|  |
| --- |
| Fontys University of Applied Sciences - ICT |
| Architecture Document |
| Amplexor Group Project – Group 3 |

|  |
| --- |
| Group 3 (Wait 4 IT) – Release Version  21 January 2021 |

Table of Contents

[1. Choice of architecture model specification 2](#_Toc62095726)

[2. Architecture models 3](#_Toc62095727)

[1.1 System Context (C1) 3](#_Toc62095728)

[1.2 Containers and Technology (C2) 4](#_Toc62095729)

[1.3 Components (C3) 5](#_Toc62095730)

[1.4 Code (C4) 6](#_Toc62095731)

# Choice of architecture model specification

There are several rather good graphical notations out there that serve the purpose of illustrating a project’s software structure. However, we as a team have decided to use the C4 model specification as it provides a well-balanced overview of our software system by dividing it into several levels and thus introducing multiple points of views to the reader. The C4 model separates the system in question into containers and components and the relationship between them is made clear using graphical models.

This document will provide an architecture model for each of the viewpoints and will be organized according to the following hierarchical level:

* **System context (Level 1)** – Level 1 diagrams show the global scope of the software system and its communication with the end-users and other systems.
* **Container context (Level 2)** – Level 2 diagrams decompose the main part of the software into several containers that showcase the workflow of the main subsystems.
* **Component context (Level 3)** – Level 3 diagrams decompose the higher level diagram even further, showing the relationship between the components that form the abovementioned subsystems.
* **Code context (Level 4)** – Level 4 diagrams provide the lowest level of information for each component – it’s code. The code context is often expressed using other graphical notations. In this case, we’ll do so using a Unified Modelling Language (UML), generated by our Integrated Development Environment (IDE) of choice: Visual Studio.

# Architecture models

## System Context (C1)

As mentioned above, the system context provides an explanation about the communication between the software system, its end users and external systems.

In this particular case, the end-user is an individual that uses the SDL Trados Studio application with our software solution (a translation assistant plugin) installed. The mainframe of the translation system is the plugin in question that was developed by our team.

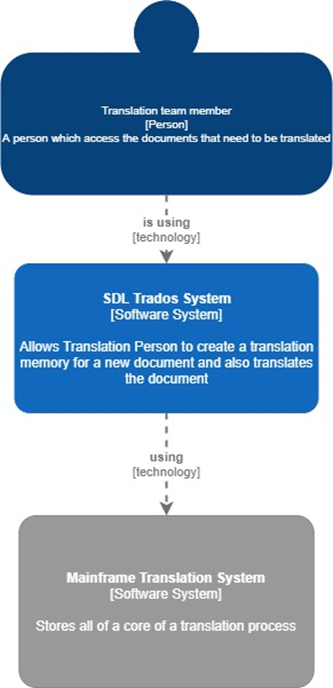


Figure - System Context (C1)

## Containers and Technology (C2)

The containers and technology context provides a clear insight into the internal communication between the SDL Trados Studio translation application and our plugin. This is made possible via an API framework that is provided to all developers who have access to the SDL Trados Studio software development kit (SDK).

Additionally, an external communication channel is established with the public EUR-Lex database to obtain requested document data. This channel was developed by the team and is exclusively part of the plugin’s software. In other words, it does not communicate in any way with SDL Trados.

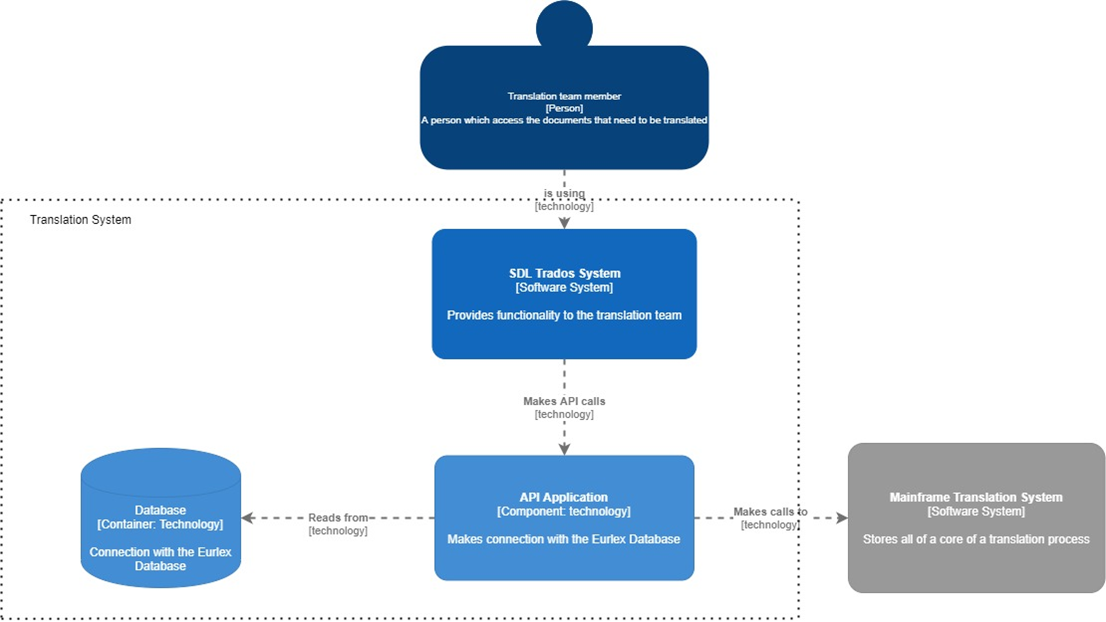


Figure - Containers & Technology (C2)

## Components (C3)

The components context dives in detail into our plug-in. Several components form the main functionality of the software solution (each component is ordered logically as if they were being used to execute the main process):

* A reference number extraction tool that goes through target SDL Trados files to locate and extract reference numbers of European documents. For that, a file reader is required. One was developed by the team to serve that purpose.
* A Simple Object Access Protocol (SOAP) communication handler that requests CELEX numbers using the found reference numbers. These CELEX numbers are needed to retrieve the required documents from the EUR-Lex database.
* A communication layer with the EUR-Lex database. Once a CELEX number has been obtained by the aforementioned component/subsystem, requested document data is retrieved in the form of HTML code and handled by the other components in the software.
* A document processing unit that handles the HTML code and structures it in the form of a translation memory. This processing unit checks for inconsistencies and errors that could occur if the source and the target language version of the documents do not follow the same structure.

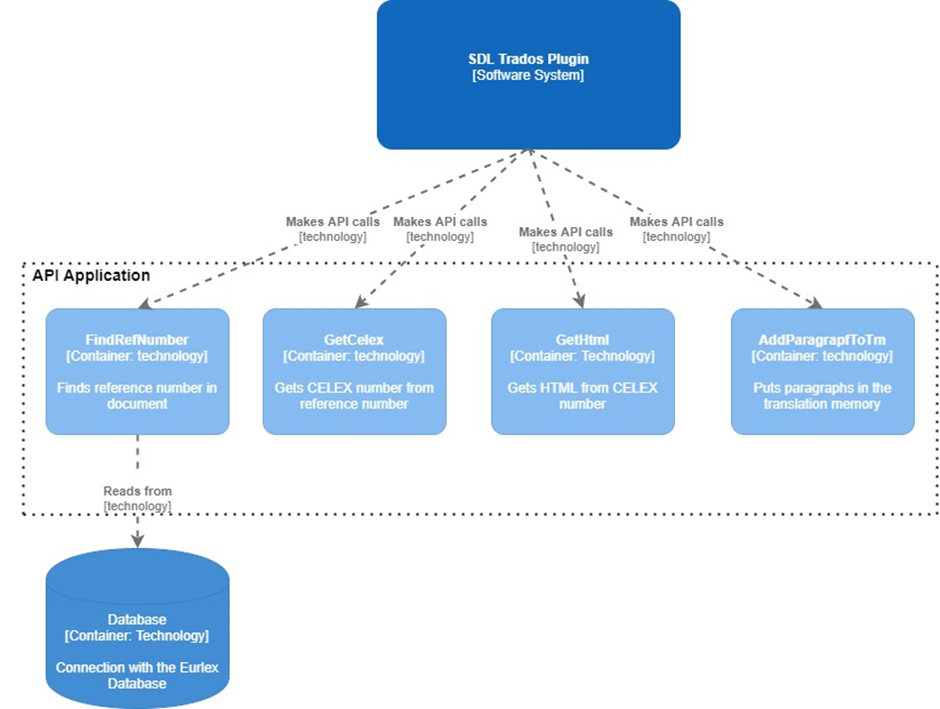


Figure - Components (C3)

## Code (C4)

The code context of the SDL Trados plugin can be segmented into three separate sections:

* **Translation memory creation & export tool (Translation memory batch task)** - This is the process that extracts reference numbers and structures a translation memory out of retrieved document data.
* **Logging tool (Logger batch task)** – If any issues occur during the creation of the translation memory, warnings and errors will be logged. These logs can be viewed at any time via SDL Trados using this logging batch task tool.
* **Toolset to send feedback (Feedback form)** – Users can contact the developers/product owners at any given time using this feedback form which will send an e-mail to the company’s mailbox.

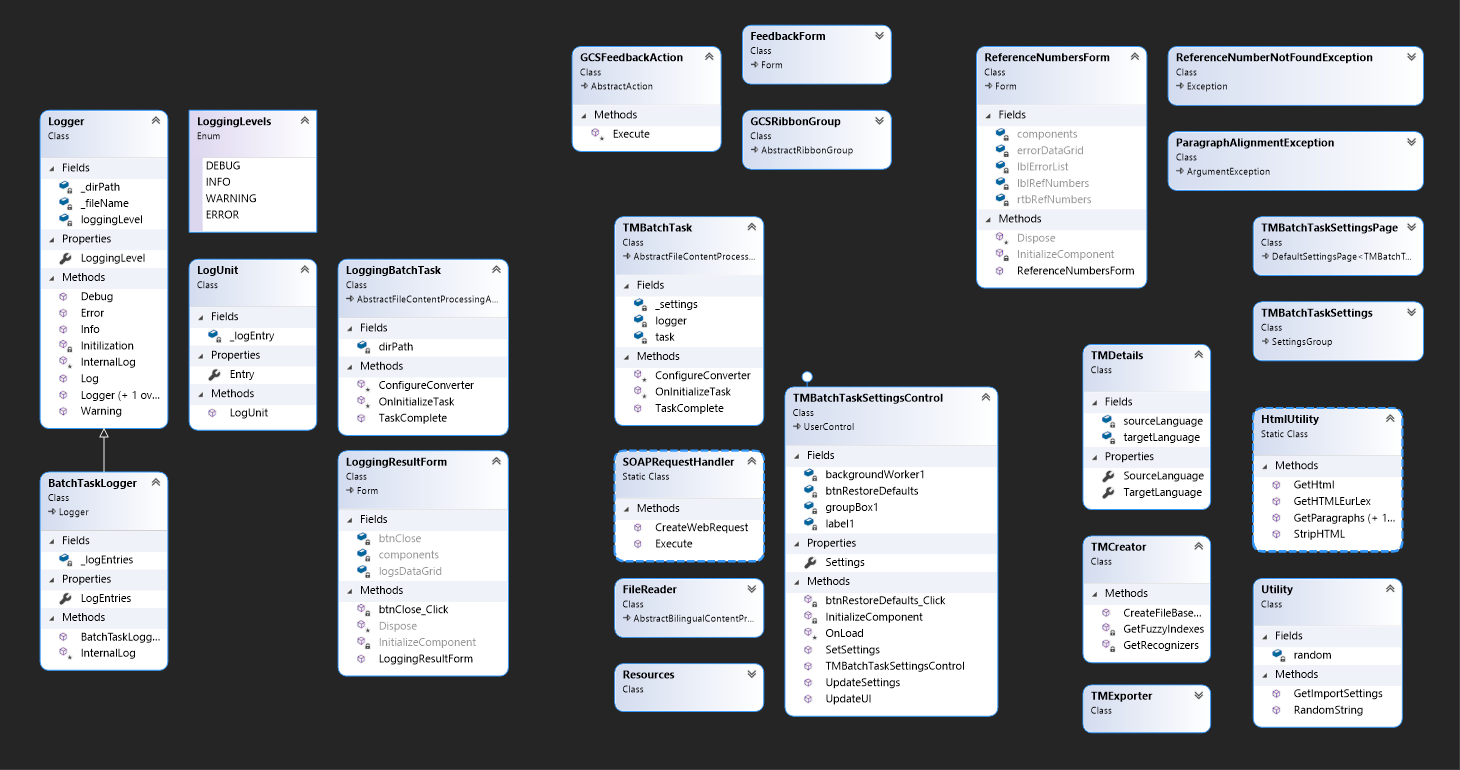


Figure - Code (C4)

All other classes serve a utility purpose to aid the functionality of the components mentioned above. The goal was to follow the SOLID programming principles and have modular code structure where each class/utility method serves only a single purpose.