SiSU
Description

Ralph Amissah

Document information:
sourcefile sisu_description.sst
Generated by SiSU www.jus.uio.no/sisu
version information: SiSU 0.59.0 of 2007w38/0

For alternative output formats of this document check:
<http://www.jus.uio.no/sisu/sisu_description/sisu_manifest.html>

Contents

SiSU - Description, Ralph Amissah

SiSU an attempt to describe

1. Description

1.1 Outline

SiSU is a flexible document preparation, generation publishing and search system.[1]

SiSU ("**SiSU** information Structuring Universe" or "Structured information, Serialized Units"),[2] is a Unix command line oriented framework for document structuring, publishing and search. Featuring minimalistic markup, multiple standard outputs, a common citation system, and granular search.

Using markup applied to a document, **SiSU** can produce plain text, HTML, XHTML, XML, OpenDocument, LaTeX or PDF files, and populate an SQL database with objects[3] (equating generally to paragraph-sized chunks) so searches may be performed and matches returned with that degree of granularity (e.g. your search criteria is met by these documents and at these locations within each document). Document output formats share a common object numbering system for locating content. This is particularly suitable for "published" works (finalized texts as opposed to works that are frequently changed or updated) for which it provides a fixed means of reference of content.

SiSU is the data/information structuring and transforming tool, that has resulted from work on one of the oldest law web projects. It makes possible the one time, simple human readable markup of documents, that

[1] This information was first placed on the web 12 November 2002; with predating material taken from <http://www.jus.uio.no/lm/lm.information/toc.html> part of a site started and developed since 1993. See document metadata section <http://www.jus.uio.no/sisu/SiSU/metadata.html> for information on this version. Dates related to the development of **SiSU** are mostly contained within the Chronology section of this document, e.g. <http://www.jus.uio.no/sisu/sisu_chronology>

[2] also chosen for the meaning of the Finnish term "sisu".

[3] objects include: headings, paragraphs, verse, tables, images, but not footnotes/endnotes which are numbered separately and tied to the object from which they are referenced.

SiSU can then publish in various forms, suitable for paper[4] , web[5] and relational database[6] presentations, retaining common data-structure and meta-information across the output/presentation formats. Several requirements of legal and scholarly publication on the web have been addressed, including the age old need to be able to reliably cite/pinpoint text within a document, to easily make footnotes/endnotes, to allow for semantic document meta-tagging, and to keep required markup to a minimum. These and other features of interest are listed and described below. A few points are worth making early (and will be repeated a number of times):

(i) The **SiSU** document generator was the first to place material on the web with a system that makes possible citation across different document types, with paragraph, or rather object citation numbering[7] a text positioning system, available for the pinpointing of text, 1997, a simple idea from which much benefit, and **SiSU** remains today, to the best of my knowledge, the only multiple format e-book/ electronic-document system on the web that gives you this possibility (including for relational databases).

(ii) Markup is done once for the multiple formats produced.

(iii) Markup is simple, and human readable (with a little practice), in almost all cases there is less and simpler markup required than basic html. In any event the markup required is very much simpler than the html, LaTeX, [lout], structured XML, ODF (OpenDocument), PostgreSQL or SQLite feed etc. that you can have **SiSU** generate for you.

(iv) **SiSU** is a batch processor, dealing with as many files as you need to generate at a time.

(v) Scalability is dependent on your file system (in my case Reiserfs), the database (currently Postgresql and/or SQLite) and your hardware.

SiSU Sabaki[8] (or just **SiSU**) is the provisional name given to the software described here that helps structure documents for web and other publication. The name **SiSU** is a loose anagram for something along the lines of ***"SiSU is structuring unit"*** , or *"**SiSU** , information structuring unit"* or the more descriptive *"Structured information, Serialized Units"* or ***"simple - information structuring unit"*** or the more descriptive *"Structured information, Serialized Units"* or what it may be directed towards ***"semantic** and **information structuring universe** "* ,[9] tongue in cheek, only just. Guess I'll get away with ***"Simple - information Structuring Universe"*** . **SiSU** is also a Finnish word roughly meaning guts, inner strength and perseverance.[10]

---

4 pdf via LaTeX or lout

5 currently html (two forms of html presentation one based on css the other on tables), and *PHP*; potentially structured XML

6 any SQL - currently PostgreSQL and *sqlite* (for portability, testing and development)

7 previously called "text object numbering"

8 **SiSU** Sabaki, release version. Pre-release version **SiSU** Scribe, and version prior to that **SiSU** nicknamed Scribbler. Pre-release versions go back several years. Both Scribbler and Scribe (still maintained) made system calls to **SiSU** 's various parts, instead of using libraries.

9 A little universe it may be, but semantic you may have a hard time getting away with, given the meaning the word has taken on with markup. On a document wide basis semantic information may be provided, which can be really useful, (and meaningful, especially) if you have a large document set, and use this with rss feeds or in an sql database etc. On a markup level, I have little inclination to add semantic markup formally beyond references, title, author [Dublin Core entities? addresses?] etc. Actually this deserves a bit of thought possibly use letter tags (including letter alias/synonyms for font faces) to create a small set of default semantic tags, with the possibility for per document adjustments. Will seek to permit XML entity tagging, within **SiSU** markup and have that ignored/removed by the parts of the program that have no use for it.

10 "Sisu refers not to the courage of optimism, but to a concept of life that says, 'I may not win, but I will gladly give my life for what I believe."' Aini Rajanen, Of Finnish Ways, 1981, p. 10.
<http://www.humanlanguages.com/finnishenglish/rlfs.htm>
"Every Finn has his own pet definition. To me, sisu means patience without passion. But there are many varieties of sisu. Sisu can be a sudden outburst or it can be the kind that lasts. A man can have both kinds. It is outside reason. It is something in the soul. It comes from oneself. For instance, it makes a soldier do things because he himself must,

SiSU was born of the need to find a way, with minimal effort, and for as wide a range of document types as possible, to produce high quality publishing output in a variety of document formats. As such it was necessary to find a simple document representation that would work across a large number of document types, and the most convenient way(s) to produce acceptable output formats. The project leading to this program was started in 1993 (together with the trade law project now known as Lex Mercatoria) as an investigation of how to effectively/efficiently place documents on the web. The unified document handling, together with features such as paragraph numbering, endnote handling and tables... appeared in 1996/97. **SiSU** was originally written in Perl,[11] and converted to **Ruby** , [12] in 2000, one of the most impressive programming languages in existence! In its current form it has been written to run on the **Gnu** /Linux platform, and in particular on **Debian** , [13] taking advantage of many of the wonderful projects that are available there.

SiSU markup is based on requiring the minimum markup needed to determine the structure of a document. (This can be as little as saying in a header to look for the word Book at a specified level and the word Chapter at another level). **SiSU** then breaks a document into its smallest parts (at a heading, and paragraph level) while retaining all structural information. This break up of the document and information on its structure is taken advantage of in the transformations made in generating the very different output types that can be created, and in providing as much as can be for what each output type is best at doing, e.g. LaTeX (professional document typesetting, easy conversion to pdf or Postscript), XML (in this case, structural representation), ODF (OpenDocument [experimental]), SQL (e.g. document search; representing constituent parts of documents based on their structure, headings, chapters, paragraphs as required; user control).[14]

From markup that is simpler and more sparse than html you get:

- far greater output possibilities, including html, XML, ODF (OpenDocument), LaTeX (pdf), and SQL;
- the advantages implicit in the very different output possibilities;
- a common citation system (for all outputs - including the relational database, search results are relevant for all outputs);

For more see the short summary of features provided below.

SiSU processes files with minimal tagging to produce various document outputs including html, LaTeX or lout (which is converted to pdf) and if required loads the structured information into an SQL database (PostgreSQL and SQLite have been used for this). **SiSU** produces an intermediate processing format.[15]

SiSU is used in constructing Lex Mercatoria <http://lexmercatoria.org/> or <http://www.jus.uio.no/lm/> (one of the oldest law web sites), and considerable thought went into producing output that would be suitable for legal and academic writings (that do not have formulae) given the limitations of html, and publication in a wide variety of "formats", in particular in relation to the convenient and accurate citation of text. However, the construction of Lex Mercatoria uses only a fraction of the features available from **SiSU** today, *vis* generation of flat file structures, rather than in addition

not because he has been told." Paavo Nurmi <http://personalweb.smcvt.edu/tmatikainen/finnishtraditions.htm>

[11] <http://www.perl.org/>

[12] <http://www.ruby-lang.org/en/>

[13] <http://www.debian.org/>

[14] where explicit structure is provided through the use of tagging headings, it could be reduced (still) further, for example by reducing the number of characters used to identify heading levels; but in many cases even that information is not required as regular expressions can be used to extract the implicit structure.

[15] This proved to be the easiest way to develop syntax, changes could be made, or alternatives provided for the markup syntax whilst the intermediate markup syntax was largely held constant. There is actually an optional second intermediate markup format in YAML <http://www.yaml.org/>

the building of ("granular") SQL database content, (at an object level with relevant relational tables, and other outputs also available).

1.2 Short summary of features

(i) markup syntax: (a) simpler than html, (b) mnemonic, influenced by mail/messaging/wiki markup practices, (c) human readable, and easily writable,

(ii) (a) minimal markup requirement, (b) single file marked up for multiple outputs,

notes:

* documents are prepared in a single UTF-8 file using a minimalistic mnemonic syntax. Typical literature, documents like "War and Peace" require almost no markup, and most of the headers are optional.

* markup is easily readable/parsed by the human eye, (basic markup is simpler and more sparse than the most basic html), [this may also be converted to XML representations of the same input/source document].

* markup defines document structure (this may be done once in a header pattern-match description, or for heading levels individually); basic text attributes (bold, italics, underscore, strike-through etc.) as required; and semantic information related to the document (header information, extended beyond the Dublin core and easily further extended as required); the headers may also contain processing instructions.

(iii) (a) multiple outputs primarily industry established and institutionally accepted open standard formats, include amongst others: plaintext (UTF-8); html; (structured) XML; ODF (Open Document text)l; LaTeX; PDF (via LaTeX); SQL type databases (currently PostgreSQL and SQLite). Also produces: concordance files; document content certificates (md5 or sha256 digests of headings, paragraphs, images etc.) and html manifests (and sitemaps of content). (b) takes advantage of the strengths implicit in these very different output types, (e.g. PDFs produced using typesetting of LaTeX, databases populated with documents at an individual object/paragraph level, making possible granular search (and related possibilities))

(iv) outputs share a common numbering system (dubbed "object citation numbering" (ocn)) that is meaningful (to man and machine) across various digital outputs whether paper, screen, or database oriented, (PDF, html, XML, sqlite, postgresql), this numbering system can be used to reference content.

(v) SQL databases are populated at an object level (roughly headings, paragraphs, verse, tables) and become searchable with that degree of granularity, the output information provides the object/paragraph numbers which are relevant across all generated outputs; it is also possible to look at just the matching paragraphs of the documents in the database; [output indexing also work well with search indexing tools like hyperesteier].

(vi) use of semantic meta-tags in headers permit the addition of semantic information on documents, (the available fields are easily extended)

(vii) creates organised directory/file structure for (file-system) output, easily mapped with its clearly defined structure, with all text objects numbered, you know in advance where in each document output type, a bit of text will be found (e.g. from an SQL search, you know where to go to find the prepared html output or PDF etc.)... there is more; easy directory management and document associations, the document preparation (sub-)directory may be used to determine output (sub-)directory, the skin used, and the SQL database used,

(viii) "Concordance file" wordmap, consisting of all the words in a document and their (text/ object) locations within the text, (and the possibility of adding vocabularies),

(ix) document content certification and comparison considerations: (a)

the document and each object within it stamped with an md5 hash making it possible to easily check or guarantee that the substantive content of a document is unchanged, (b)version control, documents integrated with time based source control system, default RCS or CVS with use of $Id: sisu_description.sst,v 1.25 2007/08/23 12:22:36 ralph Exp $ tag, which **SiSU** checks

(x) **SiSU** 's minimalist markup makes for meaningful "diffing" of the substantive content of markup-files,

(xi) easily skinnable, document appearance on a project/site wide, directory wide, or document instance level easily controlled/changed,

(xii) in many cases a regular expression may be used (once in the document header) to define all or part of a documents structure obviating or reducing the need to provide structural markup within the document,

(xiii) prepared files may be batch process, documents produced are static files so this needs to be done only once but may be repeated for various reasons as desired (updated content, addition of new output formats, updated technology document presentations/representations)

(xiv) possible to pre-process, which permits: the easy creation of standard form documents, and templates/term-sheets, or; building of composite documents (master documents) from other sisu marked up documents, or marked up parts, i.e. import documents or parts of text into a main document should this be desired

there is a considerable degree of future-proofing, output representations are "upgradeable", and new document formats may be added.

(xv) there is a considerable degree of future-proofing, output representations are "upgradeable", and new document formats may be added: (a) modular, (thanks in no small part to **Ruby**) another output format required, write another module.... (b) easy to update output formats (eg html, XHTML, LaTeX/PDF produced can be updated in program and run against whole document set), (c) easy to add, modify, or have alternative syntax rules for input, should you need to,

(xvi) scalability, dependent on your file-system (ext3, Reiserfs, XFS, whatever) and on the relational database used (currently Postgresql and SQLite), and your hardware,

(xvii) only marked up files need be backed up, to secure the larger document set produced,

(xviii) document management,

(xix) Syntax highlighting for **SiSU** markup is available for a number of text editors.

(xx) remote operations: (a) run **SiSU** on a remote server, (having prepared sisu markup documents locally or on that server, i.e. this solution where sisu is installed on the remote server, would work whatever type of machine you chose to prepare your markup documents on), (b) generated document outputs may be posted by sisu to remote sites (using rsync/scp) (c)document source (plaintext utf-8) if shared on the net may be identified by its url and processed locally to produce the different document outputs.

(xxi) document source may be bundled together (automatically) with associated documents (multiple language versions or master document with inclusions) and images and sent as a zip file called a sisupod, if shared on the net these too may be processed locally to produce the desired document outputs, these may be downloaded, shared as email attachments, or processed by running sisu against them, either using a url or the filename.

(xxii) for basic document generation, the only software dependency is **Ruby** , and a few standard Unix tools (this covers plaintext, html, XML, ODF, LaTeX). To use a database you of course need that, and to convert the LaTeX generated to PDF, a LaTeX processor like tetex or

texlive.

as a developers tool it is flexible and extensible

SiSU was developed in relation to legal documents, and is strong across a wide variety of texts (law, literature...). **SiSU** handles images but is not suitable for formulae/ statistics, or for technical writing at this time.

SiSU has been developed and has been in use for several years. Requirements to cover a wide range of documents within its use domain have been explored.

Some modules are more mature than others, the most mature being Html and LaTeX / pdf. PostgreSQL and search functions are useable and together with *ocn* unique (to the best of my knowledge). The XML output document set is "well formed" but largely proof of concept.

1.3 How it works

SiSU markup is fairly minimalistic, it consists of: a (largely optional) document header, made up of information about the document (such as when it was published, who authored it, and granting what rights) and any processing instructions; and markup within text which is related to document structure and typeface. **SiSU** must be able to discern the structure of a document, (text headings and their levels in relation to each other), either from information provided in the instruction header or from markup within the text (or from a combination of both). Processing is done against an abstraction of the document comprising of information on the document's structure and its objects,[16] which the program serializes (providing the object numbers) and which are assigned hash sum values based on their content. This abstraction of information about document structure, objects, (and hash sums), provides considerable flexibility in representing documents different ways and for different purposes (e.g. search, document layout, publishing, content certification, concordance etc.), and makes it possible to take advantage of some of the strengths of established ways of representing documents, (or indeed to create new ones).

[16] objects include: headings, paragraphs, verse, tables, images, but not footnotes/endnotes which are numbered separately and tied to the object from which they are referenced.

1.4 Simple markup

SiSU markup is based on requiring the minimum markup needed to determine the structure of a document. (This can be as little as saying in a header to look for the word Book at a specified level and the word Chapter at another level). **SiSU** then breaks a document into its smallest parts (at a heading, and paragraph level) while retaining all structural information. This break up of the document and information on its structure is taken advantage of in the transformations made in generating the very different output types that can be created, and in providing as much as can be for what each output type is best at doing, e.g. LaTeX (professional document typesetting, easy conversion to pdf or Postscript), XML (in this case, structural representation), ODF (OpenDocument), SQL (e.g. document search; representing constituent parts of documents based on their structure, headings, chapters, paragraphs as required; user control).[17]

1.4.1 Sparse markup requirement, try to get the most out of markup

One of its strengths is that very small amounts of initial tagging is required for the program to generate its output.

This is a basic markup example:

[17] where explicit structure is provided through the use of tagging headings, it could be reduced (still) further, for example by reducing the number of characters used to identify heading levels; but in many cases even that information is not required as regular expressions can be used to extract the implicit structure.

- basic markup example, text file - an international convention [18]
- view basic markup, as it would be highlighted by vim editor [19]

Emphasis has been on simplicity and minimalism in markup requirements. Design philosophy is to try keep the amount of markup required low, for whatever has been determined to be acceptable output.[20]

SiSU 's markup is more minimalistic and simpler than (the equivalent) html and for it, you get considerably more than just html, as this preparation gives you all available output formats, upon request.

1.4.2 Single markup file provides multiple output formats

For each document, there is only one (input, minimalistically marked up) file from which all the available output types are generated.[21]

Eg. the markup example:

- original text file - an international convention [22]
- view as syntax would be highlighted by vim editor [23]

[18] <http://www.jus.uio.no/sisu/sample/markup/un_contracts_international_sale_of_goods_convention_1980.sst>
output provided as example in the next section

[19] <http://www.jus.uio.no/sisu/sample/syntax/un_contracts_international_sale_of_goods_convention_1980.sst.html>
as it would appear with syntax highlighting (by vim)

[20] seems there are several "smart ASCIIs" available, primarily for ascii to html conversion, that make this, and reasonable looking ascii their goal
<http://webseitz.fluxent.com/wiki/SmartAscii>
<http://daringfireball.net/projects/markdown/>
<http://www.textism.com/tools/textile/>

[21] These include richly laid out and linked html (table or css variants), *PHP*, LaTeX (from which pdf portrait and landscape documents are produced), texinfo (for info files etc.), and PostgreSQL and/or SQLite. And the opportunity to fairly easily build additional modules, such as XML. See the examples provided in this document.

[22] <http://www.jus.uio.no/sisu/sample/markup/un_contracts_international_sale_of_goods_convention_1980.sst>

[23] <http://www.jus.uio.no/sisu/sample/syntax/un_contracts_international_sale_of_goods_convention_1980.sst.html>

Produces the following output:

- Segmented html version of document [24]
- Full length html document [25]
- pdf landscape version of document [26]
- pdf portrait version of document [27]
- clean tex ascii version of document [28]
- *xml* sax version of document [29]
- *xml* dom version of document [30]
- Concordance [31]

(and in addition to these: PostgreSQL, SQLite, texinfo and ~~YAML~~ [32] versions if desired)

1.4.3 Syntax relatively easy to read and remember

Syntax is kept simple and mnemonic.[33]

[24] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/toc.html>

[25] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/doc.html>

[26] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/landscape.pdf>

[27] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/portrait.pdf>

[28] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/plain.txt>

[29] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/sax.xml>

[30] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/dom.xml>

[31] <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980/concordance.html>

[32] discontinued for the time being

[33] **SiSU** markup syntax, an incomplete summary: http://www.jus.uio.no/sisu/sisu_markup_table/doc.html#h200306
Visual check of elementary font face modifiers: **bold** **bold emphasis** *italics* underscore ~~strikethrough~~ superscript subscript

1.4.4 Kept simple by having a limited publishing feature set, and features identified as most important, are available across several document types

To keep **SiSU** markup sparse and simple **SiSU** deliberately provides a limited publishing feature set, including: indent levels; bold; italics; superscript; subscript; simple tables; images; tables of contents and; endnotes. Which in most cases are available across the different output formats.

The publishing feature set may be expanded as required.

1.5 Designed with usability in mind

Output is designed to be uniform, easy to read, navigate and cite.

1.6 Code separate from content

Code[34] is separated from content. This means that when changes are desired in the output presentation, the code that produces them, and not the marked up text data set (which could be thousands of documents) is modified. Separating code from content makes large scale changes to output appearance trivial, and permits the easy addition of new output modules.

1.7 Object citation numbering, a text or object positioning / citation system - "paragraph" (or text object) numbering, that remains same and usable across all output formats by people and machine

Object citation numbering is a simple object (text) positioning and citation system that is human relevant and machine useable, used by **SiSU** for all manner of presentations, and that is available for use in all text mappings. It is based on the automated sequential numbering of objects (roughly paragraphs, (headings, tables, verse) or other blocks of text or images etc.). The text positioning system (in which I claim copyright) is invaluable for publishing requiring the citing text across multiple output formats, and for the general mapping of text within a document:

- in html, html not being easily citeable (change font size, or use a different browser and the page on which specific text appears has changed), and
- across multiple formats being common to all output formats html/xml/pdf/sql output,
- the results of an sql search can just be "live" citation references to the documents in which the text is found, much like an index (see image examples provided). [35]

I claim copyright on the system I use which is the most basic of all, numbering all text in headings and paragraphs sequentially (with tables and images being treated as a single paragraph) and only footnotes/endnotes not following this numbering, as their position in text is not strictly determined, (a change from footnotes to endnotes would change their numbering), footnotes instead "belong" to the paragraph from which they are referenced, and have sequential numbers of their own.

SiSU has a paragraph numbering system, that remains the same regardless of the output format. This provides an effective means of citation, pinpointing text accurately in all output formats, using the same reference. This is particularly useful where text has to be located across different output formats - for example once html is printed the number of pages and pages on which given text is found will vary depending on the browser, its settings the font size setting etc. Similarly **SiSU** produces pdf in different

[34] the program that generates the documents

[35] http://www.jus.uio.no/sisu/SiSU/1.html#search

forms, eg. on the example site Lex Mercatoria as portrait and landscape documents - here too page numbering varies, but paragraph numbering is the same, *vis a vis* all versions of the text (portrait and landscape pdf and the html versions of the text, and as stored (with "paragraphs" as records) to the PostgreSQL or SQLite database).

These numbers are placed in the text margins and are intended to be independent of and not to interfere with authors tagging. [The citation system (object citation numbering system, automated "paragraph numbering") which is automatically generated and is common and identical across all document formats] The paragraph numbering system is more accurately described as an (text) object numbering system, as headings are also numbered... all headings and paragraphs are numbered sequentially. Endnotes are automatically numbered independently and rather "belong" to the paragraph from which they are referenced, as an endnote does not (necessarily) form a part of a documents sequence, (they may be produced as either endnotes or footnotes (or both depending on what output you choose to look at - if you take the segmented html version document provided as an example, you will find that the endnotes are placed both at the end of each section, and in a separate section of their own called endnotes, and these are hyper-linked)). An attractive feature of providing citation numbering in this way is that it is independent of the document structure... it remains the same regardless of what is done about the document structure.

The rules have been kept very simple, unique incremental object citation numbers are assigned to headings, paragraphs, verse, tables and images. It is possible to manually override this feature on a per heading or comment basis though this should be used exceptionally, it may be of use where there a substantive text, and the addition of a minor comment by the publisher that should not be mapped as part of the text.

The object citation number markers contain additional numbering information with regard to the document structure, that can be used for alternative presentations, including such detail as the type of object (heading, paragraph, table, image, etc.), numbered sequentially.

An advantage is that the numbering remains the same regardless of document structure.

Text object ("paragraph") numbering is the same for all output versions of the same document, vis html, pdf, pgsql, yaml etc.

In the relational database, as individual text objects of a document stored (and indexed) together with object numbers, and all versions of the document have the same numbering, the results of searches may be tailored just to provide the location of the search result in all available document formats.

Note: there is a bug in the released behaviour of object citation numbering, (not certain when it was introduced) tables should be numbered, ie each table gets an ocn, required amongst other things for relational database. This will be corrected in a future release. Citation numbering of existing documents that contain tables will changed.

1.8 Handling of Dublin Core meta-tags making use of the Resource Description Framework

SiSU is able to use meta tags based on the Dublin Core[36] and Resource Description Framework[37]

This provides the means of providing semantic information about a document, both as computer processable meta-tags, and as human readable information that may be of value for classification purposes.

This information is provided both in html metatags, and (where available) under the section titled "Document Information - MetaData", near the end

[36] <http://dublincore.org/>
[37] <http://www.w3.org/RDF/>

of a document, for example in the segmented html version of this text at: <http://www.jus.uio.no/sisu/SiSU/metadata.html>

1.9 Easy directory management

1. Directory file association, skins and special image management, made simpler.[38]

The last part of the name of the work directory in which markup is being done, or rather from where **SiSU** is run in order to generate document output, is used in determining the sub-directory name for output files, that is created in the document output directory. This provides a rather easy way to associate documents e.g. of a given subject, or by owner.

```
/www/docs
  /intellectual_property
  /arbitration
  /contract_law
/www/docs
  /ralph
  /sisu
```

all are placed in their own directories within the directory structure created. Similar rules are used in the creation of sql type databases (though they can be overridden).

There are a couple of further associations with these directories.

Directory wide skins.

Directory specific images.

2. If there is a "directory skin", that is a skin of the same name as the directory, it is used in the generation of the documents within it, rather than the default skin, unless the document has a specific skin associated with it.

a. default skin (always available)

b. directory skin (precedence over default if exists)

c. document skin (takes precedence wherever document requests a specific skin)

Skins are defined in the document skin directory and if a directory association is desired a softlink made to the relevant skin. Skins (directory association auto load) auto load skin if a directory skin exists of same name as directory stub, (and there is no specific doc skin)

3. If the working directory has within it a sub-directory called image_local, the images within that directory are used for references to images, that are not part of the default site build.

1.10 Document Version Control Information

The possibility of citing an exact document version.

Permits the inclusion of document version control information to the document body and metatags.[39] This provides a much more certain method of referring to the exact version of a particular document, (assuming that the document is from a trusted source, that will retain earlier versions of a document).[40]

This information (where available) is provided under the section of the document titled "Document Information - MetaData", near the end of a document, for example in the segmented html version of this text at: <http://www.jus.uio.no/sisu/SiSU/metadata.html>

1.11 Table of contents

SiSU produces a rudimentary a table of contents based on document

[38] The previous way was directory associations for file output were set up in the configuration file. The present system is a more natural way to work requireing less configuration.

[39] from a version control system such as CVS

[40] The version control system must be run, so the version number is obtained, prior to the

headings.

1.12 Auto-numbering of headings

Headings can be automatically numbered, (and automatically named for hyper-linking)

1.13 Numbering and cross-hyperlinking of endnotes

SiSU can automatically number footnotes/endnotes. This is the default operation where no number is provided.

Footnotes/endnotes may also be manually numbered. Where a number, or numbers are provided for a footnote/endnote, this does not increment the automatic footnote/endnote number counter.

In the html output footnotes/endnotes are cross-hyper-linked (to their reference point and vice versa). In th pdf output footnotes are linked from their reference point only.

1.14 "Skinnable"

SiSU is skinnable, on a site-wide, directory-wide and per document basis, so different looking versions of things may be produced with little difficulty. There is a default skin which may be modified, as the background site skin, and each working directory may have a skin associated with it, as may each individual document. The hierarchy of application is document, directory, then site... ie if a document skin exists it gets precedence.

Whilst it is skinnable, the default output styles are selected to work across the widest possible range of document types.

1.15 Multiple Outputs

From markup that is simpler and more sparse than html you get:

- far greater output possibilities, including multiple html types, XML (different structured types), LaTeX (pdf landscape, portrait), and SQL (Postgresql or SQLite or other);
- the advantages implicit in these very different output possibilities;[41]
- a common citation system

As many output formats/presentations as one cares to write modules for - several types of html (e.g. structure based on css, or structure based on tables); *LaTeX/pdf* and *Lout/pdf*; pgsql other databases easily added; yaml...

1.15.1 html - several presentations: full length and segmented; css and table based

Most documents are produced in single and segmented html versions, described below:

The Scroll (full length text presentations)

The full length of the text in a single scrollable document.[42] As a rule the files they are saved in are named: *doc* or more precisely *doc.html*

SiSU document generation, and subsequent posting of the document.

[41] e.g. LaTeX (professional document typesetting, easy conversion to pdf or Postscript), XML (in this case, structural representation), SQL (e.g. document set searches; representation of the constituent parts of documents based on their structure, headings, chapters, paragraphs as desired; control of use)

[42] CISG <http://www.jus.uio.no/lm/un_contracts_international_sale_of_goods_convention_1980/doc>
The Unidroit Contract Principles <http://www.jus.uio.no/lm/unidroit.contract.principles.1994/doc>
or
The Autonomous Contract <http://www.jus.uio.no/lm/autonomous.contract.2000.amissah/doc>

For various reasons texts may only be provided in this form (such as this one which is short), though most are also provided as segmented texts.

"Scroll" is a reference to the historical scroll, a single long document/ parchment, and also no doubt to what you will have to do to get to the bottom of the text.[43]

The Segmented Text

The text divided into segments (such as articles or chapters depending on the text)[44] As a rule the files they are saved in are named: *toc* and *index* or more precisely *toc.html* and *index.html*

If you know exactly what you are looking for, loading a segment of text is faster (the segments being smaller). Occasionally longer documents such as the WTA 1994 <http://www.jus.uio.no/lm/wta.1994/toc> are only provided in segmented form.

Cascading Style Sheet, and Table based html

SiSU outputs html, two current standard forms available are:

css based

and

table based [largely discontinued][45]

The html is tested across several browsers

I like to remind you that there are other excellent browsers out there, many of which have long supported practical features like tabbing.

The html is tested across several browsers, including:

- **Firefox** (Mozilla-Firefox) [46]
- Kazehakase [47]
- Konqueror [48]
- Mozilla [49]
- MS Internet Explorer [50]
- Netscape [51]
- Opera [52]

Also lighter weight graphical browsers:

- Dillo [53]
- **Epiphany** [54]
- **Galeon** [55]

And for console/text browsing:

- **elinks** [56]
- **links2** [57]
- **w3m** [58]

[43] Scrolling is not however necessarily confined to full length documents as you will have to scroll to get to the bottom of any long segment (eg. chapter) of a segmented text.

[44] CISG <http://www.jus.uio.no/sisu/un_contracts_international_sale_of_goods_convention_1980>
The Unidroit Principles <http://www.jus.uio.no/lm/unidroit.contract.principles.1994>
The Autonomous Contract <http://www.jus.uio.no/sisu/the.autonomous.contract.2000.amissah> or WTA 1994 <http://www.jus.uio.no/lm/wta.1994>

[45] formatting possibility still exists in code tree but maintenance has been largely discontinuted.

[46] <http://www.mozilla.org/products/firefox/>

[47] <http://kazehakase.sourceforge.jp/>

[48] <http://www.konqueror.org/>

[49] <http://www.mozilla.org/>

[50] <http://www.microsoft.com/windows/ie/default.asp>

[51] <http://home.netscape.com/comprod/mirror/client_download.html>

[52] <http://www.opera.com/>

[53] <http://www.dillo.org/>

[54] <http://www.gnome.org/projects/epiphany/>

[55] <http://galeon.sourceforge.net/>

[56] <http://elinks.or.cz/>

[57] <http://links.twibright.com/>

The html tables output is rendered more accurately across a wider variety set and older versions of browsers (than the html css output).

1.15.2 XML

SiSU generates well formed XML, and multiple versions. An XML SAX version with a flat/shallow structure, and XML DOM version with a deeper (embedded) structure. There is also a released working xhtml module. Examples of SAX and DOM versions are provided within this document.

1.15.3 ODT:ODF, Open Document Format - ISO/IEC 26300:2006

SiSU generates Open Document Output format.

1.15.4 PDF - portrait and landscape, (through the generation of LaTeX output which is then transformed to pdf)

SiSU outputs LaTeX if required which is easily transformed to PDF.[59] PDF documents are generated on the site from the same source files and **Ruby** program that produce html. Landscape oriented pdf introduced, providing easier screen viewing, they are also (paper saving, being currently) formatted to have fewer pages than their portrait equivalents.

- Adobe Reader [60]
- **Evince** [61]
- xpdf [62]

1.15.5 Search - loading/populating of relational database while retaining document structure information, object citation numbering and other features (currently PostgreSQL and/or SQLite)

SiSU (from the same markup input file) automatically feeds into PostgreSQL[63] and/or SQLite[64] database (could be any other of the better relational databases)[65] - together with all additional information related to document structure, and the alternative ways in which it is generated on the site retained. As regards scaling of the database, it is as scalable as the database (here Postgresql or SQLite) and hardware allow. I will prune the images later.

This is one of the more interesting output forms, as all the structural data for the documents are retained (though can be ignored by the user of the database should they so choose). All site texts/documents are (currently) streamed to four pgsql database tables:

- one containing semantic (and other) headers, including, title, author, subject, (the Dublin Core...);
- another the substantive texts by individual “paragraph” (or object) - along with structural information, each paragraph being identifiable by its paragraph number (if it has one which almost all of them do), and the substantive text of each paragraph quite naturally being

[58] <http://w3m.sourceforge.net/>

[59] LaTeX and pdf features introduced 18th June 2001, Landscape and portrait pdfs introduced 7th October 2001., Lout is a more recent addition 22th April 2003

[60] <http://www.adobe.com/products/acrobat/readstep2.html>

[61] <http://www.gnome.org/projects/evince/>

[62] <http://www.foolabs.com/xpdf/>

[63] <http://www.postgresql.org/>
<http://advocacy.postgresql.org/>
<http://en.wikipedia.org/wiki/Postgresql>

[64] <http://www.hwaci.com/sw/sqlite/>
<http://en.wikipedia.org/wiki/Sqlite>

[65] Relational database features retaining document structure and citation introduced 15th July 2002

searchable (both in formatted and clean text versions for searching); and

- a third containing endnotes cross-referenced back to the paragraph from which they are referenced (both in formatted and clean text versions for searching).

- a fourth table with a one to one relation with the headers table contains full text versions of output, eg. pdf, html, xml, and ascii.

There is of course the possibility to add further structures.

At this level **SiSU** loads a relational database with documents broken in to their smallest logical structurally constituent parts, as text objects, with their object citation number and all other structural information needed to construct the structured document. Text is stored (at this text object level) with and without elementary markup tagging, the stripped version being so as to facilitate ease of searching.

Because the document structure of sites created is clearly defined, and the text object citation system is available for all forms of output, it is possible to search the sql database, and either read results from that database, or just as simply map the results to the html output, which has richer text markup.

The combination of the **SiSU** citation system with a relational database is pretty powerful, giving rise to several possibilities. As individual text objects of a document stored (and indexed) together with object numbers, and all versions of the document have the same numbering, complex searches can be tailored to return just the locations of the search results relevant for all available output formats, with live links to the precise locations in the database or in html/xml documents; or, the structural information provided makes it possible to search the full contents of the database and have headings in which search content appears, or to search only headings etc. (as the Dublin Core is incorporated it is easy to make use of that as well).

This is a larger scale project, (with little development on the front end largely ignored), though the "infrastructure" has been in place since 2002.

1.15.6 Search - database frontend sample, utilising database and SiSU features, including object citation numbering (backend currently PostgreSQL)

Sample search frontend [66] A small database and sample query front-end (search from) that makes use of the citation system, object citation numbering to demonstrates functionality.[67]

SiSU can provide information on which documents are matched and at what locations within each document the matches are found. These results are relevant across all outputs using object citation numbering, which includes html, XML, LaTeX, PDF and indeed the SQL database. You can then refer to one of the other outputs or in the SQL database expand the text within the matched objects (paragraphs) in the documents matched.

(further work needs to be done on the sample search form, which is rudimentary and only passes simple booleans correctly at present to the SQL engine)

A few canned searches, showing object numbers. Search for:

English documents matching Linux OR Debian

GPL OR Richard Stallman

invention OR innovation in English language

copyright in English language documents

Note that the searches done in this form are case sensitive.

[66] <http://search.sisudoc.org>

[67] (which could be extended further with current back-end). As regards scaling of the database, it is as scalable as the database (here Postgresql) and hardware allow.

Expand those same searches, showing the matching text in each document:

English documents matching Linux OR Debian

GPL OR Richard Stallman

invention OR innovation in English language

copyright in English language documents

Note you may set results either for documents matched and object number locations within each matched document meeting the search criteria; or display the names of the documents matched along with the objects (paragraphs) that meet the search criteria.[68]

OCN index mode, (object citation number) the numbers displayed are relevant (and may be used to reference the match) in any sisu generated rendition of the text[69] the links provided are to the locations of matches within the html generated by **SiSU** .

Paragraph mode, you may alternatively display the text of each paragraph in which the match was made, again the object/paragraph numbers are relevant to any **SiSU** generated/published text.

Several options for output - select database to search, show results in index view (links to locations within text), show results with text, echo search in form, show what was searched, create and show a "canned url" for search, show available search fields. Also shows counters number of documents in which found and number of locations within documents where found. [could consider sorting by document with most occurrences of the search result].

[68] of this feature when demonstrated to an IBM software innovations evaluator in 2004 he said to paraphrase: this could be of interest to us. We have large document management systems, you can search hundreds of thousands of documents and we can tell you which documents meet your search criteria, but there is no way we can tell you without opening each document where within each your matches are found.

[69] OCN are provided for HTML, XML, pdf ... though currently omitted in plain-text and opendocument format output

Earlier version of the search frontend - Simple search, results with files in which search found, and locations where found within files.

Simple search, results with files in which search found, and text object (paragraph or endnote) where found within files.

1.15.7 Other forms

There are other forms as well, YAML file, **Ruby** Marshal dumps, document pre-processing (processing of documents prior to the steps described here, to produce input suitable for the program) snap in a new module as required/desired, well formed XML, no problem.

1.16 Concordance / Word Map or rudimentary index

Concordance /WordMaps:[70] **SiSU** produces a rudimentary index based on the words within the text, making use of paragraph numbers to identify text locations. This is generated in html and hyper-linked but identifies these words locations in the other document formats. Though it is possible to search using a search engine, this is a means for browsing an alphabetical list of words which may suggest other useful content.

1.17 Managed (document) directory, database, or site structure

SiSU builds the web site (or more generically provides a suitable directory structure) - placing various output texts in the hierarchy of the website (or db), which (for directories) is a sub-directory with the name of the text file.

1.18 Batch processing

SiSU is a batch processing tool, handling and transforming multiple (or individual) documents (in many ways) with a single instruction.

1.19 Integration to superior Gnu/Linux and Unix tools

As should have been noted by the above description of **SiSU** , it makes use of existing programs found on **Gnu** /Linux and Unix, amongst those already mentioned include the LaTeX to pdf converters and the database PostgreSQL or SQLite.

1.19.1 Backup and version control

Unix provides many tools for version control. For documents Subversion, CVS and even the old RCS are useful for the per-document histories they provide.

For writing code superior (more recent) version control system exist. These can also be used for documents though they tend to take stamps of changes across the repository as a whole, rather than for each individual file that is tracked, (as CVS and RCS do). My personal preference is for distributed systems such as Git, Mercurial or Darcs, of which I use Git for both code and documents.

Several backup tools exist. At the base level I tend to use rdiff.

1.19.2 Editor support

SiSU documents are prepared / marked up in utf-8 text you are free to use the text editor of your choice.

Syntax highlighting for a number of editors are provided. Amongst them Vim, Kwrite, Kate, Gedit and diakonos. These may be found with configuration instructions at <http://www.jus.uio.no/sisu/syntax_highlight>. Vim [71] as of version 7 has built in sytax highlighting for **SiSU** .

1.20 Modular design, need something new add a module

Need a new output format that does not already exist, write a new module.

Prefer a new input syntax, you could write a new syntax matching the existing design, though my personal preference is some uniformity in entry appearance. If necessary has been fairly easy to extend the design parameters. It is intended to incorporate some additional basic semantic tagging, (book, article, author etc.) However, keeping the requirements for input minimal, and relatively simple has been a design goal.

[70] Concordance/ WordMaps introduced 15th August 2002

[71] <http://www.vim.org/>

DOCUMENT INFORMATION (METADATA)

Generated by: SiSU 0.59.0 of 2007w38/0 (2007-09-23)

Ruby version: ruby 1.8.6 (2007-06-07 patchlevel 36) [i486-linux]

Metadata

Document Manifest @

<http://www.jus.uio.no/sisu/sisu_manual/sisu_description/sisu_manifest.html>

Dublin Core (DC)

DC tags included with this document are provided here.

DC Title: SiSU - Description

DC Creator: Ralph Amissah

DC Rights: Copyright (C) Ralph Amissah 2007, part of SiSU documentation, License GPL 3

DC Type: information

DC Date created: 2002-11-12

DC Date issued: 2002-11-12

DC Date available: 2002-11-12

DC Date modified: 2007-08-30

DC Date: 2007-08-30

Version Information

Sourcefile: sisu_description.sst

Filetype: SiSU text 0.57

Sourcefile Digest, MD5(sisu_description.sst)= b89ccdad9f6d9c2260d8d383d6b35ccc

Skin_Digest: MD5(/home/ralph/grotto/theatre/dbld/sisu-dev/sisu/data/doc/sisu/sisu_markup_samples/sisu_manual/_sisu/skin/doc/skin_sisu_manual.rb)= 20fc43cf3eb6590bc3399a1aef65c5a9

Generated

Document (metaverse) last generated: Mon Sep 24 15:34:36 +0100 2007

Information on this document copy and an unofficial List of Some web related information and sources

”Support Open Standards and Software Libre for the Information Technology Infrastructure” RA

Information on this document copy www.jus.uio.no/sisu/

W3 since October 3 1993 SiSU 1997, current 2007.
SiSU presentations at **www.jus.uio.no/sisu/**

SiSU **pdf** versions can be found at:
http://www.jus.uio.no/sisu/sisu_description/portrait.pdf
http://www.jus.uio.no/sisu/sisu_description/landscape.pdf

SiSU **html** versions may be found at:
http://www.jus.uio.no/sisu/sisu_description/toc.html or
http://www.jus.uio.no/sisu/sisu_description/doc.html

SiSU Manifest of document output and metadata may be found at:
http://www.jus.uio.no/sisu/sisu_description/sisu_manifest.html

SiSU found at: **www.jus.uio.no/sisu/**

Links that may be of interest at SiSU and elsewhere:

SiSU Manual
http://www.jus.uio.no/sisu/sisu_manual/

Book Samples and Markup Examples
http://www.jus.uio.no/sisu/SiSU/2.html

SiSU @ Wikipedia
http://en.wikipedia.org/wiki/SiSU

SiSU @ Freshmeat
http://freshmeat.net/projects/sisu/

SiSU @ Ruby Application Archive
http://raa.ruby-lang.org/project/sisu/

SiSU @ Debian
http://packages.qa.debian.org/s/sisu.html

SiSU Download
http://www.jus.uio.no/sisu/SiSU/download.html

SiSU Changelog
http://www.jus.uio.no/sisu/SiSU/changelog.html

SiSU help
http://www.jus.uio.no/sisu/sisu_manual/sisu_help/

SiSU help sources
http://www.jus.uio.no/sisu/sisu_manual/sisu_help_sources/

SiSU home:
www.jus.uio.no/sisu/