OSCake : Open Source Compliance artifact knowledge engine

Development • Documentation • Support • Contribute • Contributors • Licensing

develop branch

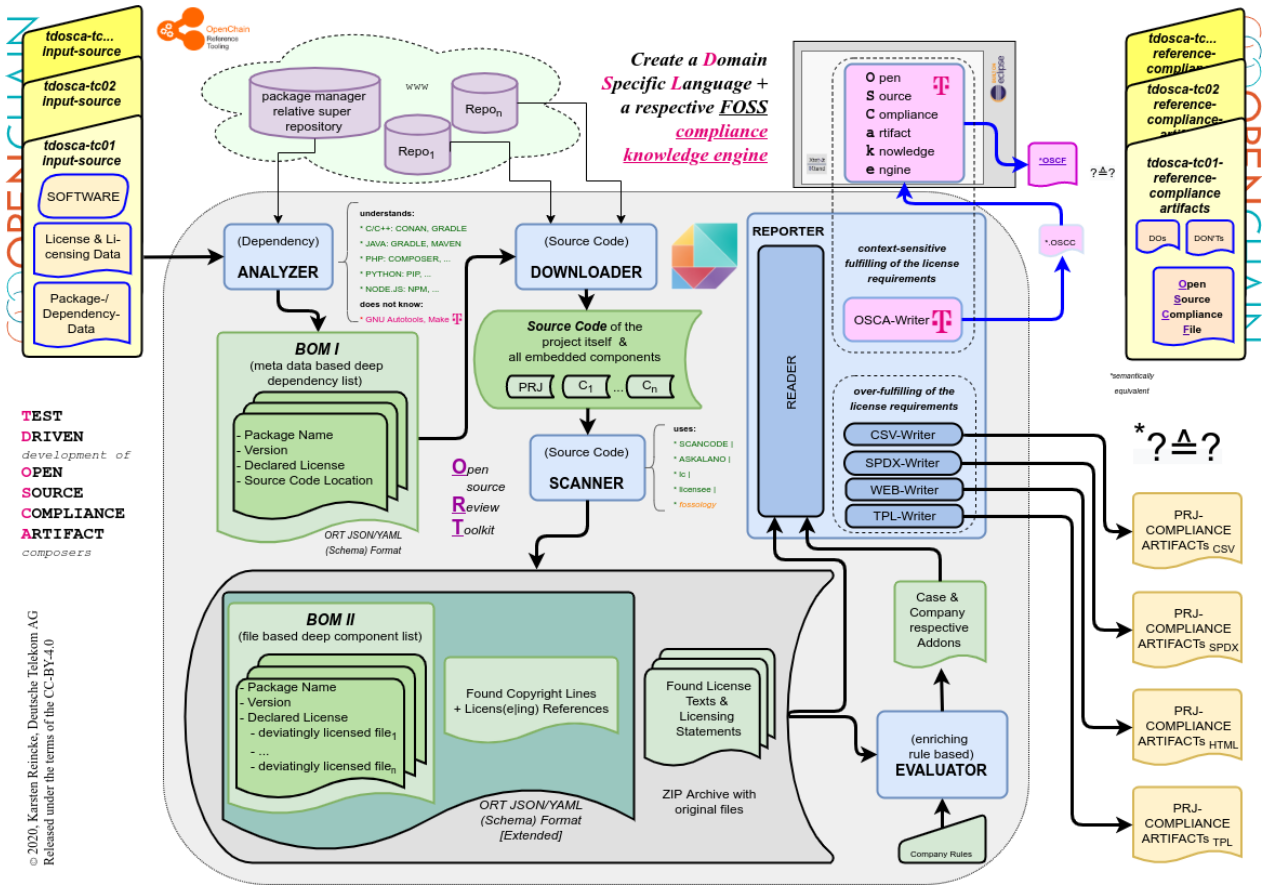
The goal of the OSCake project is to develop an XTEXT / XTEND based intelligent Open Source Compliance artifact knowledge engine, that

- takes a description of a package collection and the compliance artifacts found in the packages
- creates the one **O**pen **S**ource **C**ompliance **F**ile that - if distributed together with package collection - assures that the package collection is distributed compliantly = in accordance with the requirements of the involved licenses.

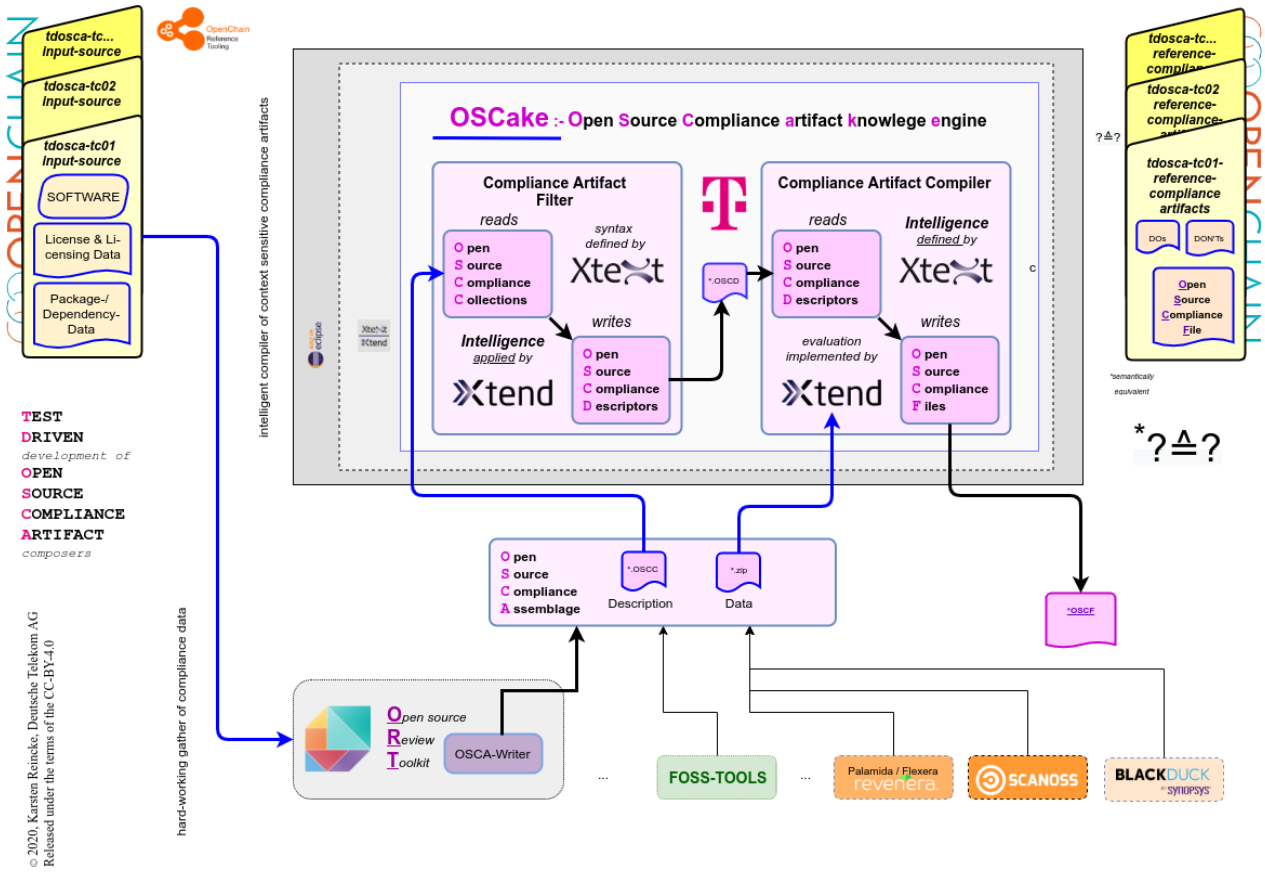
The point of this project is, that the knowledge which Open Source compliance artifacts have to be created / delivered in the context of which licenses and which architectural constraints is inherently embedded into the Domain Specific Language(s) defined and evaluated by XText and XTend.

About this component

If you read the reasons to set up the TDOSCA initiative and especially the transcription of our lecture given on the Open Compliance Summit 2020, then you end up in sheet signaling in which sense (for example) ORT and OSCake will cooperate: OSCake takes results gathered by ORT and compiles the really license adequate Open Source Compliance File:



OSCake applies the Open Source License Compliance knowledge - inherently in a declarative manner represented into the domain specific language - and creates the inherently license adequate Open Source Declaration File from which it derives the distributable markdown version. The more precise architecture of *OSCake* looks like this



So, existing Open Source scan tools create large lists of compliance entities that in any sense could be relevant for creating Open Source Compliance Artifact(s). *OSCake* takes these more or less complete and mostly over-fulfillinf sets. The *Open Source Compliance artifact knowledge engine* knows which of the articats found by the Open Source scanning tools must used in which license context and derives the one Open Source Compliance File which really meets the requirements of the involved licenses.

Getting the OSCX language definitions run:

1. Install the *Eclipse IDE for Java and DSL Developers* from https://www.eclipse.org/downloads/packages/. (Alternatively install the Xtext and Xtend via the Eclipse Marketplace)
2. Install a markdown viewer (optional)
3. Create a new *Eclipse Working Directory* `ews.dsl`.
4. Inside of this directory create the *Eclipse Working Directories* `ews.xtx.` and `ews.osc`
5. Start Eclipse and select `ews.dsl/ews.xtx` as working directory.
6. Switch to the XText view and create two new XText projects with the parameters:
   - Project a:
   - Project name: `de.oscake.strict`
   - Language name: `de.oscake.strict.Oscf`
   - Extensions: `oscf`
   - Project b:
   - Project name: `de.oscake.weak`
   - Language name: `de.oscake.weak.Oscc`
   - Extensions: `oscc`
7. Inside of your Eclipse, call `run as/Generate XText Artifacts` from the context menu of the files which have automatically been created:
   - *src/de.oscake.strict/Oscf.Xtext*
   - *src/de.oscake.weak/Oscc.Xtext*
8. On the file level copy the following files from the OSCake repository to the eclipse working directory:
   - `cp src/Oscc.xtext -> $HOME/ews.dsl/ews.xtx/de.oscake.weak/src/de/oscake/weak/`
   - `cp src/OsccGenerator.xtend -> $HOME/ews.dsl/ews.xtx/de.oscake.weak/src/de/oscake/weak/generator/`
   - `cp src/Oscf.xtext -> $HOME/ews.dsl/ews.xtx/de.oscake.strict/src/de/oscake/strict/`
   - `cp src/OscfGenerator.xtend -> $HOME/ews.dsl/ews.xtx/de.oscake.strict/src/de/oscake/strict/generator/`
9. Inside of your Eclipse, recall `run as/Generate XText Artifacts` from the context menu of the replaced file *src/de.oscake.strict/Oscf.Xtext* and *src/de.oscake.weak/Oscc.Xtext*
10. Call `run as/Eclipse Application` from the context menu of *de.oscake.strict*
11. Select `ews.dsl/ews.osc` as working directory for the automatically started second eclipse instance
12. Create a new Java project.
13. Inside of this project, create a directory `src-gen` as sibling of the directory `src`
14. Create a new file *src/what-ever-you-want*.oscf.
15. Play around with inserting your first *Open Source Compliance Declaration*. (Keep in mind: `String Space` allows you to select the next syntactically valid input)
16. On the file level copy test/a-input.oscc/*.oscc from the OSCake repository -> $HOME/ews.dsl/ews.osc/src/`
17. Press key F5
18. Exec the following steps to test a complete round trip from oscc via oscf to oscf.md:
    - open `tc05.oscf`
    - insert a blank outside of the code and save the file (that triggers the automatical generation of tc05.oscf)
    - open `tc05.oscf`
    - insert a blank outside of the code and save the file (that triggers the automatical generation of tc05.oscf.md)
    - open `tc05.oscf.md`

Modifyiing the language definition of OSCF

- Work on `src/de.oscake.strict/Oscf.Xtext` for improving the strict Open Source Compliance Definition language.
- Work on `src/de.oscake.strict.generator/OscfGenerator.xtend` for improving the evaluation of oscf-files.
- Work on `src/de.oscake.weak/Oscc.Xtext` for improving the weak Open Source Compliance Collection language.
- Work on `src/de.oscake.weak.generator/OsccGenerator.xtend` for improving the evaluation of oscf-files.

Keep in mind:

- The definition of a valid OSCF file (written in the XText file Oscf.Xtext) declaratively defines the compliance knowledge.
- The corresponding Open Source Compliance File (im Markdown format) is derived from the OSCF file by the OscfGenerator.xtend.
- The definition of a valid OSCC file (written in the XText file Oscc.Xtext) defines the elements a scanner can collect / handover to OSCake.
- The OsccGenerator.xtend applies the knowledge defined in OSCF: he derives OSCF file from the OSCC file (by throwing away what's unnecessary etc.) and to mark what's still missed to create a valid OSCF = a appropriate OSCF.

Code of Conduct

This project has adopted the Contributor Covenant in version 2.0 as our code of conduct. Please see the details in our CODE_OF_CONDUCT.md. All contributors must abide by the code of conduct.

Working Language

We decided to apply *English* as the primary project language.

Consequently, all content will be made available primarily in English. We also ask all interested people to use English as language to create issues, in their code (comments, documentation etc.) and when you send requests to us. The application itself and all end-user facing content will be made available in other languages as needed.

Documentation

TBD

Support and Feedback

The following channels are available for discussions, feedback, and support requests:

Type	Channel
Issues	
Other Requests	

How to Contribute

Contribution and feedback is encouraged and always welcome. For more information about how to contribute, the project structure, as well as additional contribution information, see our Contribution Guidelines. By participating in this project, you agree to abide by its Code of Conduct at all times.

Contributors

Our commitment to open source means that we are enabling -in fact encouraging- all interested parties to contribute and become part of its developer community.

Licensing