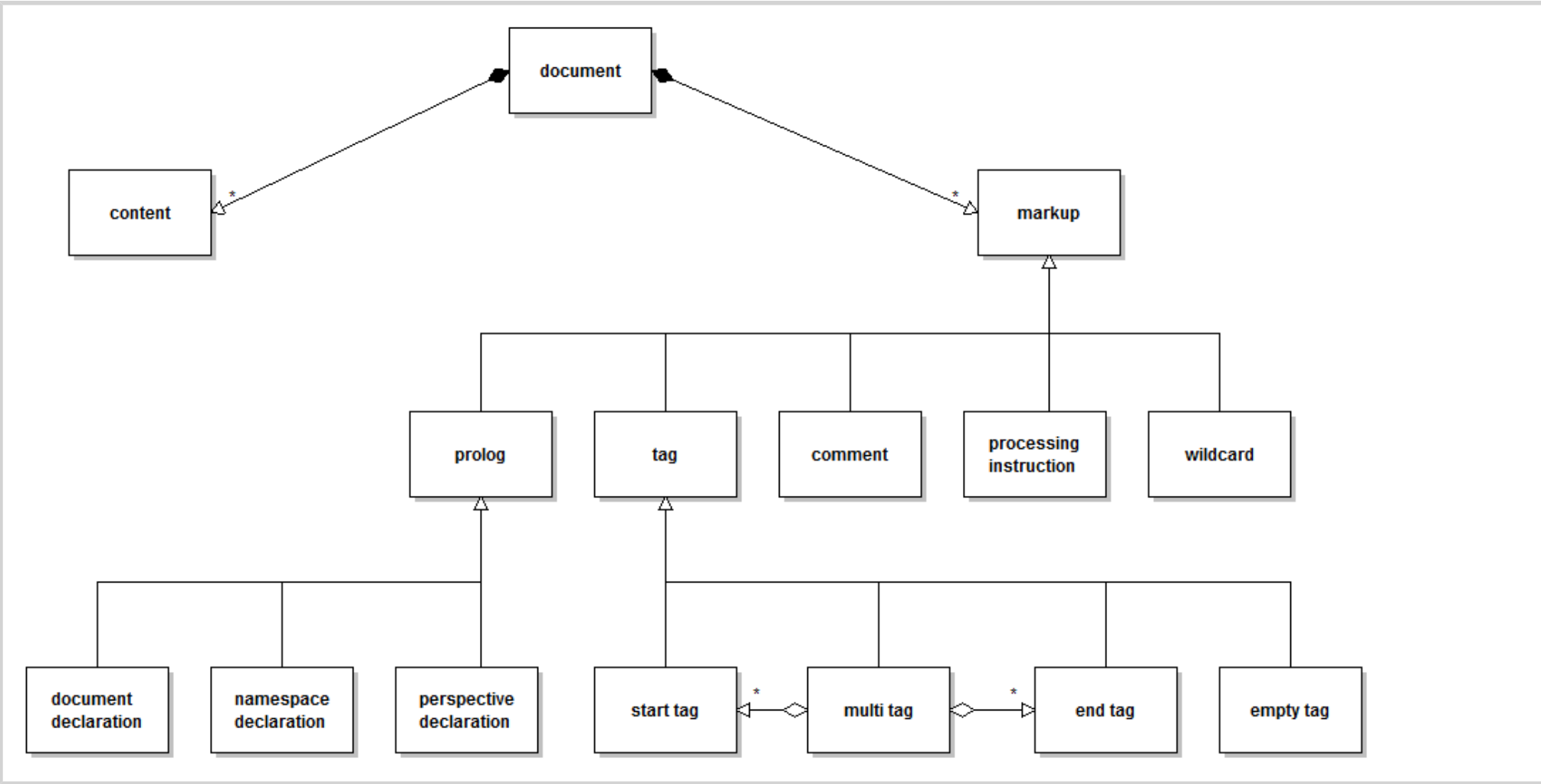


Freestyle Document

> Everything should be made as easy as possible, but not easier.
>
> — Albert Einstein

Components

A compound of content and markup will form an FML document. Each functional unit is called a component. The following graph shows all components and relations within a document:



Document

The component document is composed of an ordered sequence of distinct components of the type

- content and
- markup

in any distribution and of unlimited number.

A document represents the whole FML document. It corresponds with the node of type fml.node.document as the root in an FML graph.

Production rule: fml.document

Content

The component content contains the underlying real content of an FML document before applying any markup: An arbitrary *UTF-8* character sequence.

No special handling of formatting symbols (as in *XML*) applies in the grammar (`trim` may apply at the implementation level): All symbols, also clear spacing, tabulators, wrappings, etc. are integral part of the content sequence. Only the markup initiating symbol < has to be escaped properly: `\<`.

Production rule: fml.content

Markup

Whatever is not content but embedded markup encoding, is markup. The component markup is a generic component and gets specialized by

- prolog,
- tag,
- comment,
- processing instruction,
- wildcard.

All markup-components border from content by the delimiters < and >.

Production rule: fml.prolog, fml.tag, fml.comment, fml.pi, fml.wildcard

Prolog

The prolog of an FML document encapsulates the following three optional declarations:

1. **document-declaration**
   global document informations with the optional attributes
   - `fml.name`: name of the document
   - `fml.uri`: unique *URI* of the document
   - `fml.description`: document description
   - `fml.version`: *FML* version number (`default="1.0"`)
   - `fml.fragment`: fragment identification
   - `fml.schema`: physical *URI* of a validating schema

- `fml.trim="true"|"false"`: parser instruction about handling formatting symbols for FML graph-nodes of type fml.node.content
- `fml.writing-direction="lr"|"rl"`: determination of the writing-direction for external content processors

2. **perspective-declaration**
   global informations about possible perspectives within the document with the optional attributes
   - `fml.perspective.name`: unique name of the perspective (may apply as a prefix for all markup-components)
   - `fml.perspective.uri`: unique *URI* of the perspective
   - `fml.perspective.description`: perspective description
   - `fml.perspective.schema`: physical *URI* of a validating schema (overrides global schema-assignment for this perspective)
3. **namespace-declaration**
   global informations about possible namespaces within the document with the optional attributes
   - `fml.namespace.name`: unique name of the namespace (may apply as a tag-name-prefix)
   - `fml.namespace.uri`: unique *URI* of the namespace
   - `fml.namespace.description`: namespace description

These informations all support readers, authors and external processors in the cognition of document properties for parsing, validating or other processing actions.

Production rule: fml.prolog

Tag

Also in *FML* tags are the primary instrument for content structuring. Four different types of tags exist:

1. **Start Tag**
   The start-tag will begin an annotation and forms an element together with a corresponding end-tag to the right of itself .
2. **End Tag**
   The end-tag will end an annotation and forms an element together with a corresponding start-tag to the left of itself.
3. **Empty Tag**
   The empty-tag self-sufficiently stands in the document content and does not entwine any content.
4. **Multiple Tag**
   The multiple-tag combines several components of type start-tag, end-tag or empty-tag in an irrelevant order.

The start-tag and the empty-tag optionally contain any number of ordered attributes. Each of them contains an unlimited number of ordered values, a data-list. Attribute assignment may as well apply for start-tags and empty-tags in a multiple-tag. The name of an assigned attribute must be unique within one tag. Language inherent attributes are

- `fml.trim`: instruction to a parser about the treatment of formatting symbols during document interpretation
- `fml.segment.id`: `ID` of the segment in which a tag optionally participates (see concept segmentation) (shortcut: tag-suffix $\%_1$)
- `fml.segment.pos`: position within the segment (shortcut: tag-suffix $\%_2$)

Each tag is unambiguously assigned to a namespace. The name of the namespace precedes the name of the tag as a prefix. If there is no namespace assigned, then the

tag is in the default-namespace. Also each tag is unambiguously assigned to a perspective (or default-perspective).

An `ID` may be assigned as a suffix to a start-tag or to an end-tag (see concept identification). In the same perspective p_i, in the same namespace ns_j and for the same tag-name tn_k `ID`s must be unique in the set of start-tags $T_S[p_i][ns_j][tn_k]$ and they must be unique in the set of end-tags $T_E[p_i][ns_j][tn_k]$.

Production rule: fml.tag

Comment

A comment is a simple markup-component for the representation of any documentations. It has absolutely no structuring motive and produces no semantic dependencies. A comment may not explicitly be assigned to a perspective, but is a direct or indirect child of all perspectives.

Production rule: fml.comment

Processing Instruction

*PI*s are targeted by external processors and neither have a structuring motive nor they are of any relevant semantic for the language *FML*. Actually processing instructions are a procedural relict within a declarative markup language. A processing instruction may explicitly be assigned to another perspective than the default-perspective.

A processing instruction encapsulates two literals: `target` for identifying the external processor and `instruction` as a command for that processor.

Production rule: fml.pi

Wildcard

A wildcard is a simple placeholder for later substitution into one component of type tag, comment or processing instruction. Before substitution no structuring motive has to be assumed. A placeholder is just a declaration of intent. A wildcard may explicitly be assigned to another perspective than the default-perspective.

Production rule: fml.wildcard

Grammar

We present the formal grammar of an FML document with 30 production rules (fml-grammar.ebnf):

```
fml.document = fml.prolog? (fml.content | fml.tag | fml.comment | fml.pi | fml.wildcard)* ;
```

```
fml.prolog = fml.prolog.document fml.prolog.perspective* fml.prolog.namespace* ;
```

```
fml.prolog.document = '<@'
          ('fml.name="' fml.attribute.value '"')
          (space 'fml.uri="' fml.attribute.value '"')?
          (space 'fml.description="' fml.attribute.value '"')?
          (space 'fml.version="' fml.attribute.value '"')?
          (space 'fml.fragment="' fml.attribute.value '"')?
          (space 'fml.schema="' fml.attribute.value '"')?
          (space 'fml.trim="' ('true'|'false') '"')?
          (space 'fml.writing-direction="' ('lr'|'rl') '"')?
          '>' linewrap ;
```

```
fml.prolog.perspective = '<@'
```

```
          ('fml.perspective.name="' fml.attribute.value '"')
          (space 'fml.perspective.uri="' fml.attribute.value '"')?
          (space 'fml.perspective.description="' fml.attribute.value '"')?
          (space 'fml.perspective.schema="' fml.attribute.value '"')?
          '>' linewrap ;
```

```
fml.prolog.namespace = '<@'
          ('fml.namespace.name="' fml.attribute.value '"')
          (space 'fml.namespace.uri="' fml.attribute.value '"')?
          (space 'fml.namespace.description="' fml.attribute.value '"')?
          '>' linewrap ;
```

```
fml.content = ( (UTF-8-character - '<') | '\<' )* ;
```

```
fml.tag = '<' (fml.tag.start | fml.tag.end | fml.tag.empty | fml.tag.multiple) '>' ;
```

```
fml.tag.start = (fml.perspective.name '|')? (fml.namespace.name ':')? fml.tag.name (fml.tag.id)?
          (fml.tag.segment)? (space fml.attribute)* ;
```

```
fml.tag.end = (fml.perspective.name '|')? '/' (fml.namespace.name ':')? fml.tag.name (fml.tag.id)? ;
```

```
fml.tag.empty = (fml.perspective.name '|')? (fml.namespace.name ':')? fml.tag.name
          (fml.tag.segment)?  (space fml.attribute)* '/' ;
```

```
fml.tag.multiple = (fml.tag.start | fml.tag.end | fml.tag.empty)
          (fml.tag.start | fml.tag.end | fml.tag.empty)+ ;
```

```
fml.tag.name = fml.name ;
```

```
fml.tag.id = '#' fml.name ;
```

```
fml.tag.segment = '%' fml.segment.id '%' fml.segment.pos ;
```

```
fml.segment.id = fml.name ;
```

```
fml.segment.pos = number ;
```

```
fml.perspective.name = fml.name ;
```

```
fml.namespace.name = fml.name ;
```

```
fml.attribute = fml.attribute.name '="' fml.attribute.value '"' (',"' fml.attribute.value '"')* ;
```

```
fml.attribute.name = fml.name ;
```

```
fml.attribute.value = ( (UTF-8-character - '"') | '\"' )* ;
```

```
fml.comment = '<!' fml.comment.content '!>' ;
```

```
fml.comment.content =  | UTF-8-character |
          (UTF-8-character?
              ( ('!' (UTF-8-character - '>') ) | ( (UTF-8-character - '!') UTF-8-character) )+
          );
```

```
fml.pi = '<?' (fml.perspective.name '|')? fml.pi.target space fml.pi.instruction '>' ;
```

```
fml.pi.target = fml.name ;
```

```
fml.pi.instruction = ( (UTF-8-character - '>') | '\>' )+ ;
```

```
fml.wildcard = '<' (fml.perspective.name '|')? '>' ;
```

```
fml.name = ( (UTF-8-character - escape-symbols.exclude) | escape-symbols.include )+ ;
```

```
UTF-8-character = [U+0000 - U+FFFF] ;
```

```
space = U+0020 ;
```

```
linewrap = U+000A ;
```

```
number = ( '0' | '1', '2', | '3' | '4' | '5' | '6' | '7' | '8' | '9' )+ ;
```

```
escape-symbols.exclude = ( '>', '<', '\', '@', '?', '!', '/', '|', ':', '#', '%', space ) ;
```

```
escape-symbols.include = ( '\>' | '\<', '\\', | '\@' | '\?' | '\!' |
          '\/' | '\|' | '\:' | '\#' | '\%' | '\' space ) ;
```

The grammar displays that there are not many constraints regarding character usage for names and identifiers. Actually, pretty much cryptic and delicate literals are constructable. But *FML* takes up the position of maximum freedom and justifiable demands minimal restrictions.

One additional rule applies: Names and identifiers must not begin with 'fml.'. This literal is generally reserved for language inherent constructs. A lexical parser has to check this constraint within postprocessing.

Validity

Three states of validity will be distinguished:

- *wellformed*
  A *FML*-document conforming strictly to the grammar of the language definition (see section “Grammar”) will be in the state wellformed, and thus, may be transformed (see transformation) into a corresponding *FML*-graph (see FML graph).
- *nested*
  If an *FML*-document is wellformed and its corresponding *FML*-graph has exactly one root-element-node and none of the other element-nodes have more than one parent (monohierarchical structure) it will be in the state nested.

- *valid*
  If a *FML*-document *D* is wellformed and in accordance with any schema definition (see section “Future Work”) *S*, then *D* is said to be *valid* against *S*.

Valid implies wellformed but not necessarily nested. The state nested always implies wellformed.

The validity status nested responds to the question, whether the new monohierarchy-suspending *freestyle* features Interference (see concept interference), Congruence (see concept congruence), Independence (see concept independence), or Segmentation (see concept segmentation) have been used or whether structural *XML*-compatibility exists.

Freestyle Graph

Due to the requirement graph representation, for each wellformed FML document exists a corresponding FML graph for document interpretation, visualization and model transformation.

Now the graph gets defined, graph-theoretic characteristics are outlined.

Vertices

All nodes of the polyhierarchical FML graph are defined: Type, representation form, corresponding components of an FML document, possible children and parents, and encapsulated informations.

Document

Virtual root-node of the polyhierarchical FML graph. This node represents the whole FML document and encapsulates all global prolog-declarations.

Display[4]: Trapezoid; `#07F9FC` ; `#03022F` ; Possessa [Guitton 2010] | Courier New; fml.name | 'FML'

Components[5]: document, prolog

Parents: none

Children[6]: fml.node.perspective+

Informations:

- `fml.node-type = 'fml.node.document'`
- `fml.name?`
- `fml.uri?`

fml.description?

- fml.version?
- fml.fragment?
- fml.schema?
- fml.trim? = 'true' | 'false'
- fml.writing-direction? = 'lr' | 'rl'
- (fml.perspective.name, fml.perspective.uri, fml.perspective.description, fml.perspective.schema)*
- (fml.namespace.name, fml.namespace.uri, fml.namespace.description)*

Perspective

Virtual node underneath the document-node for the subordination of all components that belong to a particular perspective. Not the perspectives declared in the prolog apply, only the perspectives actually in use. A special case is the default -perspective, that applies for all components that are not assigned to any perspective (also if the optional perspective-concept is not in use at all). No content may get excluded: The nodes of the type fml.node.perspective include all nodes of the type fml.node.content as their direct or indirect children.

Display: Trapezoid; #000000 ; #FFFFFF ; Courier New; fml.perspective.name | 'fml.default'

Components: document

Parents: fml.node.document

Children:

- fml.node.content *
- fml.node.element *
- fml.node.comment *
- fml.node.pi *
- fml.node.wildcard *

fml.node.perspective+

Informations:

- fml.node-type = 'fml.node.perspective'
- fml.perspective.name?

Content

The *UTF-8* character-sequence between two markup-components forms a node of type fml.node.content. The content may also be empty ( ) as an insert position for further editing, or filled only with formatting characters (~). So these nodes represent the real content and are always leafs in an FML graph.

Display: Round Rectangle; `#000000` ; `#FFFFFF` ; Arial; fml.content | ' ' | '~'

Components: content

Parents:

- fml.node.perspective *
- fml.node.element *
- fml.node.attribute ?

Children: none

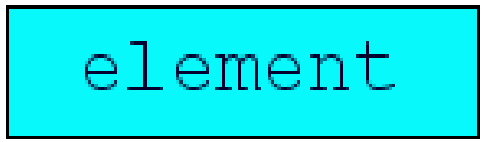
Informations:

- `fml.node-type = 'fml.node.content'`
- `fml.content`

Element

Elements are the primary instrument for structuring a polyhierarchical FML graph. Two corresponding tags will form a node of type fml.node.element. This node directly or indirectly subordinates exactly to one perspective, may be a child of several elements, and may itself be parent of other elements. Elements content may be enriched by attributes.

Display: Rectangle; `#03022F` ; `#07F9FC` ; Courier New; fml.element.name



Components: tag

Parents:

fml.node.perspective ?

- fml.node.element *

Children:

- fml.node.content *
- fml.node.element *
- fml.node.attribute *
- fml.node.comment *
- fml.node.pi *
- fml.node.wildcard *

Informations:

- `fml.node-type = 'fml.node.element'`
- `fml.element.namespace.name?`
- `fml.element.name`
- `fml.element.id?`
- `fml.element.autocompleted? = 'start-tag' | 'end-tag'`
- `fml.element.trim? = 'true' | 'false'`

Attribute

As a child an attribute enriches a node of type fml.node.element. It contains an ordered list of content.

Display: Rectangle; `#03022F` ; `#FFFFFF` ; Courier New; fml.attribute.name

Components: tag

Parents: fml.node.element

Children: fml.node.content+

Informations:

- `fml.node-type = 'fml.node.attribute'`
- `fml.attribute.name`

Comment

Comments represent any documentation an have no motive to structure anything.

Display: Hexagon; `#000000` ; `#C8FFCC` ; Arial; fml.comment.content

Components: comment

Parents:

- fml.node.perspective `?`
- fml.node.element `*`

Children: none

Informations:

- `fml.node-type = 'fml.node.comment'`
- `fml.comment.content`

Processing Instruction

Processing instructions are addressed by external document processors and have no structuring motive.

Display: Parallelogram; `#05ED04` ; `#000000` ; Courier New; fml.pi.target

Components: processing instruction

Parents:

- fml.node.perspective `?`
- fml.node.element `*`

Children: none

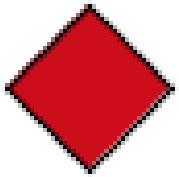
Informations:

- `fml.node-type = 'fml.node.pi'`
- `fml.pi.target`
- `fml.pi.instruction`

Wildcard

Wildcards are placeholders for a later substitution into a node of type fml.node.element, fml.node.comment or fml.node.pi.

Display: Diamond; - ; `#CC0E1C` ; - ; -



Components: wildcard

Parents:

- fml.node.perspective `?`
- fml.node.element `*`

Children: none

Informations: fml.node-type = 'fml.node.wildcard'

Edges

The edge-set exclusively consists of directed edges between two nodes that are arranged in a hierarchical relation: $Parent \xrightarrow{n} Child$. Except of fml.node.document each node possesses at least one incoming edge. fml.node.content, fml.node.comment, fml.node.pi and fml.node.wildcard will never have outgoing edges.

An FML graph will have one node of type fml.node.document, the other nodes may occur unlimited. An empty document will be represented by exactly one node of type fml.node.document.

Each edge is labeled with $n \in \mathbb{N}$. For all outgoing edges of a node, numbering starts with `0` and increments without any gaps. That value indicates the position of the corresponding components within the linear sequence of an FML document.

Parent-child-relations between node-types are possible with directed edges and the following cardinalities[7]:

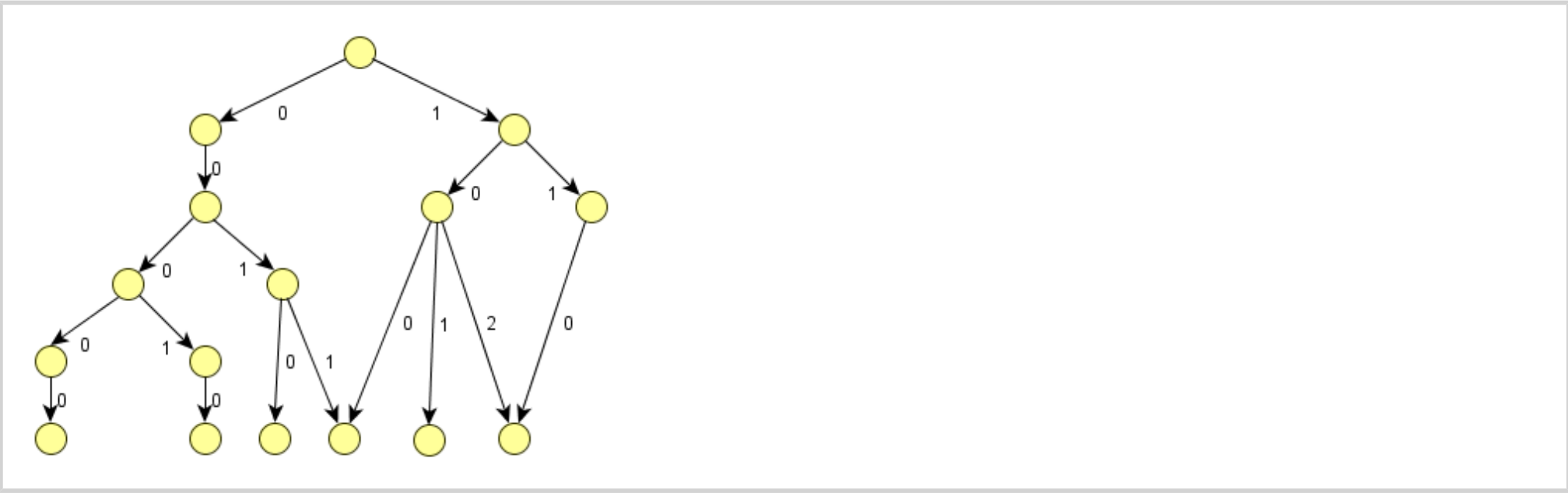
Characteristics

An FML graph has the following characteristics:

- **acyclic**
  No loop road exists.
- **directed**
  Each edge represents a parent→child relation.
- **simple**
  At most one edge between two different vertices.
- **connected**
  A least one path exists from the root node to any other node.
- **edge weighted**
  Edges are labeled with numbers.
- **attributed nodes**
  Nodes encapsulate attribute-value pairs.
- **root**
  There is exactly on node with children and without any parents.

- **leafs**
  Nodes without children exist.
- **path restriction**
  A direct and an indirect path between a parent and a child may never exist simultaneously. Redundant paths between two nodes may exist, but exclude direct connections.
- **polyhierarchical**
  Nodes may have multiple parents. Each node (except root-node) can be reached by at least one directed path from the root. Polyhierarchies are self-similar: Each subgraph is as well a polyhierarchy.

The following graph exemplifies a polyhierarchy and is in accordance with the graph-theoretic characteristics:



With *GXL* , graphs and graph-schemata can be described and exchanged. At http://www.freestyle-markup.org/gxl/fml-graph.schema.gxl a *GXL* -schema is presented, that formally describes FML graphs and may validate particular graph-instances.

Freestyle Concepts

There are 9 fundamental *FML*-concepts, that consider all requirements and use the features of FML documents and FML graphs. These concepts mainly arise from new possibilities in an unrestricted polyhierarchical environment.

Annotation

According to the two general characteristics for markup ("[. . .] part of the text, yet distinguishable from it." Raymond 1993), annotations are embedded in the content and separable from it. The position of markup-components (tag, comment, processing instruction, wildcard) in the content-sequence is very impotant: Changing it leads to a

totally different document interpretation, a different FML graph. Markup embedding is based upon simple rules:

1. Markup section start is indicated by the familiar symbol "<".
2. Markup section end is indicated by the familiar symbol ">".
3. If occurring in content, the markup-symbols "<" and ">" (as well as "\") have to be escaped properly ("\<", "\>", "\\") [8] .
4. The prolog-components are optional but have to be well ordered and the first annotations in the document.
5. The other markup-components tag, comment, processing instruction and wildcard may be inserted unrestricted into the content.

The following example illustrates the use of all markup-components (excl. prolog) in an wellformed FML document and the use of all nodes types in a corresponding FML graph.

FML content:

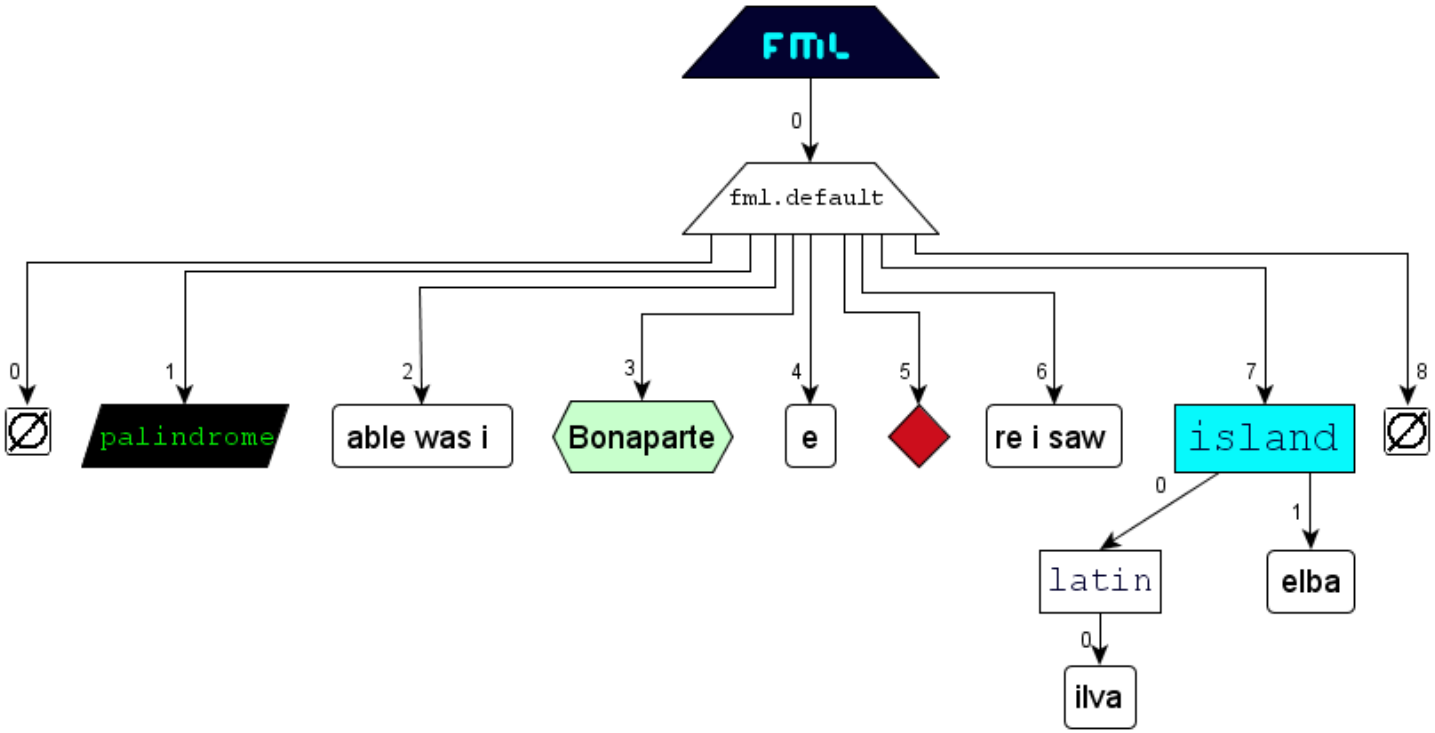
able was i ere i saw elba

FML document:

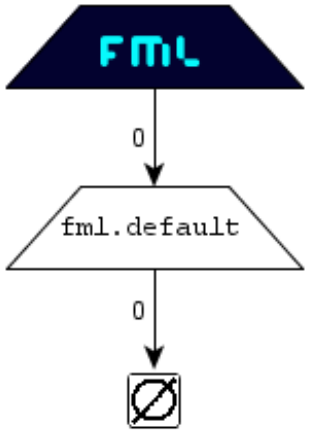
```
<?palindrome start>
able was i
<!Bonaparte!>
e
<>
re i saw
<island latin="ilva">
    elba
</island>
```

annotation.fml

FML graph:



The following FML graph represents the *empty* FML document:



Declaration

Informations about the document itself, perspectives and namespaces may be declared at the very beginning of an FML document. These declarations are optional and a support for authors, editors and external processors for the evaluation of the document. The attributes of each declaration are self-describing and relevant for all use cases of a declarative markup language. For document interpretation all declared informations will be lossless transformed into the attributed root of an FML graph, a node of the type fml.node.document.

The following example contains all of the optional declarations with all possible attributes.

FML document:

```
<@fml.name="important institutions" fml.uri="http://www.freestyle-markup.org"
fml.description="institutions relevant to FML" fml.version="1.0" fml.fragment="f1"
fml.schema="temp.fsd" fml.trim="true" fml.writing-direction="lr">

<@fml.perspective.name="org"
 fml.perspective.uri="http://de.wikipedia.org/wiki/Organisation"
fml.perspective.description="organisations" fml.perspective.schema="org.fsd">

<@fml.perspective.name="geo"
 fml.perspective.uri="http://de.wikipedia.org/wiki/Geographie"
fml.perspective.description="geographic inf." fml.perspective.schema="geo.fsd">

<@fml.namespace.name="iso31661"
 fml.namespace.uri="http://de.wikipedia.org/wiki/ISO_3166"
 fml.namespace.description="iso country codes">

<org|university>Universität Bremen<org|/university>,
<geo|iso31661:DE>Bremen<geo|/iso31661:DE>

<org|institute>Institut für Deutsche Sprache<org|/institute>,
<geo|iso31661:DE>Mannheim<geo|/iso31661:DE>

<org|conference>Balisage Conference<org|/conference>,
<geo|iso31661:CA>Montréal<geo|/iso31661:CA>
```

declaration.fml

Tagging

For *FML* the primary instrument for structuring document content are tags: empty-tags and start-tags with corresponding end-tags. A corresponding tag-pair forms an element in an FML graph interpretation and subordinates the data sequence in between. Tagging and interpretation is handled about the same as in *XML*/*SGML*, except that there are less restrictions and more features.

A root-element is possible, but not necessary. Isolated tags (tags with a missing corresponding start- or end-tag) are alright: Missing start or end is assumed beyond document borders (see concept fragmentation).

FML content[9] :

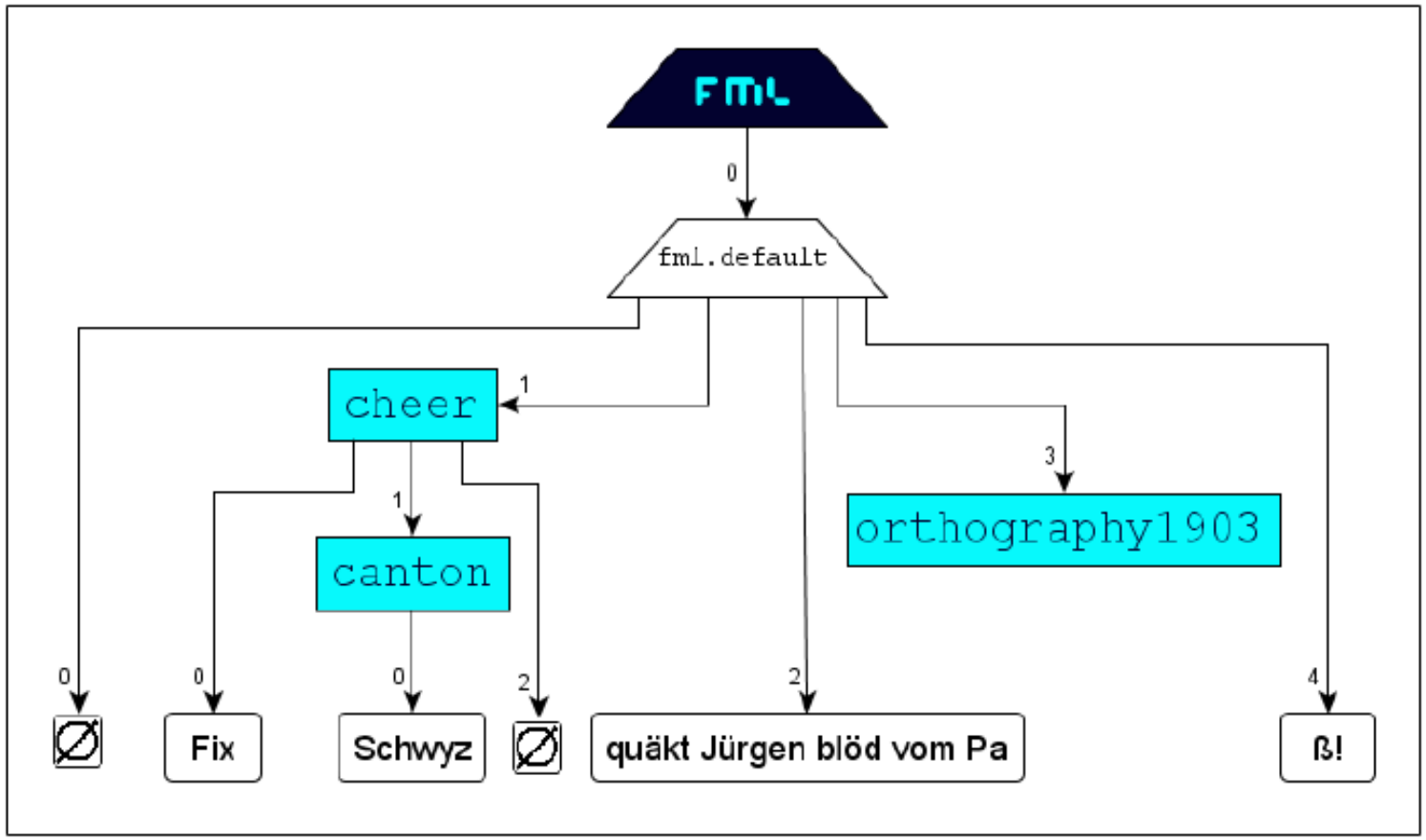
Fix Schwyz quäkt Jürgen blöd vom Paß!

FML document:

```
<cheer>
Fix
        <canton>
                Schwyz
        </canton>
</cheer>
quäkt Jürgen blöd vom Pa
<orthography1903/>
ß!
```

tagging.fml

FML graph:



Attribution

Attributes refine tags and enrich them with additional informations. This specialization is realized by declaring an ordered list of attributes within start-tags or empty-tags in an FML document. Within one tag each attribute name must be unique. Therefore elements may possess an unlimited number of children, nodes of type fml.node.attribute. Each attribute again contains a non-empty ordered value-list.

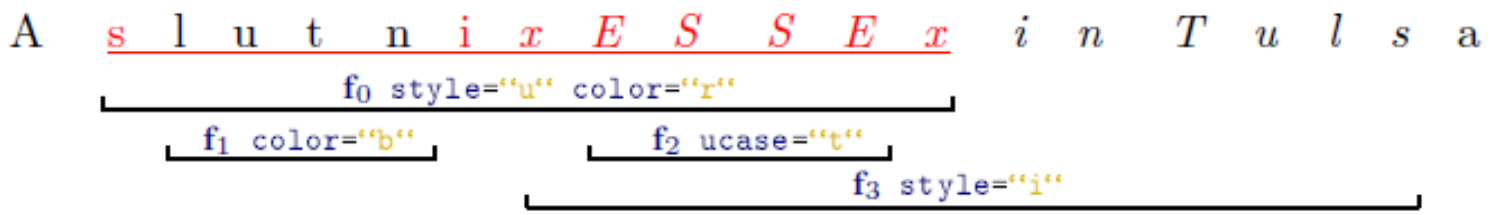
According to the object-oriented inheritance paradigm *FML* offers a mechanism for passing on attributes from parents to children: An element e_1 will inherit from element e_0, if e_1 is a direct or indirect child of e_0. Neither this concept has syntactical implications for an FML document nor it influences the transformation into an FML graph. Rather the inheritance concept reflects on the level of implementation: With the method `Attribut[] Element.getAttributes(boolean enableInheritance)` attributes are queried with or without inheritance, depending on the parameter `enableInheritance`.

The following scenario illustrates concept and usage of inheritance.

FML content: Markup of the content

A slut nixes sex in Tulsa

by the typographic properties **bold**, *italic*, black, red, and UPPERCASE is relized by the general formatting-tag `f*` and the specializing attributes `style`, `color` and `ucase`. Four markups along these properties will result in the following document structure:

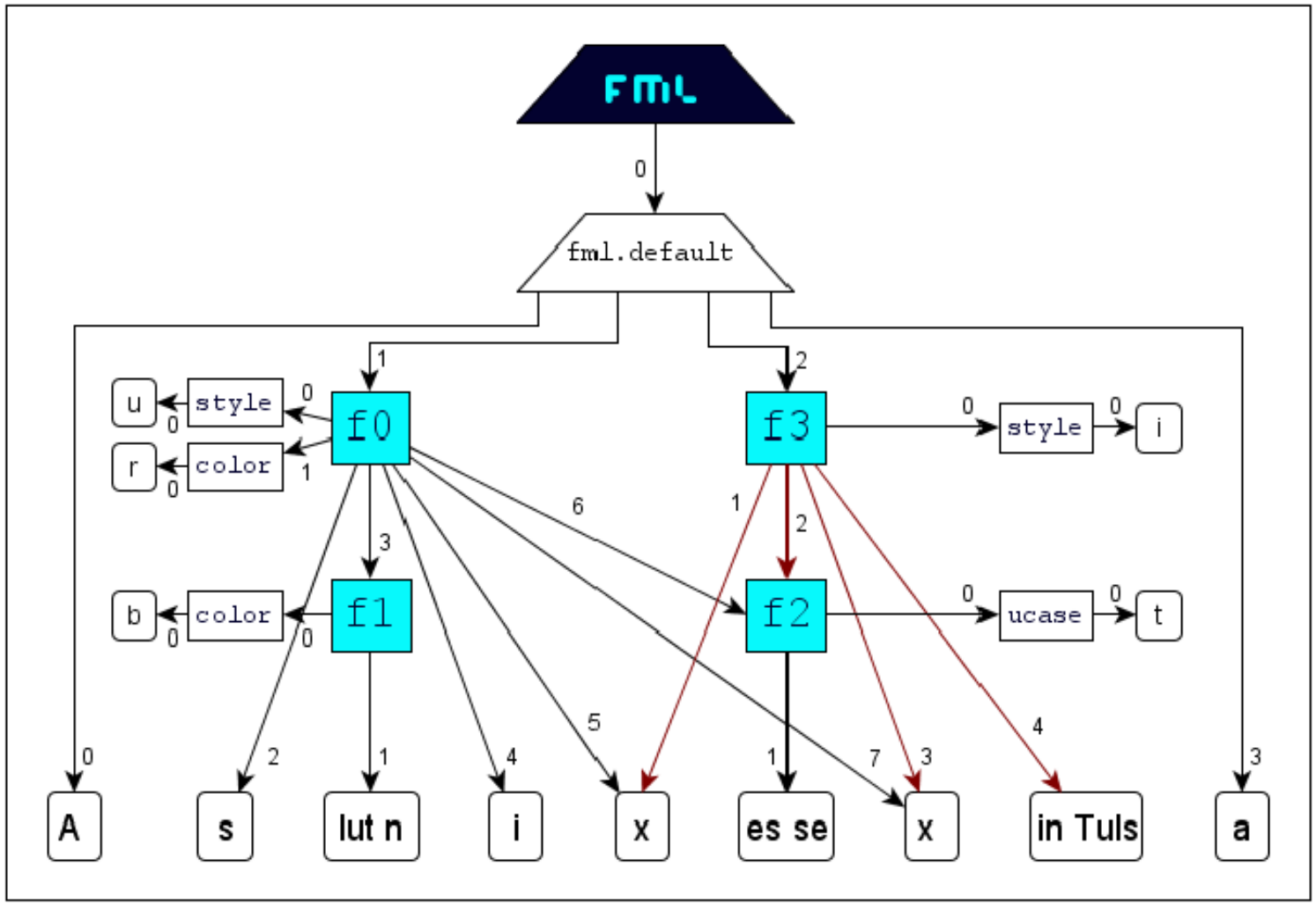


FML document:

```
A
<f0 style="u" color="r">
        s
        <f1 color="b">lut n</f1>
        i
<f3 style="i">
        x
        <f2 ucase="t">es se</f2>
        x
</f0>
        in Tuls
</f3>
a
```

attribution.fml

FML graph:



Implementation:

```
f0.getAttributes(false): style="u", color="r"
f0.getAttributes(true):  style="u", color="r"

f1.getAttributes(false): color="b"
f1.getAttributes(true):  color="b", style="u"

f2.getAttributes(false): ucase="t"
f2.getAttributes(true):  ucase="t", style="i","u", color="r"

f3.getAttributes(false): style="i"
f3.getAttributes(true):  style="i"
```

Interference

The markup of interfering structures is not subject to any restrictions and demands no special attention. Start- and endpositions of semantic segments are marked up separately and independently. Resulting interferences are absolutely permitted. A content-segment that is located in the markup-area of `n` corresponding tag-pairs will directly or indirectly be a child of `n` elements in an FML graph.

The following scenario illustrates the handling of overlapping. For clearness, the semantic markup segments are typographic ones.

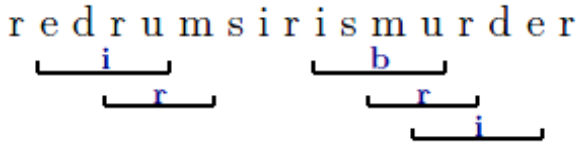
FML content: Marking up the content

redrumsirismurder

with the typographic properties *italic* (`i`), **bold** (`b`) and red (`r`)

redrumsirismurder

will result in the following document structure:

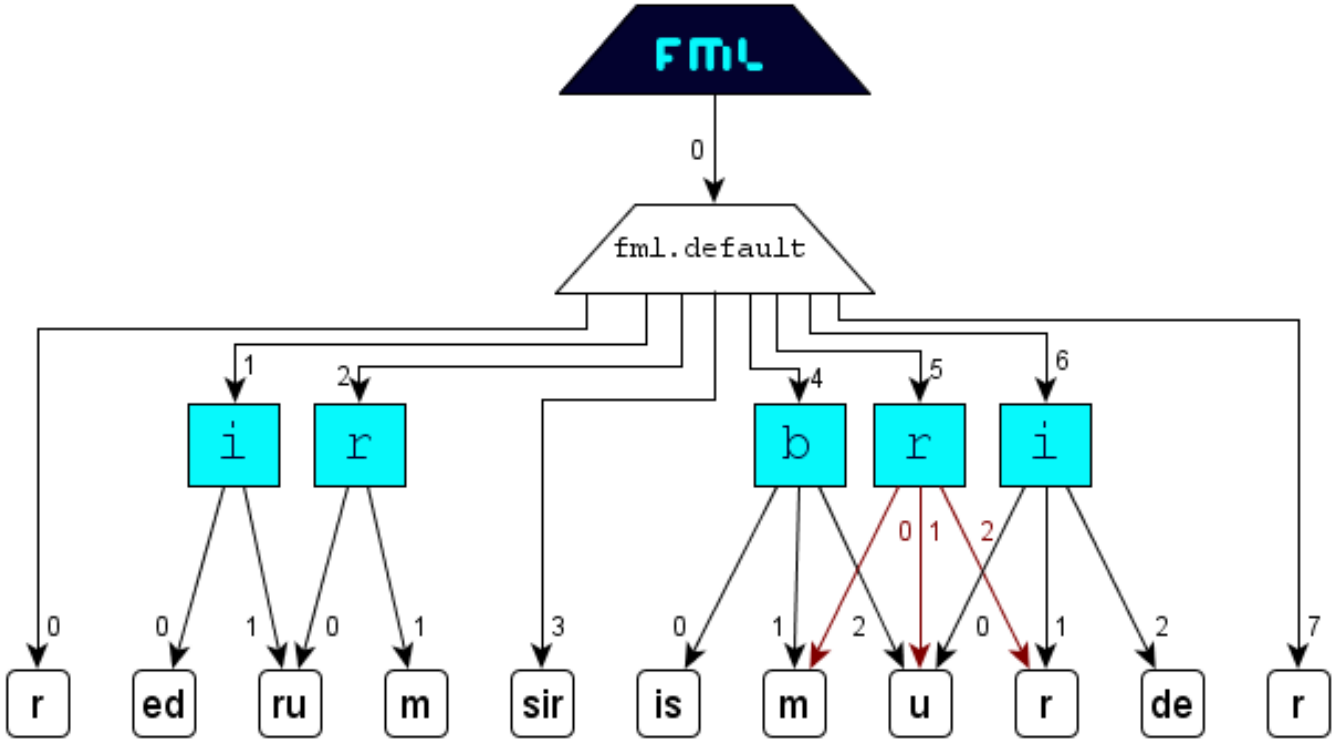


FML document:

```
r <i> e d <r> r u </i> m </r> s i r <b> i s <r> m <i> u </b> r </r> d e </i> r
```

interference.fml

FML graph:



Identification

The concept *Identification* serves for the secure assignment of corresponding tags. Generally a start-tag and an end-tag will correspond, if they exhibit the same name in the same perspective and in the same namespace. But if a document contains several start-tags of the same name or several end-tags of the same name, then the identification of corresponding tags will be ambivalent for element-construction. This interpretation uncertainty will be dissolved with the concept identification: The assignment (applying tag-name-suffix `#ID`$_i$) of an unique identification key `ID`$_i$ to both parts of a tag-pair will weld them and assure correspondency.

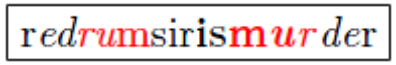
This concept is convenient for scenarios where complex structures with interferring markup occur and where the available tag-set has a high degree of semantic overload.

The following typographic example illustrates the use of the concept identification:

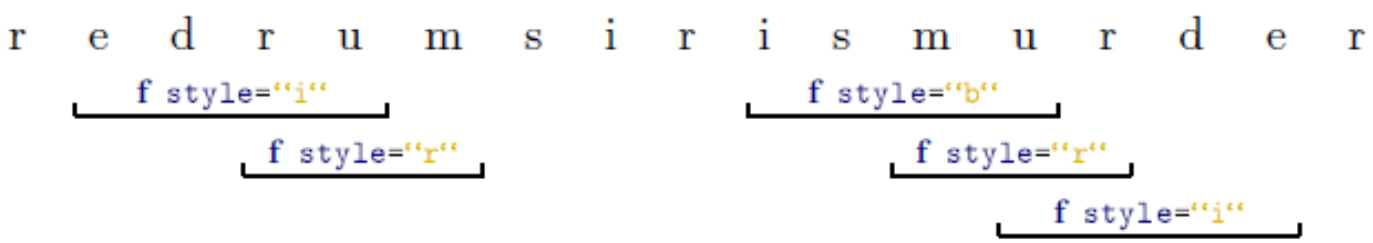
FML content: For the markup of the content

```
redrumsirismurder
```

along the typographic properties *italic*, **bold** and red



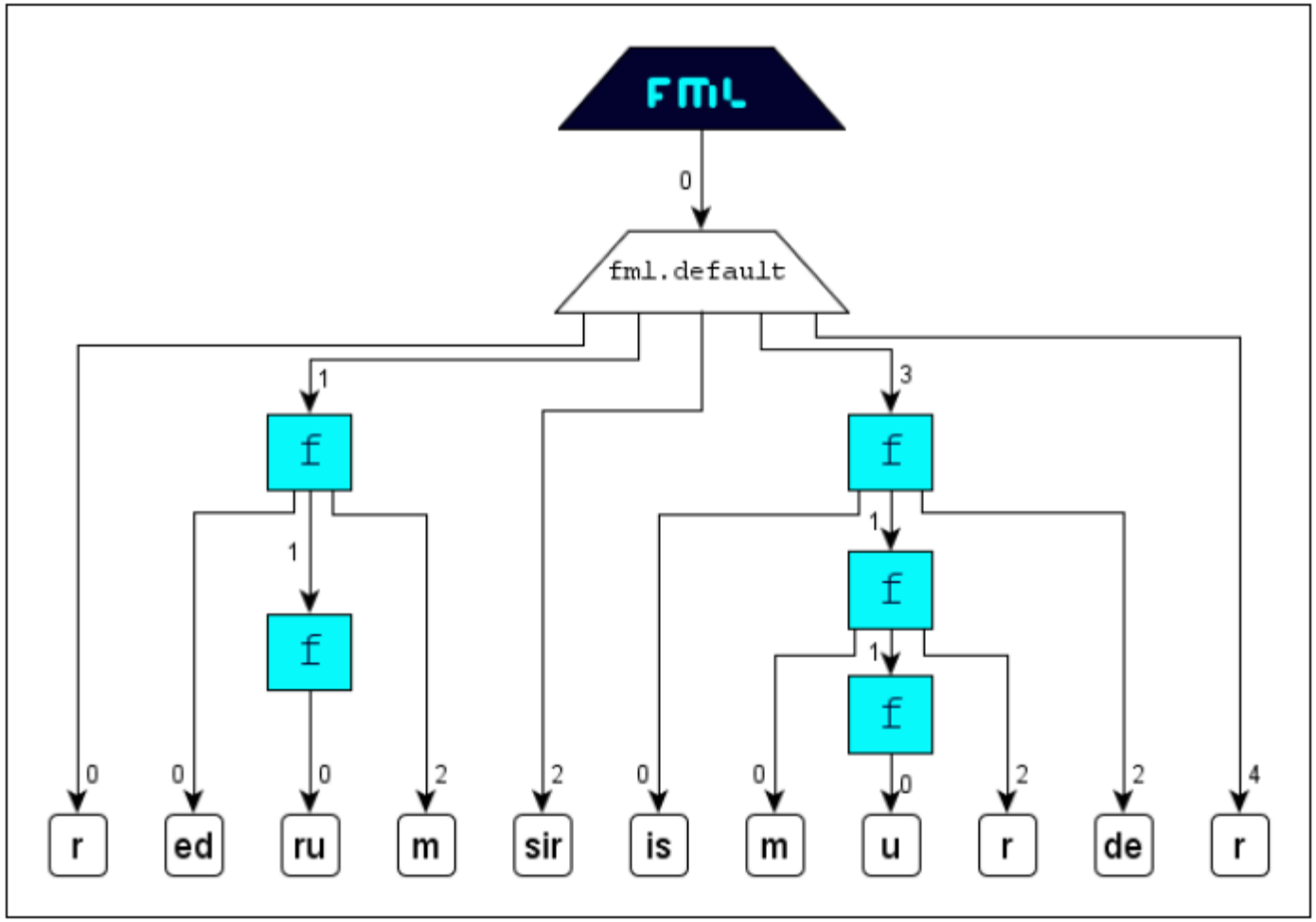
only one formatting element (tag-name `f`) is available. The specialization of `f` gets effected with the attribute `style` and its value-range `i` for *italic*, `b` for **bold** and `r` for red. The resulting document structure is the following:



If we now markup the interfering structures without applying the concept identification

```
r <f style="i"> e d <f style="r"> r u </f> m </f> s i r <f style="b">
i s <f style="r"> m <f style="i"> u </f> r </f> d e </f> r
```

then corresponding tags are defined from outside to inside (according to the *properly-nested* principle of *XML*). Thus that FML document would get interpreted in an FML graph as follows:



This *default*-interpretation of course does not reflect the real scenario, the real document structure. The question "Which `<f>` forms with which `</f>` an element?" must be determinable (actually `2! + 3! = 8` element arrangements are possible).

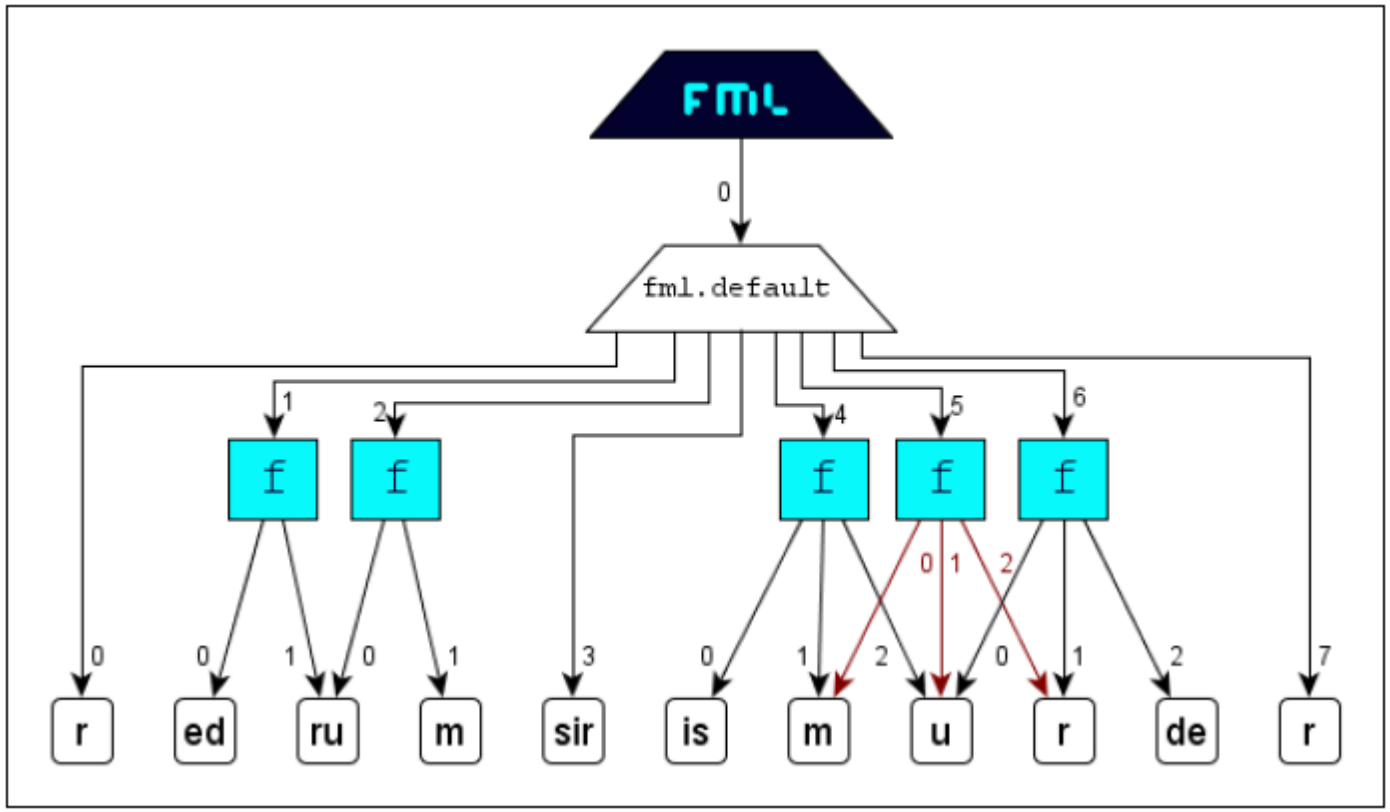
The right result can be achieved applying the simple and intuitive concept identification:

FML document:

```
r <f #i1 style="i"> e d <f #r1 style="r"> r u </f #i1> m </f #r1>
s i r <f #b1 style="b"> i s <f #r2 style="r"> m <f #i2 style="i">
u </f #b1> r </f #r2> d e </f #i2> r
```

identification.fml

FML graph:



Congruence

The markup of congruent structures gets realized using indecomposable multiple-tags. A multiple-tag unites any number of start-tags, end-tags and empty-tags in an irrelevant order, not allowing any content in between.

The concept congruence ensures structural security and consistency. Placing tags one after another in a sequence results in unintended hierarchical relations and the feasibility to break congruence by inserting content between the tags.

The following scenario illustrates the markup of congruent structures:

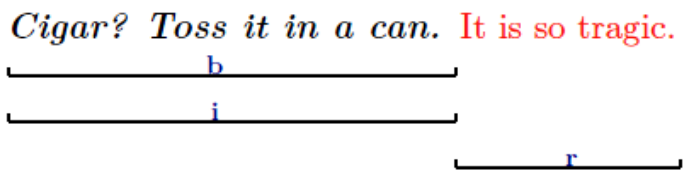
FML content: Marking up the content

Cigar? Toss it in a can. It is so tragic.

with the typographic properties *italic* (`i`), **bold** (`b`) and red (`r`)

Cigar? Toss it in a can. It is so tragic.

will result in the following document structure:

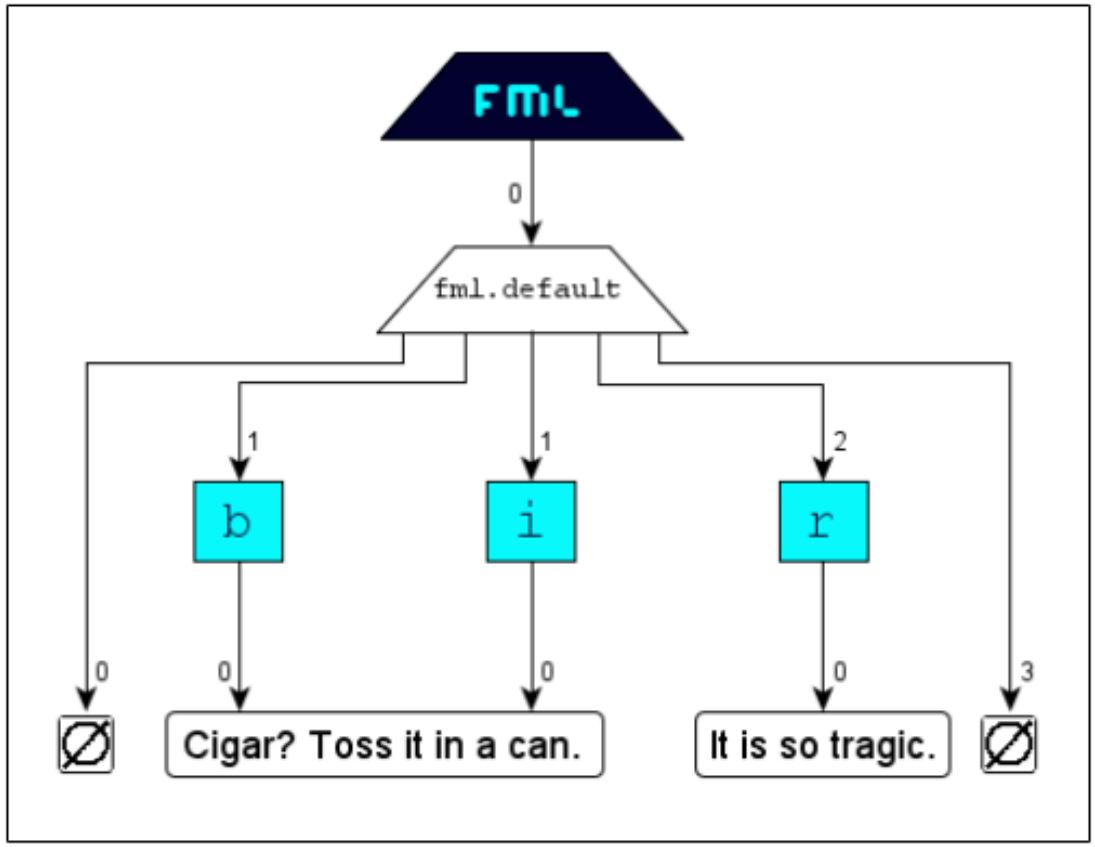


FML document:

```
<b i>Cigar? Toss it in a can. </b /i r>It is so tragic.</r>
```

congruence.fml

FML graph:



Independence

FML admits a consolidated markup of heterogeneous perspectives into one redundance-free document. One perspective represents an identified annotation layer. The components tag, processing instruction and wildcard may optionally be assigned to exactly one perspective. The assignment to a perspective P_i with the unique name P_i`name` will be effected by inserting the prefix P_i`name|` right after the markup delimiter <. This assignment concept applies for components, not for attributes or for each tag in a multiple-tag.

All of the perspectives in use will share the documents content. Components that are not explicit assigned to a perspective will be subordinated to the default-perspective. Thus each FML document contains at least one perspective.

Perspectives may optionally be declared in the prolog. Neither declared perspectives have to be in use in the document nor the perspectives in use in the document have to be declared in the prolog. For a document interpretation within an FML graph only the perspectives that are really in use will be transformed to a node of type fml.node.perspective. During transformation optionally any perspectives may get included or excluded.

The following example illustrates the application of the concept independence with two indepentent annotation layers structuring the same content:

FML content: Marking up the content

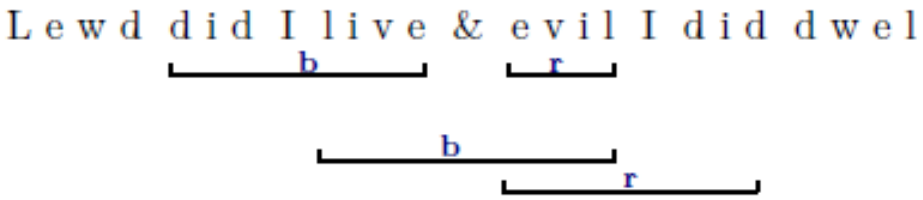
Lewd did I live & evil I did dwel

with the typographic properties **bold** and red is accomplished differently by two Illustrators I_1 and I_2

I_1: Lewd **did I live** & evil I did dwel

I_2: Lewd did I **live & evil** I did dwel

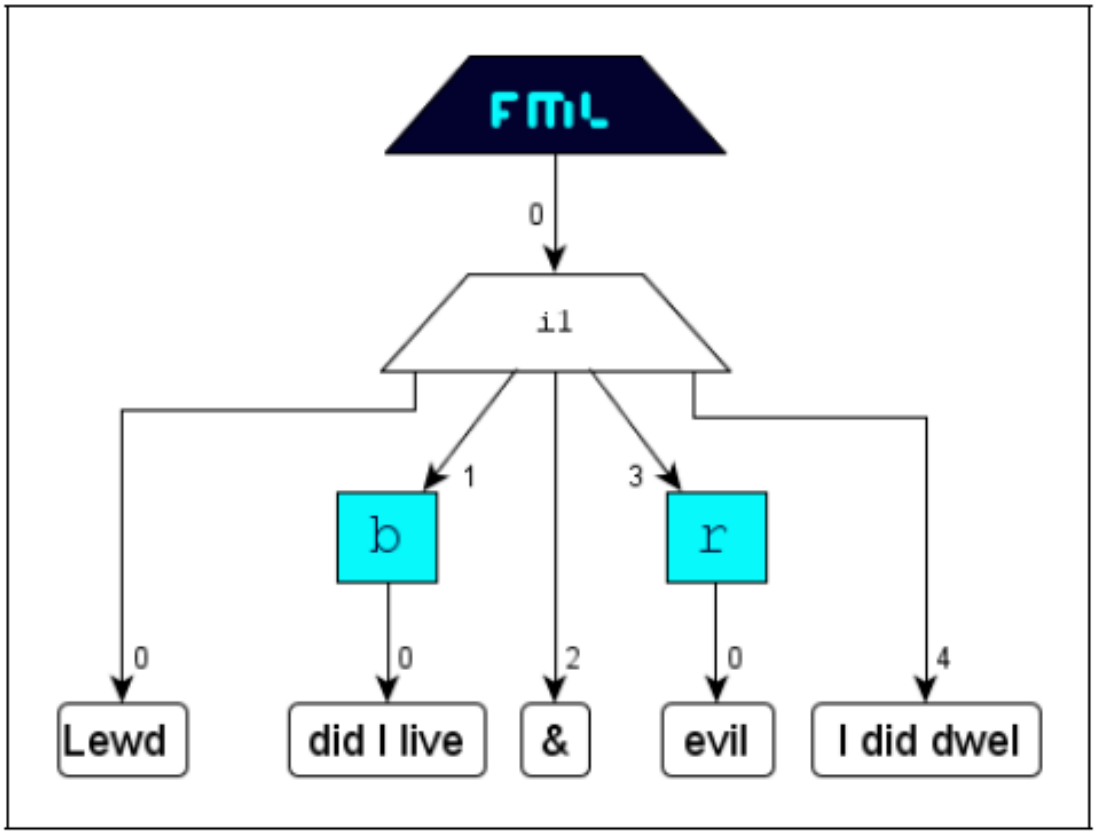
and results in the following document structures (**bold**=b, red=r):



FML document I_1:

```
Lewd <i1|b>did I live<i1|/b> & <i1|r>evil<i1|/r> I did dwel
```

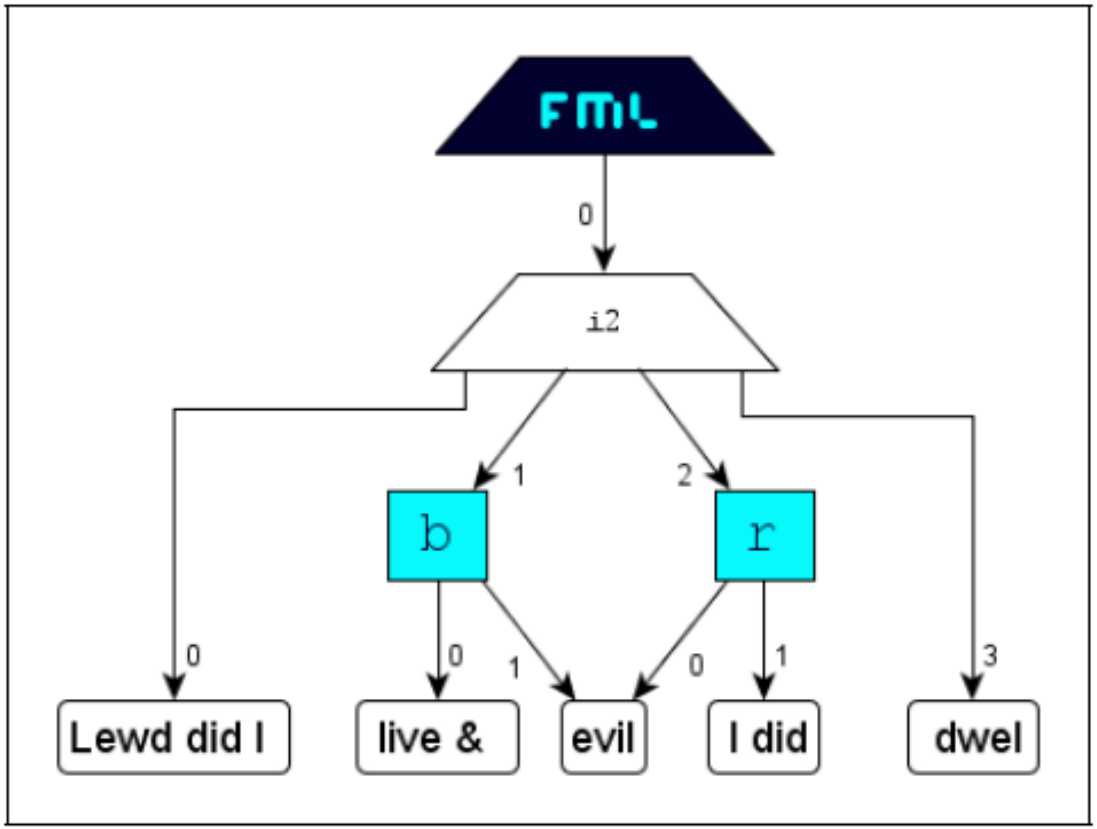
FML graph I_1:



FML document I_2:

```
Lewd did I <i2|b>live & <i2|r>evil<i2|/b> I did<i2|/r> dwel
```

FML graph I_2:

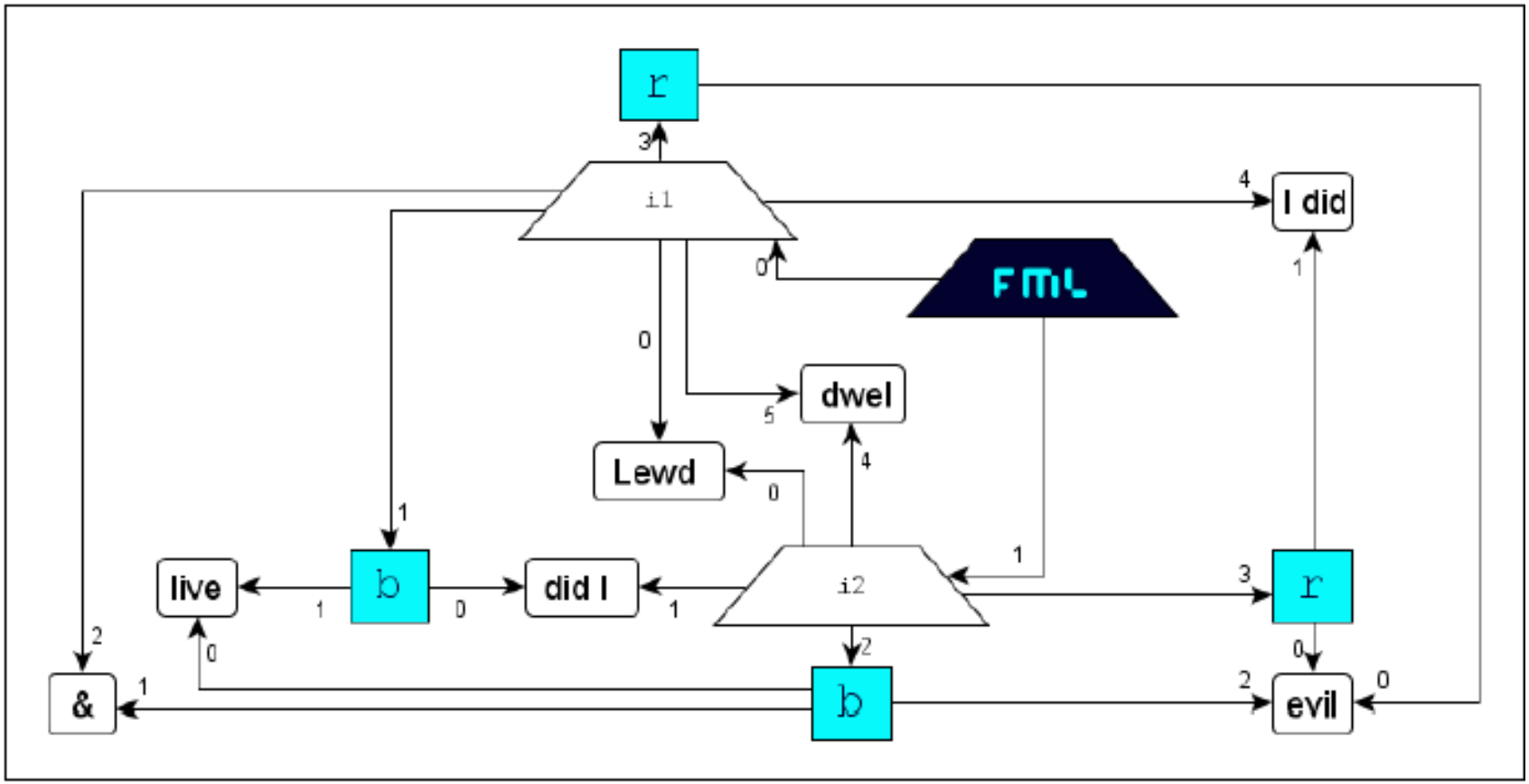


FML document I_1 + I_2:

```
Lewd <i1|b>did I <i2|b>live<i1|/b> & <i2|r><i1|r>evil
<i1|/r><i2|/b> I did<i2|/r> dwel
```

independence.fml

FML graph I_1 + I_2:



Segmentation

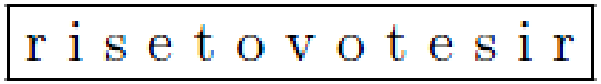
The powerful concept segmentation offers the composition of new content based on segments of the existing content. A virtual element encapsulates the content of participating tags and will be subordinated to the corresponding perspective-node (fml.node.perspective) in an FML graph as a direct child with the constant position `-1`. Since beside position and run length also quantity and order of involved segments is arbitrary via the language inherent tag-attributes `fml.segment.id` and `fml.segment.pos`, boundless composition possibilities arise from the charachter pool of the linear content sequence of a document.

An intensive use of this concept will decrease readability, but for pure mechanical operating fulminant potentials arise. For instance the treasury of words of a language could without redundancy be entirely marked up based on only the alphabet characters of that language.

In the following example

```
e v e
```

gets assembled from

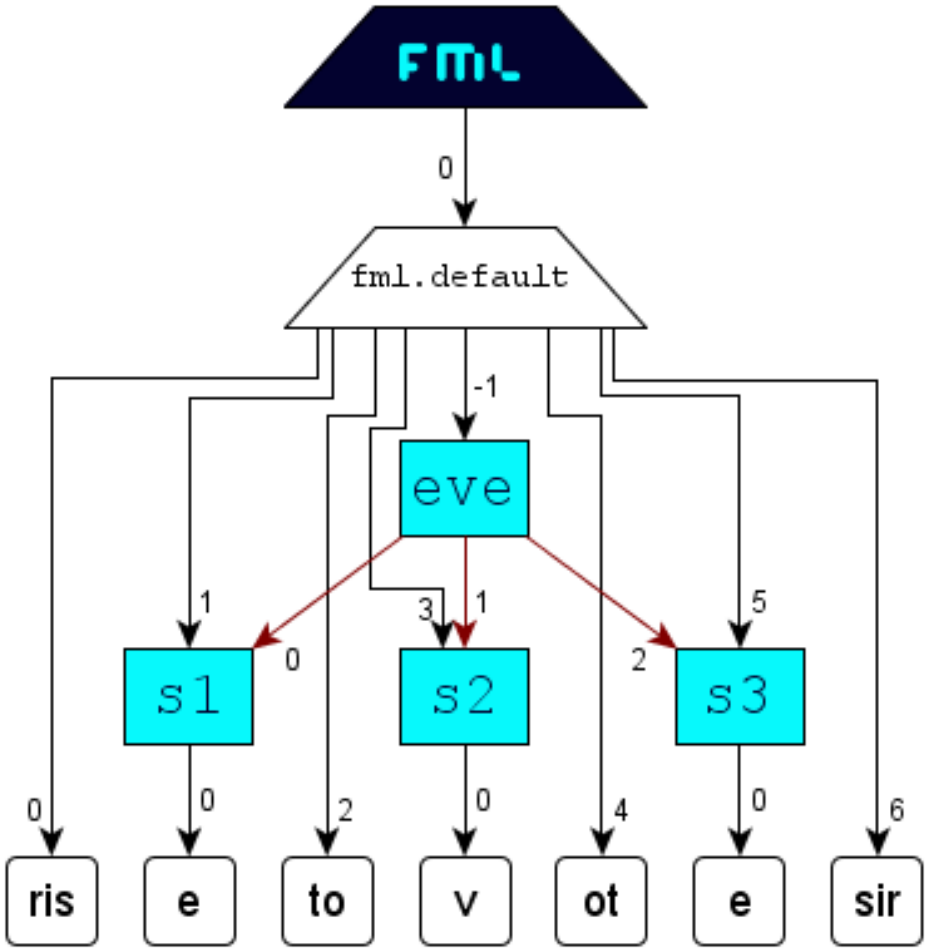


FML document:

```
ris
<s1 fml.segment.id="eve" fml.segment.pos="0">
    e
</s1>
to
<s2 fml.segment.id="eve" fml.segment.pos="1">
    v
</s2>
ot
<s3 fml.segment.id="eve" fml.segment.pos="2">
    e
</s3>
sir
```

segmentation.fml

FML graph:



Fragmentation

A fragment of a wellformed FML document is also a wellformed FML document. As a consequence of that, start-tags or end-tags may also be arranged outside of the document borders. If a start-tag can not find a corresponding end-tag, then it may be assumed that it marks up to the very end of the document. For the purpose of document interpretation the missing end-tag will be inserted at the end of the document. If a end-tag can not find a corresponding start-tag, then it may be assumed that it marks up from the very beginning of the document. For the purpose of document interpretation the missing start-tag will be inserted at the beginning of the document. If several tags have to be added at the beginning or at the end of the document, then those are subsumed into one multiple-tag.

That automatic completion takes place during the transformation of an FML document to an FML graph. For an FML graph-node of type fml.node.element the attribute `fml.element.autocompleted = "start-tag" | "end-tag"` indicates that automatic completion applies. The re-transformation to an FML document will reconstruct the fragmented document with isolated tags.

The main benefits of the concept fragmentation are, that documents can be serialized and exchanged fragmented, and that - according to the requirement entropy - a sparingly encoding is generally possible.

In the following example FML document$_1$ has been marked up with 4 tags. Each tag is isolated. Automatic completion (row `01` and `03`) will lead to FML document$_2$, which then may be interpreted in an FML graph.

FML document$_1$:

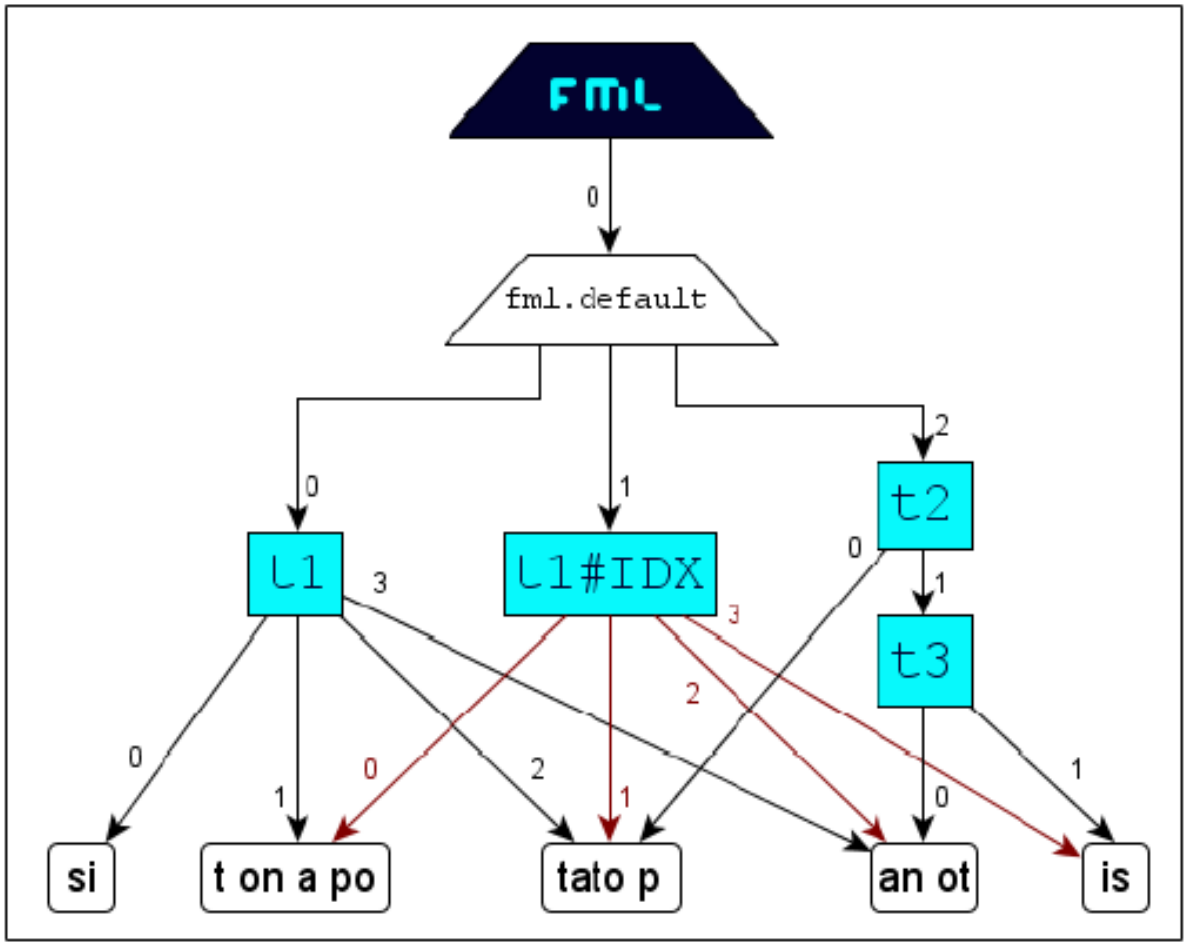
```
si<t1 #IDX>t on a po<t2>tato p<t3>an ot</t1>is
```

FML document$_2$:

```
01 <t1>
02 si<t1#IDX>t on a po<t2>tato p<t3>an ot</t1>is
03 </t1#IDX /t2 /t3>
```

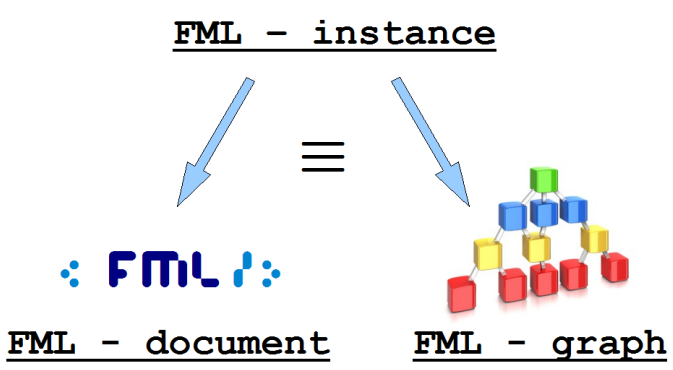
fragmentation.fml

FML graph:



Transformation

During transformation, an FML document will be converted to an FML graph. Markup has to be transformed into nodes. Parent-child-relations between the nodes have to be recognized. The result is a graph that visualizes the polyhierarchical relations between content and markup. The graph will be clear without ambiguity and may be transformed back into exactly the same original document.



The followings steps have to be taken for a secure transformation *FML*-document→*FML*-graph:

1. Lexical analysis: The *FML*-document gets checked for coding- and grammar-consensus.
2. Parsing: Transforming the *FML*-document into a graph representing the document grammar (see section "Grammar").
3. Extension: According to the fragmentation missing start- or end-tags will be put in front or at the end of the document using multiple-tags.
4. Identification: Identifying corresponding tag pairs.
5. Tag→Element: Corresponding tag pairs will be transformed. At this point the term element (representing the major *FML*-graph-node) gets introduced.
6. Reorganization: Realization of various reorganization actions for building up a consistent *FML*-graph compliant to the definition (see FML graph).
7. Segment node: According to the segmentation, virtual segment-nodes will be inserted.

XML Representation

A milestone-based *XML*-representation for *FML* is available:

- `fml.xml.xsd`
- `xml2fml.xslt`
- `document.fml`
- `document.xml`

Survey

The evaluation of the *FML* requirements has been accompanied by a survey about the *freestyle*-concepts

- interference (see concept interference),
- congruence (see concept congruence),
- independence (see concept independence),
- segmentation (see concept segmentation).

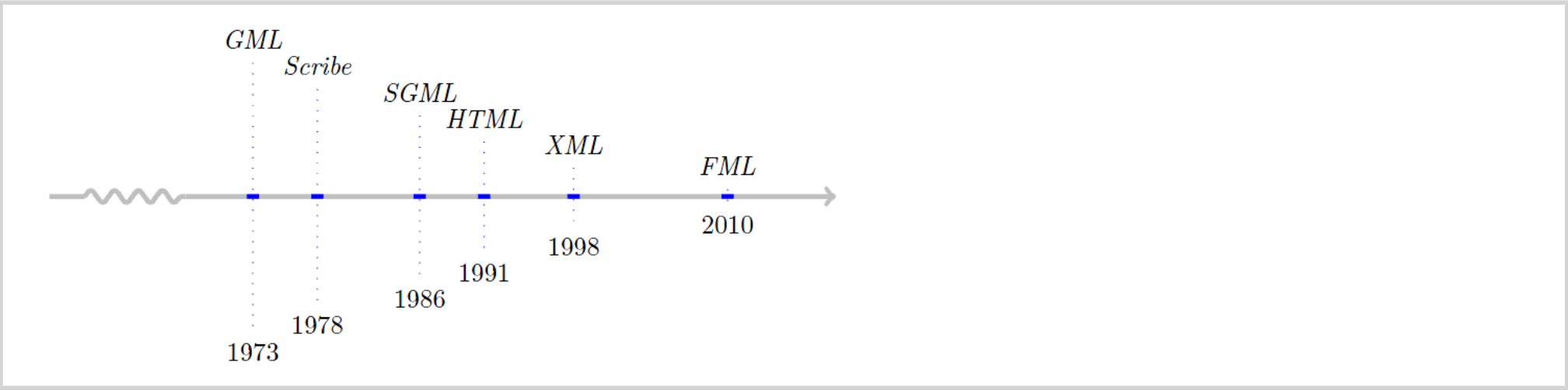
The conversation with many randomly selected *XML*-experienced participants at the fair trade CeBIT, Hannover, Germany, was indeed very inspiring. Details you may retrieve at http://www.freestyle-markup.org/survey.

As expected, the acceptance correlates with the clearness of the document and graph visualisation, and lies between 13-79 percent. The primary reasons for refusal or

scepticism were missing *XML*-conformance and a lack of imaginable application scenarios for the unfamiliar concepts. That is understandable. But we have to point out that the four discussed concepts are totally optional. If document structures are simple enough, there will be no need to care about features beyond monohierarchies. Therefore, each single positive voice justifies markup evolution in general.

Reference Analysis

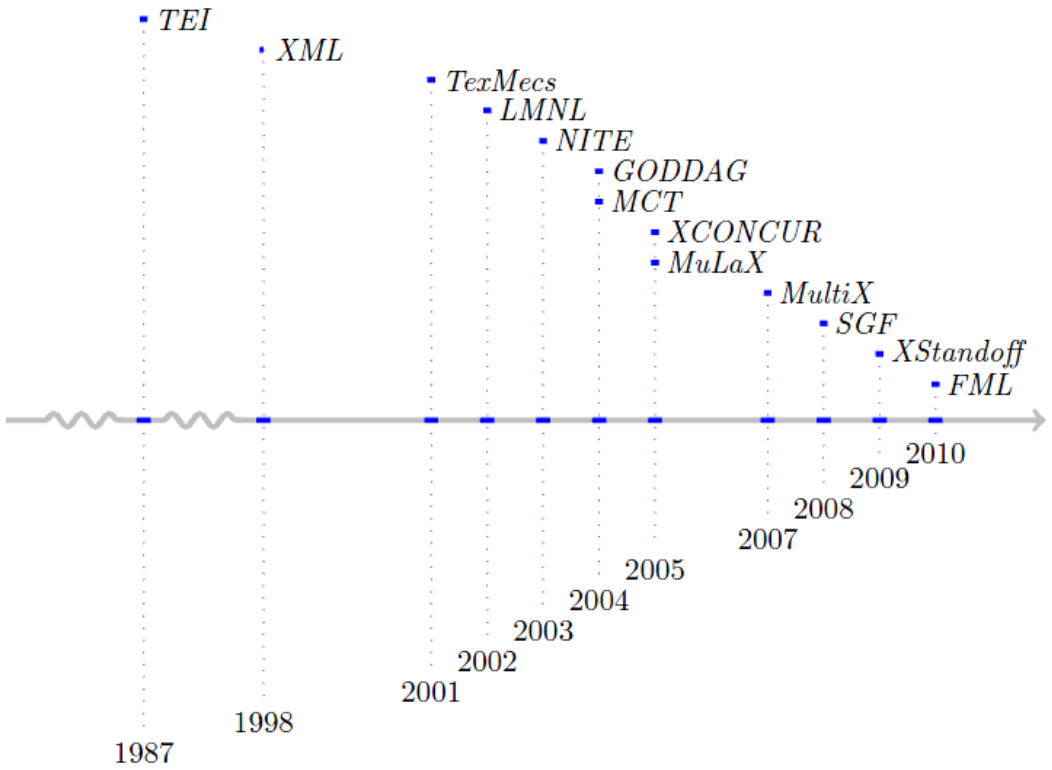
The first timeline illustrates how *FML* arranges proportional among the most established comparable markup technologies:



For the figuration of the *FML*-requirements recent markup approaches that deal with comparable issues, mainly interference and independence, have been identified and inspected:

- GODDAG [Sperberg-McQueen 2000],
- LMNL [Tennison 2002],
- MCT [Jagadish 2004],
- MuLaX/XCONCUR [Hilbert 2005],
- MultiX [Chatti 2007],
- NITE [Carletta 2003],
- SGF/XStandoff [Stührenberg 2009],
- TEI [Sperberg-McQueen 2009],
- TexMECS [Huitfeldt 2001].

The second timeline shows how *FML* proportionally integrates among these approaches:



Using the extended classification from Witt 2004, the listed comparable approaches deal with interfering and independent structures in the following ways[11]:

	FML	GODDAG	LMNL	MCT	MuLaX/XCONCUR	MultiX	NITE	SGF/XStandoff	SGML	TEI	TexMECS	XML
freestyle	✔	✔	✔								✔	
concur				✔	✔				✔			
milestone			✔							✔		
fragmentation			✔			✔				✔	✔	
virtual elements			✔	✔		✔	✔	✔		✔	✔	
stand-off				✔			✔	✔		✔		
redundant encoding				✔						✔		✔

The following overview evaluates how the listed approaches refer to the *FML* requirements. Does the technology discuss the requirement aspect in a solution oriented way (+), or is the aspect off its research (-)[12] ?

technology	FML requirements																		
	grammar	compatibility	monohierarchy	interference	identification	congruence	independence	segmentation	fragmentation	wildcard	inheritance	graph representation	transformation	unambiguousness	entropy	ergonomics	internationalization	reference implementation	specification
FML	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
GODDAG	−	−	+	+	−	−	−	−	−	−	−	+	+	−	−	−	−	−	−
LMNL	+	+	+	+	−	−	−	−	−	−	−	+	−	−	−	−	−	−	−
MCT	−	+	+	−	−	−	+	−	−	−	−	+	−	−	−	−	−	+	−
MuLaX/XCONCUR	+	+	+	−	−	−	+	−	−	−	−	+	+	+	−	−	−	+	+
MultiX	−	−	+	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−
NITE	−	−	+	−	−	−	+	−	−	−	−	+	+	+	−	−	−	+	+
SGF/XStandoff	−	+	+	−	−	−	+	−	−	−	−	+	−	−	−	−	−	−	−
TEI	−	−	+	+	−	−	+	−	−	−	−	−	−	−	−	−	−	−	+
TexMECS	+	−	+	+	+	−	−	+	−	−	−	+	+	−	−	−	−	−	−
*XML**	+	+	+	−	−	−	−	−	−	−	−	+	+	+	−	−	+	+	+

Many markup syntaxes other than *XML* have existed. Also various approaches for representing markup structures in data models. Why *FML*? Why should *FML* be different? First of all *FML* generally addresses more requirements at once, therefore provides more features. And second of all it is more than a thought experiment or an academic mind game. In particular the data-centric approach and the need of data modelers and software engineers is addressed. The development of *FML* initially began with a carefull reference analysis and the identification of appropriate requirements and broadly desired markup features. The development will end with a consistent specification and reference implementation. *FML* strives for the goal to provide completed and verified results, an implementation and specification of a base technology, ready for immediate software-integration.

Future Work

For the near future the following actions are planned:

- setting up a reference implementation for *FML*,
- launching an editor for authoring *FML*-documents and visualizing *FML*-graphs,
- integrating expected feedback,

- proofing sufficiency with concrete scenarios,
- maintenance of the project platform http://www.freestyle-markup.org,
- conceptualization for secondary technologies:
    - *FML Schema*
    - *FML Query*
    - *FML Transformation*

Working with markup implicates far more than only serializing and deserializing between *FML*-documents and corresponding *FML*-graphs. Markup documents will be validated, transformed, scanned, extracted, edited, enriched, visualized, compressed, transported, archived, and processed in various ways. The situation is comparable to other theoretical foundations like Codd's RDBMS:

> relational databases required hundreds of thousands algorithmic innovations to make it work well
>
> — Orlowski 2003

Basically, processing a polyhierarchical structure will cost more efforts than processing *XML*, since the structure is less regular. Secondary technologies have to be discussed intensely. Future work continues here.

Résumé

The future of markup is an incontrovertible matter [Adler 2010].

The new and innovative technology

> *Freestyle Markup Language*

has been set up to the fundamental state of a preliminary specification.

The basis of its constructive development was given by the dominant de facto standard *XML*, various deficit-discourses, discussions about alternative solutions, as well as personal developer experiences in the areas single source publishing, cross-media content production, content management systems, and data modelling in general.

The objective of *FML* is to overcome current markup restrictions and to encourage the evolution of markup. In particular, the terminologies "overlapping markup", "concurring markup", "crossover markup", "multiple annotated markup", "multi-dimensional markup", and "multi-hierarchical markup" are addressed. *FML* acts with its contrary *freestyle*-approach against the monohierarchical "properly nested"-idea of the restrictive language *XML* and encourages an almost unlimited and unrestricted use of markup without any root- or hierarchy-bondage. Even corresponding tags may be omitted: If a semantic unit starts and applies to the rest of the document, just and only place a start-tag, if a semantic unit ends and applies from the beginning of the document, just and only place an end-tag. Do not care about nesting! Concentrate on semantically marking your content up!

The identified and within a survey evaluated requirements for *FML* have been discussed, *freestyle*-concepts were pictured, a formal document-grammar and graph-representation were presented, a *XML*-milestone-representation was defined, and future work has been outlined.

Scenarios for the usage of *FML* would mostly arise from systems, where a

transparent basis technology for an unambiguous interchange of data is required,
- various perspectives across the same redundance-free document and
- semistructured data have to be marked up.

The following areas of application would be predestinated for a cooperation:

- knowledge data bases,
- typographic description languages,
- document digitalization,
- CMP, CMS, EDMS-applications,
- multiple authoring systems,
- collaboration software,
- revision control systems,
- communication protocols and
- persistence media.

Finally, a typographic marked up scenario[13], illustrated as a word cloud, will visualize the primary terminologies relevant to this study:

References

[Adler 2006] Sharon Adler, Roberta Cochrane, John F. Morar Alfred Spector: “Technical context and cultural consequences of XML”. IBM SYSTEMS JOURNAL ’Celebrating 10 Years of XML’, 45(2):207–223, Jun. 2006, http://www.research.ibm.com/journal/sj45-2.html.

[Adler 2010] Sharon C. Adler: “Why is Markup Still Important after 30 years and Will it Still be Important in Another 30”. XML Prague 2010 - Conference Proceedings, Prague, Mar. 2010, http://www.xmlprague.cz/2010/presentations/Sharon_C_Adler_Why_is_Markup_Still_Important_after_30_years_and_Will_it_Still_be_Important_in_Another_30.pdf.

[Brüggemann 1993] Anne Brüggemann-Klein: “Formal Models in Document Processing”. Habilitation, Institut für Informatik, Universität Freiburg, 1993, ftp://ftp.informatik.uni-freiburg.de/documents/papers/brueggem/habil.ps.

[Cajori 1928] Florian Cajori: “Cajori's A History of Mathematical Notations: Notations in Elementary Mathematics”. Open Court, 1928.

[Carletta 2003] Jean Carletta, Jonathan Kilgour, Timothy J. O’Donnell, Stefan Evert, Holger Voormann: “The NITE Object Model Library for Handling Structured Linguistic Annotation on Multimodal Data Sets”. In: Proceedings of the EACL Workshop on Language Technology and the Semantic Web (3rd Workshop on NLP and XML, NLPXML-2003), Apr. 2003, http://www.ims.uni-stuttgart.de/projekte/corplex/paper/evert/CarlettaEtal2003.pdf.

[Chatti 2007] Noureddine Chatti, Souha Kaouk, Sylvie Calabretto, Jean-Marie Pinon: “MultiX: an XML-based formalism to encode multi-structured documents”. In:

Proceedings of Extreme Markup Languages, Montréal, Aug. 2007.

[Collins 1969] A. Collins, M. Quillian: “Retrieval time from semantic memory”. Journal of Verbal Learning and Verbal Behavior, 8(2):240–247, Apr. 1969, doi:10.1016/S0022-5371(69)80069-1.

[Coombs 1987] James H. Coombs, Allen H. Renear und Steven J. DeRose: “Markup systems and the future of scholarly text processing”. Commun. ACM, 30(11):933–947, 1987, http://www.fdi.ucm.es/profesor/jlsierra/e-learning/primera-sesion/MarkupSystems.pdf, doi:10.1145/32206.32209.

[DeRose 1990] Steven J. DeRose, David G. Durand, Elli Mylonas, Allen H. Renear: “What is text, really?”, Journal of Computing in Higher Education, 1(2):3–26, Dec. 1990, http://antonietta.philo.unibo.it/IUcorso2006-07/risorse/ohco.pdf, doi:10.1007/BF02941632.

[Durand 1996] David G. Durand, Elli Mylonas, Steven J. Derose: “What Should Markup Really Be? Applying theories of text to the design of markup systems”. 1996, http://xml.coverpages.org/DurandWhatShouldTextBe-ALLC1996.pdf.

[Durusau 2002] Patrick Durusau, Matthew Brook O’Donnell: “Coming down from the trees. Next step in the evolution of markup?”. In: Proceedings of Extreme Markup Languages, Montréal, Aug. 2002.

[Ernst 2009] Daniel J. Ernst, Daniel E. Stevenson, Paul J. Wagner: “Hybrid and custom data structures: evolution of the data structures course”. SIGCSE Bull, Vol. 41, Nr. 3, 2009, pages 213-217, http://www.cs.uwec.edu/~ernstdj/Papers/hybrid.pdf, doi:10.1145/1595496.1562945.

[Firesmith1995] Donald Firesmith: “Inheritance Guidelines”. JOOP - Journal of Object-Oriented Programming, 8(2):67–72, 1995.

[Fischer 2004] Steven Roger Fischer: “A History of Writing”. Reaktion Books, Apr. 2004.

[Freytag 2007] Asmus Freytag, Martin Dürst: “Unicode in XML and other Markup Languages”. W3C Note, Mai 2007, http://www.w3.org/TR/2007/NOTE-unicode-xml-20070516.

[Goldbarth 2003] Albert Goldbarth: “Pieces of Payne: A Novel”. Graywolf Press, Apr. 2003.

[Guitton 2010] Pedro Guitton: “A homage to Typography”. Gingko Press, Barcelona, Spain, Apr. 2010.

[Hilbert 2005] Mirco Hilbert, Andreas Witt, Oliver Schonefeld: “Making CONCUR work”. In: Proceedings of Extreme Markup Languages, Montréal, Aug. 2005.

[Huitfeldt 2001] Claus Huitfeldt, C. M. Sperberg-McQueen: “TexMECS: An experimental markup meta-language for complex documents”. Jan. 2001, http://decentius.aksis.uib.no/mlcd/2003/Papers/texmecs.html.

[Ide 2006] Nancy Ide und Keith Suderman: “Integrating linguistic resources: The american national corpus model”. In: Proceedings of the 6th International Conference on Language Resources and Evaluation, 2006, http://www.cs.vassar.edu/~ide/papers/ANC-LREC06.pdf.

[Idehen 2003] Kingsley Uyi Idehen: “Data Structures and RDF”. OpenLink Software, Inc., Lexington, MA, USA, May 2003, http://www.openlinksw.com-WEBLOG.

[Iverson 1980] Kenneth E. Iverson: “Notation as a Tool of Thought”. Commun. ACM, 23(8):444–465, 1980, http://www.jdl.ac.cn/turing/pdf/p444-iverson.pdf, doi:10.1145/358896.358899.

[Jagadish 2004] H. V. Jagadish, Divesh Srivastava, Laks V. S. Lakshmanan, Monica Scannapieco, Nuwee Wiwatwattana: “Colorful XML: One hierarchy isnt enough.”. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, Jun. 2004, pages 251–262, http://www.eecs.umich.edu/db/timber/files/mct.pdf, doi:10.1145/1007568.1007598.

[Orlowski 2003] Andrew Orlowski: “Bruce Lindsay on Codd’s relational legacy”. The Register, Apr. 2003, http://www.theregister.co.uk/2003/04/25/bruce_lindsay_on_codds_relational.

[Prescod 2000] Paul Prescod: “From markup to object model. The XML abstraction problem and XML property objects”. XML EUROPE 2000, 6 2000, ISOGEN Inc., Paris, France, http://www.gca.org/papers/xmleurope2000/pdf/s12-02.pdf.

[Raymond 1993] Darrell R. Raymond, Frank W. Tompa: “Markup reconsidered”. Technical Report 356, Department of Computer Science, The University of Western

Ontario, 1993. Presented at the First International Workshop on the Principles of Document Processing, Washinton DC, Oct. 21-23, 1992, http://www.oasis-open.org/cover/raymmark.ps.

[Reid 1998] Brian Reid: “20 years of abstract markup - Any progress?”, Markup Technologies ’98 Conference, Nov. 1998, http://www.reid.org/~brian/markup98.ppt.

[Renear 1993] Allen Renear, Elli Mylonas und David Durand: “Refining our Notion of What Text Really Is: The Problem of Overlapping Hierarchies”. In: N. Ide und S. Hockey (publisher): Research in Humanities Computer, Oxford University Press, 1993, http://www.ideals.illinois.edu/bitstream/handle/2142/9407/RefiningOurNotion.pdf.

[Schmidt 2009] Desmond Schmidt, Robert Colomb: “A data structure for representing multi-version texts online”. Int. J. Hum.-Comput. Stud., 67(6):497–514, 2009, http://www.itee.uq.edu.au/~schmidt/_articles/elsevier.pdf, doi:10.1016/j.ijhcs.2009.02.001.

[Sperberg-McQueen 2000] C. M. Sperberg-McQueen, Claus Huitfeldt: “GODDAG: A Data Structure for Overlapping Hierarchies”. In: DDEP/PODDP, Springer, Sep. 2007, pages 213-217, http://decentius.aksis.uib.no/mlcd/2000/Papers/Goddag.pdf.

[Sperberg-McQueen 2007] C. M. Sperberg-McQueen: “Representation of overlapping structures”. In: Proceedings of Extreme Markup Languages, Montréal, Aug. 2007, http://conferences.idealliance.org/extreme/html/2007/SperbergMcQueen01/EML2007SperbergMcQueen01.html.

[Sperberg-McQueen 2008] C. M. Sperberg-McQueen, Claus Huitfeldt: “Markup Discontinued: Discontinuity in TexMecs, Goddag structures, and rabbit/duck grammars”. In: Proceedings of Balisage: The Markup Conference 2008, Montréal, Aug. 2008, doi:10.4242/BalisageVol1.Sperberg-McQueen01.

[Sperberg-McQueen 2009] C. M. Sperberg-McQueen, C. Michael, Lou Burnard: “TEI P5: Guidelines for Electronic Text Encoding and Interchange”. The Text Encoding Initiative Consortium, Jul. 2009, http://www.tei-c.org/release/doc/tei-p5-doc/en/Guidelines.pdf.

[Stührenberg 2006] Maik Stührenberg, Andreas Witt, Daniela Goecke, Dieter Metzing, Oliver Schonefeld: “Multidimensional Markup and Heterogeneous Linguistic Resources”. In: Proceedings of the 5th Workshop on NLP and XML (NLPXML-2006): Multi-Dimensional Markup in Natural Language Processing., Trento, Italy, Apr. 4, 2006, http://www.aclweb.org/anthology/W/W06/W06-2715.pdf.

[Stührenberg 2009] Maik Stührenberg, Daniel Jettka: “A toolkit for multidimensional markup: The development of SGF to XStandoff”. In: Proceedings of Balisage: The Markup Conference 2009, Montréal, Aug. 2009, doi:10.4242/BalisageVol3.Stuhrenberg01.

[Tennison 2002] Jeni Tennison, Wendell Piez: “The Layered Markup and Annotation Language (LMNL)”. In: Extreme Markup Languages Conference, Montréal, Aug. 2002, http://xml.coverpages.org/LMNL-Abstract.html.

[XInclude] W3C: “XML Inclusions (XInclude) Version 1.0 (Second Edition)”. Recommendation REC-xinclude-20061115, Nov. 2006, http://www.w3.org/TR/xinclude.

[Walsh 2003] Norman Walsh: “Escaped Markup Considered Harmful”. xml.com, Aug. 2003, http://www.xml.com/pub/a/2003/08/20/embedded.html.

[Wikipedia 2009] Wikipedia: “Freistil — Wikipedia, Die freie Enzyklopädie”. Online, 2009, http://de.wikipedia.org/w/index.php?title=Freistil&oldid=58624703 [online as at 5.7.2010].

[Winter 2002] Andreas Winter, Bernt Kullbach, Volker Riediger: “An Overview of the GXL Graph Exchange Language”. Lecture Notes in Computer Science, 2269:324–337, 2002, http://www.gupro.de/GXL/Publications/winter+2002.pdf, doi:10.1007/3-540-45875-1_25.

[Witt 2002] Andreas Witt: “Meaning and interpretation of concurrent markup”. In: ALLC/ACH 2002, New directions in humanities computing. The 14th Joint International Conference of the ALLC and ACH., pages 145–147. Tübingen, 2002, http://xml.coverpages.org/Witt-allc2002.html.

[Witt 2004] Andreas Witt: “Multiple hierarchies: new aspects of an old solution”. In: Extreme Markup Languages Conference, Montréal, Aug. 2004, http://conferences.idealliance.org/extreme/html/2004/Witt01/EML2004Witt01.html.

[Witt 2007] Andreas Witt: “Guideline: Multiple Hierarchies”. In: Digital Historical Corpora - Architecture, Annotation and Retrieval (Dagstuhl Seminar Proceedings), 2007, http://drops.dagstuhl.de/volltexte/2007/1040/pdf/06491.WittAndreas.Paper.1040.pdf.

[1] Organisation for the administration of the text corpus of American English. Currently 22 million words. http://www.anc.org.

[2] The term fragmentation means document-fragmentation and has absolutely nothing in common with element-fragmentation, an *XML*-workaround for handling interference.

[3] *UTF-8* is used by 55.3% of *HTML*/*XHTML*-websites whose character encoding is known [http://w3techs.com/technologies/overview/character_encoding as at 5.7.2010].

[4] Geometrical shape, foreground, background, font, text (labeling is optional and may be shortened arbitrarily by '...'-substitution).

[5] Corresponding components in an FML document. Mapping between components and nodes is realized during transformation.

[6] Notation: *UML*-multiplicity with `0..1 ='?'`, `1..* ='+'`, `0..* ='*'`.

[7] Notation: *UML*-multiplicity with `0..1 ='?'`, `1..* ='+'`, `0..* ='*'`.

[8] Backslash escaping instead of character entities has been choosen as a design choice of question. Character entities are cryptic and not necessary in an *UTF-8*-document (*FML*-documents must be *UTF-8* encoded!). Developers commonly prefer the accepted escape character '\'.

[9] That is the only known german true pangram.

[10] This first palindromic sentence in English is credited to John Taylor (*1580 - 1653*). It uses an acceptable 17th century spelling of *dwell*.

[11] The tabular classification gives a basic overview, but only pretends an absolute assignment: Each markup approach has to be distinguish for itself individually.

[12] This binary classification is not a judgment, just a subjective appraisal in the in the authors discretion.

[13] A kindly "thank-you" to http://www.wordle.net.

Author's keywords for this paper: freestyle; markup; language; crossover; concurring; overlapping; polyhierarchical; multi-hierarchical; multiple annotations; multiple perspectives; specification

Denis Pondorf
<contact@freestyle-markup.org>
IT-Freelancer and PhD student
Hamburg, Germany

Works as a senior consultant in the fields of customer relationship management, business continuity management and cross media publishing for various IT-companies.

As a computational scientist his research interests focuses on the evolution of markup languages and on the formal generation of pangrams, isograms and lipograms.

Andreas Witt
<witt@ids-mannheim.de>
Senior Researcher
Institute for the German Language (IDS), Mannheim, Germany

Witt received his Ph.D. in Computational Linguistics and Text Technology from the Bielefeld University in 2002 (dissertation title: Multiple Informationsstrukturierung mit Auszeichnungssprachen. XML-basierte Methoden und deren Nutzen für die Sprachtechnologie).

After graduating in 1996, he started as a researcher and instructor in Computational Linguistics and Text Technology. He was heavily involved in the establishment of the minor subject Text Technology in Bielefeld University´s Magister and B.A. program in 1999 and 2002 respectively. After his Ph.D. in 2002 he became an assistant lecturer, still at the Text Technology group in Bielefeld. In 2006 he moved to Tübingen University, where he was involved in a project on "Sustainability of Linguistic Resources" and in projects on the interoperability of language data. Since 2009 he is senior researcher at "Institute für Deutsche Sprache" (Institute for the German Language) in Mannheim.

Witt is and was a member of several research organizations, amongst them the TEI Special Interest Group on overlapping markup, for which he was involved in the writing of the latest version of the chapter "Multiple Hierarchies", which is included in TEI-Guidelines P5.

Witt's main research interests deal with questions on the use and limitations of markup languages for the linguistic description of language data.