Samstarf um íslenska máltækni (SÍM)
Icelandic Language Technology Consortium

DRAFT

Standards and Guidelines for Software Development and Delivery Processes

Version 0.4
2020-01-06

Introduction

This document is created as a part of SÍM's implementation plan for the *Language Technology for Icelandic 2018-2022* project, as commissioned by Almannarómur for the Icelandic Ministry of Education, Science and Culture (MESC). It describes unified standards and guidelines for software development and delivery processes that will be used within SÍM in the implementation of its projects.

The document was originally drafted by a three-member panel, in coordination with the SÍM project manager, and then submitted for comments and suggestions to the wider SÍM group. It was adopted as a SÍM-wide standard on 2020-01-XX.

Goals

The goal of this document is to describe, clearly and concisely, the common standards and guidelines that will be applied during the development of software within SÍM and when delivering the software to the CLARIN-IS repository/ Almannarómur on behalf of the MESC for open release to the public.

The standards and guidelines are intended to help ensure, as far as possible, that the delivered software:

- meets the agreed requirements, in letter and in spirit,
- is open and under the appropriate licenses,
- is thoroughly tested,
- has sufficient performance and other characteristics appropriate for commercial use,
- can be easily installed and works on major server and client platforms,
- is well documented,
- is maintainable,
- is aligned with generally accepted language technology (LT) and industry standards,
- is compatible and interoperable both with other SÍM-developed tools and with other common LT software.

Structure

This document first describes general aspects of software development processes within the SÍM projects, i.e. the aspects that apply to all projects regardless of the intended target audience or delivery platforms.

Subsequently, specific aspects that relate to individual target audiences and/or platforms are described in separate chapters.

To recap, SÍM delivery formats specified in the technology description document are as follows:

- Stand-alone executable applications (APP)

- Modules that can be embedded in other applications (MOD)
- Add-ons to existing platforms (ADD-ON)
- Web-based services, i.e. websites and/or HTTP APIs (WEB)
- Language resource files (RES)
  Note that standards for language resource files are defined in a separate project and are outside the scope of this document. Any resource files included with SÍM software deliverables should conform to these standards.

1. General Aspects

This chapter describes standards and guidelines that apply to all SÍM software development projects, regardless of target audience or delivery platform.

1.1. Text Encoding

SÍM source code, as well as text-based configuration and resource files, should be encoded in UTF-8.

SÍM software shall be able to handle input text using all valid Unicode code points, and pass all valid Unicode code points through to its output, as applicable.

1.2. Natural Language

SÍM software should be written in U.S. English, using English-language identifiers and comments. Icelandic (or other non-English) text within source code files should be kept to a minimum, with a preference for storing such text in external configuration files or databases.

An exception may be made where there is a direct and logical correspondence between Icelandic language constructs and the source code in question, for instance when naming functions that handle particular grammatical constructs, words or phrases.

1.3. Licensing

All source code developed within the SÍM projects should be licensed according to the Apache 2.0 license or a compatible license that is equally or more permissive (such as the MIT license).

A notice to this effect should be included in a header comment in all source files, alongside a standard copyright notice that identifies the original copyright holder to whom an attribution should be made in derivative works.

Source code repositories should include the applicable license and clearly state the licensing scheme in README files.

1.4. Supporting Software and Tools

The requirements of clause 1.3 mean that any third party software or other resources included in SÍM deliverables or specifically depended upon at run-time must also be available under the Apache 2.0 license, or a compatible license that is equally or more permissive.

Products of the SÍM projects should thus not require third-party proprietary or closed software to be present in the deployment environment, apart from the host operating system and its standard software environment.

If third-party database engines or other supporting tools are required, they should be well-known, open, available without charge on all intended deployment platforms, and actively maintained.

1.5. Version Control

SÍM development projects should use Git as a source code repository and version control system.

Projects can be hosted as Git remotes on GitHub or GitLab.

Developers should commit and push changes frequently to the Git remote, with a preference for multiple smaller commits over few large ones.

Individual commits should be described with meaningful comments.

New functionality under development should be bundled within dedicated Git branches and submitted as a pull request to remote/master after having been tested and reviewed.

It is preferred to use public repositories, and in any case repositories will be made public no later than upon delivery of the software to CLARIN/Almannarómur. However, repositories that are private to the SÍM consortium can be used during early development phases if public disclosure at that point is deemed by the development team to be premature.

For peer review and quality control reasons, repositories should not be exclusive to a single entity or to a subset of entities within SÍM.

1.6. Test-Driven Development

Unit tests for new or modified functionality should, as far as possible, be written in parallel with the development of the functionality itself.

Unit tests should be included in source code repositories and be considered to form an integral part of the delivered products.

Standard unit test frameworks should be used, such as PyTest for Python and Catch2 for C++.

1.7. Continuous Testing and Integration

SÍM development projects should be configured with continuous testing and integration (CT/CI) using standard tools (such as GitLab or Travis CI). This means

that either upon every commit to a Git remote branch, or at fixed time intervals (preferably at least once a day), the software is automatically built into a deliverable package and a suite of tests is run, on both individual modules/components and on the entire package, as applicable.

1.8. Programming Languages

As far as possible, source code in the SÍM project should be written in the following standard programming languages:

- Python 3.6 (executable on CPython and PyPy)
- C++17 (ISO/IEC 14882:2017)
- HTML 5
- ECMAScript 2015 (ES6, JavaScript)
- CSS 2.1

It is permitted to use tools that add or mix annotations on top of the above syntaxes, for instance HTML templating engines, JSX and Sass/Less.

If mixing Python and C++, CFFI or cppyy are preferred as the communication mechanisms between the two.

Other standard, mainstream programming languages may be used in SÍM projects if required by the host platform, for instance on mobile devices. When choosing a programming language, openness, standardization, interoperability, availability of knowledgeable programmers in Iceland, educational resources, community support and other relevant factors shall be considered.

1.9. Coding Guidelines

Common, generally accepted coding guidelines and styles should be applied to source code that is developed as a part of the SÍM projects. The use of automatic formatters, with common default settings, is encouraged.

For specific programming languages, the following are mentioned as examples:

Python	PEP-8, Black formatter
C++	C++ Core Guidelines, Google C++ Style Guide
ECMAScript	AirBnB JavaScript Style Guide

Source code, including HTML and CSS files, should preferably be indented using spaces rather than tabs. Indentation should be consistent.

1.10. Comments

Source code should be liberally commented in succinct and correct U.S. English.

All source code files should start with a header comment describing the file and its contents, including a copyright notice and a reference to the applicable software license (cf. section 1.3).

Individual classes, functions, global constants and variables should be described in comments. Within blocks of code, the intent of each section should be described. It is not necessary or encouraged, however, to describe in comments what is already and directly apparent from the code itself.

1.11. Delivery

All public deliverables of the SÍM project, both initial releases and any subsequent updates, should be delivered to the CLARIN-IS repository ERIC (https://repository.clarin.is/repository/) for hosting and publication. Such delivery and accompanying metadata shall be according to CLARIN's requirements and specifications. Delivery will typically include at least a zip and/or tar.gz archive of a complete git repository snapshot (release), containing all accompanying documentation, resources, build/make and test files as well as the source code of the software.

It is recommended that a delivery package include Docker and, as the case may be, Docker Compose files that illustrate how all necessary components are assembled from well-known base images into a final executable image that passes all applicable tests when run on appropriate hardware.

A delivery package should include detailed and easily reproducible configuration, setup and installation instructions for each delivery platform. Such instructions are considered to be implicitly provided by Docker/Docker Compose files, if present.

2. Executable Applications (APP)

This chapter describes standards and guidelines that apply in particular to executable, stand-alone applications.

2.1. Platforms

Executable applications shall be tested and delivered on the following operating system platforms:

Linux (Ubuntu 18.04 LTS x86_64)

MacOS (Mojave, version 10.14.6, or newer)

Windows 10, 64-bit

2.2. Installation

Executable applications shall be installable and uninstallable using standard tools on each OS platform.

3. Embeddable Modules (MOD)

This chapter describes standards and guidelines that apply in particular to modules that are intended to be embedded into other software.

3.1. Platforms

Embeddable modules shall be tested and delivered on the following operating system platforms:

Linux (Ubuntu 18.04 LTS x86_64)

MacOS (Mojave, version 10.14.6, or newer)

Windows 10, 64-bit

3.2. Installation

Embeddable modules shall be packaged in such a manner as to be installable and uninstallable using standard tools for each combination of OS platform and programming language.

Python packages should be installable using `pip`, and preferably hosted on the Python Package Index (PyPI).

C/C++ packages should typically be compiled and linked into position independent code (PIC) shared object libraries (`.so`) files on Linux, and comparable delivery formats for Windows (DLL and/or LIB files) and MacOS.

Embeddable modules may require a set of standard packages from the operating system distribution to be present on the host system. To the extent possible, installation of the embeddable modules should cause such packages to be installed automatically. If that is not feasible, the required packages on each platform should be clearly specified in the module documentation, along with instructions on how to install them in a typical scenario. Installation should preferably only require common OS package installation commands, such as `apt/apt-get` for Ubuntu/Debian or

`brew` for MacOS, eventually invoked via `sudo`. If multiple installation steps are required, they should preferably be packaged into shell scripts for each platform.

4. Add-Ons to Existing Platforms (ADD-ON)

This chapter describes standards and guidelines that apply in particular to add-on modules for existing platforms.

4.1. Platforms

Add-ons to existing platforms shall be delivered in such format and packaged in such a way as is standard and normal for the particular platform.

5. Web-Based Services (WEB)

This chapter describes standards and guidelines that apply in particular to functionality that will be delivered as Web-Based Services.

5.1. Platforms

Web-based services should be hostable on web server platforms that are well-known, proven for production use, open, and free. They should at a minimum run on Linux-based servers, such as Ubuntu 18.04 LTS x86_64.

The nginx and Apache web servers are preferred as test and delivery reference platforms.

SÍM test and reference Web services as delivered shall include necessary configuration files for the web server.

Delivery in the form of Docker containers that are compatible with the above platforms is considered equivalent to delivery on the platform itself.

5.2. Security

Test and reference Web services delivered by SÍM should use SSL/TLS-based protocols (such as HTTPS) and be configured to receive at least an “A” rating on the Qualys® SSL Labs HTTPS security test suite.

5.3. Protocols

Web services should use HTTPS as their request/response protocol, with REST, JSON and XML as the preferred higher-level protocols and data formats. Other W3C standard protocols, such as WebSockets (over TLS), can also be used when appropriate.

Request and response text shall be encoded in UTF-8.

5.4. Configuration and Tuning

It is up to the parties deploying the Web-Based Services to further configure their server or server cluster for the required performance characteristics, such as via load balancing, caching, container management frameworks, and/or other means.

Web-Based Services delivered from the SÍM project shall be designed and implemented in a manner as to not unduly hinder or prevent such configurations or such tuning.

6. Language Resource Files (RES)

Standards for language resource files are defined in a separate SÍM project and are outside the scope of this document.

Appendix I: Abbreviations

SÍM	Samstarf um íslenska máltækni *Icelandic Language Technology Consortium*
MESC	Ministry of Education, Science and Culture
LT	Language Technology
UTF-8	Unicode Transformation Format, using 8-bit representation
PEP-8	Python Enhancement Proposal 8 *A coding style standard for Python*
CLARIN	Common Language Resources and Technology Infrastructure
TLS	Transport Layer Security
SSL	Secure Sockets Layer
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol, over SSL/TLS
REST	Representational State Transfer
JSON	JavaScript Object Notation
XML	Extended Markup Language
W3C	Word Wide Web Consortium
OS	Operating System
DLL	Dynamic Link Library
LIB	Object code library file
LTS	Long-Term Support
PIC	Position Independent Code