# Moses2

*M.T. Carrasco Benitez*
*March 2013, version 0.11*

# 1. Summary

This document gathers information to facilitate the eventual rewriting [SP] of Moses [MOSES] and associated programs such as Giza [GIZA]; indeed, in the context of this document, Moses2 refers to Moses itself and **all the programs** needed to have a full open and free statistical machine translation system (SMT) .

Creating an open and free SMT system based in the current state of the art is mostly an organisational endeavour, as opposed to an engineering challenge. Hence, the rationale is to **create a permanent organisation** that will develop an SMT system that is **valid for research and industrial** use, so everybody has an interest in building a proper system. The **present SMT community should be fully involved**; indeed the first intention of this document is to gather the rough consensus of the SMT community, in particular on what has to be done with the different modules: re-use, re-write or extend by developing from scratch.

## 1.1. Main aspects

- **Organisational aspects**
  - **Foundation**: create a permanent legal entity
  - **Research and industrial**: programs valid for **both** worlds
  - **Whole cycle**: place in the context of *authoring, translation and publishing*
  - **Documentation**: a project similar to *The Linux Documentation Project*
- **Standards**
  - Running environment
  - Data structures
- **Technical aspects**
  - **Installation packages**: at least for Linux (Debian, Ubuntu, Redhat), OSX, Windows
  - **Automatic testing**: similar to Python
  - **Libraries**: similar to *libxml2*
  - **Virtual data layers**: similar to the virtual file system in Linux [VFS]

# 2. Foundation

This must be a **permanent non-profit** organisation for Moses2, where the SMT is only the first project; other projects could come later. Indeed, data preparation could be a separated project because it is common to other language technology fields.

The Foundation could be quite sparse: just a legal entity with the required minimal attributes such as one director, management board and a postal address; it would not required a physical building and the collaborators could be employed and paid by other entities. It would need a website.

Translation stakeholders such as the Directorate-General for Translation [DGT] of the European Commission should get involved: the organisational aspects are as important as the system architecture. One should be aiming to create an ecosystem similar to Apache [AF] or Linux [LF].

It makes sense to pool resources for an infrastructure project such as SMT, as practically any organisation (public or private) would struggle to get the right people to develop such a project internally.

### 2.2. Resources

The stakeholders should contribute with people, machines and monies. The seed monies could be public, though eventually the Foundation must self-finance. For-profit businesses could develop, for example, like in the case of PHP and Zend. From the European Commission point of view, one has to see which financing instrument is the most appropriate.

Individual contributors will write free software. But one must be able to award a small contract to an individual to rewrite a module; monies must be available.

### 2.3. Names

It is essential getting the right names from the very beginning. Moses2 refers to the new version of SMT: the present Moses proper and associated programs, though one might decide on a different name. The Foundation must have a different name; for example, the Language Technology Foundation. Also:

- One must be able to register the trademark. The same for the logo.
- The domain name must be free.
- The name must have none or low Google count.

## 3. Current system

The current Moses system is the base of many different projects and systems: academic and commercial. One could say that it has been over-successful.

## 4. Authoring, Translation, Publishing, Chain (ATP-chain)

Though the main intention is to address SMT, one has to be aware that translation (human or machine) is only one of the phases in the production of *multilingual parallel texts*; indeed, SMT is one of several techniques of machine translation (MT). The main phases in the production of multilingual parallel texts are:

- **Authoring**
- **Translation**: processing common for CAT and MT
    - Human translation: computer-aided translation (CAT)
    - Machine translation (MT)
        - Translation memories (TM): a form of machine translation
        - Statistical machine translation (SMT)
        - Rule-based machine translation (RBMT)
- **Publishing**

This is an Authoring, Translation, Publishing, Chain (ATP-chain). The key factor is **interoperability**: one needs to standardise how to create *linguistic pipes* (extending the meaning from Unix).

# 5. Development

## 5.1. Programming philosophy

It should be the Unix philosophy [AUP]. The code must be highly **portable**: Debian could be the primary reference operating system, though one has to take into account POSIX, Unix in general, OSX and Windows; i.e., the programs must be as independent as possible from the underlying operating system and the specificities must be fully documented. The programs must be available as installation packages for other Linux, OSX and Windows.

## 5.2. Development

One should use the usual techniques:

- Source control system; e.g., git
- Bug tracking system
- Complete binary distributions, at least for the following operating systems:
    - Ubuntu: this implies having the Debian structure
    - OSX
    - Windows

## 5.3. The environment

Convention in the file system, program names, etc.

### 5.3.1 Program names

There should be a list of programs mirroring the required functionalities for SMT; each program must have a man page. The name convention should be similar to `git`; e.g., `foo-myprog`, `foo-yourprog`.

## 5.4. Programs, libraries and modules

The programs should be well-behaved Unix programs; not only for obvious aspects such as using the standard input (`stdin`), but also for aspects such as `syslog` for logs. Daemons should be properly designed to function without an attached tty, such as managing the output. When appropriate, parameters should be URIs as opposed to files; e.g., `http://example.com/input.txt`. Also, the concerned programs must support several algorithms, similar to the approach of the compression or encrypting programs.

Commands must be in the user's path. Either in a directory such as `/usr/local/bin` or in a private `bin` directory such as `/opt/foo/bin`.

The code must be highly portable and it must be able to bind to different programming languages (e.g., C, C++, Python, PHP) and to facilitate the construction of an Apache module (e.g., `mod_foo`). For example, the style of *libxml2* [LX]. One should consider having the functionalities in a library with an appropriate name (e.g., `libfoo.a`, `libfoo.so`) placed in a location such as `/usr/local/lib`.

There should be monitoring techniques for programs that run for a long time, so one knows what is going on; e.g., if the program is stuck somewhere.

Programs must not have a file extension; e.g., `foo` and not `foo.perl`. The rationale is that a first version might be an interpreted script and one could rewrite it in a compilable language such as C. Files with data or source code should have file extensions; e.g., `foo.txt`, `foo.en.txt`, `foo.c` (`foo.c` would be compiled into `foo`).

## 5.5. Data

### 5.5.1 Data characteristics

The main characteristics of SMT are:

- **Dataset**: The data constitute a complex dataset; e.g., corpora, models, etc.
- **Big**: The source data and the intermediary steps could jump into the terabytes.

### 5.5.2 Multilingual Dataset Format (muset)

A format for packing all types of multilingual datasets, such as corpora or translation memories [MUSET]. The rationale is to have a format that can be used directly without any preparation or cleaning. This is mature and practically ready

### 5.5.3 Multilingual Dataset Toolbox (mux)

A collection of programs for the processing of multilingual datasets.

### 5.5.4 Language Interoperability Portfolio (Linport)

This is the dossier for the translator. This is under development.

### 5.5.5 Big data

Economise every possible bit of data.

### 5.5.6 Virtual data layer

This is a similar concept to the *virtual file system* [VFS] in Linux that allows the implementation of many real file systems (e.g., ext3, ext4). Hence, the programs should access the data using this data abstraction and at compiling time different technologies could be chosen. For example, SQLite, Berkeley DB or perhaps a bespoke data technique maximised for SMT.

#### 5.5.7 File extension for data

The data files must have the appropriate file last extension to trigger the associated program. For example `foo.txt, foo.en.txt, foo.en.txt.zip`.

### 5.6. Development phases

First phase, adapt the current Moses to the new environment; i.e., just minimal housekeeping changes.

## 6. Documentation

A project within the Foundation must be dedicated to this purpose, similar to The Linux Documentation Project [LDP].

## 7. Terminology

Language technology. RFC2119.

## 8. Miscellaneous

### 8.1. Status of this document

Work in progress. Send comments to the author.

### 8.2. Acknowledgement

Hilario Fontes is a general consultant. Andreas Eisele suggested the automatic testing.

### 8.3. Disclaimer

This document represents only the views of the author and not necessarily the views of any other parties. In particular, it does not necessarily represent the opinion of the European Commission.

### 8.4. Author

Manuel Tomas CARRASCO BENITEZ
Directorate-General for Translation
European Commission
L-2920 Luxembourg

Telephone: +352 4301 36943
`Manuel{DOT}Carrasco-Benitez@ec{DOT}europa{DOT}eu`

# 9. References

[AUP] The Art of Unix Programming
http://www.catb.org/~esr/writings/taoup

[LDP] The Linux Documentation Project
http://www.tldp.org/index.html

[LX] libxml2
http://xmlsoft.org

[MW] Mediawiki Collection
http://www.mediawiki.org/wiki/Extension:Collection

[MOSES] Moses
http://www.statmt.org/moses

[MUSET] Multilingual Dataset Format
http://dragoman.org/muset

[PHPDOC] PHP documentation
http://doc.php.net

[RFC2119] Key words for use in RFCs to Indicate Requirement Levels
http://www.ietf.org/rfc/rfc2119.txt

[SMT] Statistical Machine Translation
http://www.statmt.org

[SP] Straw poll on the re-writing of Moses
http://dragoman.org/strawpoll.pdf

[VFS] Anatomy of the Linux file system
http://www.ibm.com/developerworks/linux/library/l-linux-filesystem

**Ω**