How To Contribute Telekom Software to Existing Open Source Projects

(1) Introduction

Deutsche Telekom not only encourages its employees and projects to use open source software, but also to contribute its improvements to other foreign Open Source Software (FOSS) projects. Besides fulfilling the open-source license requirements, publishing (parts of) one's own work as open-source software means successfully participating in the open-source community and playing the game in accordance with the open-source rules and customs.

This guideline describes how to contribute Telekom improvements to existing (foreign) open-source projects.

(2) Existing Tools

For contributing software to a GitHub repository correctly, Deutsche Telekom has designed a process documented by several files:

- The process flow chart
  - the complete process chart as PDF file
  - the main process as PNG file
- The guideline how to create an appropriate GitHub account
  - in English
  - in German

(3) The Process at a Glance

If you - on behalf of Deutsche Telekom - want to contribute your work to an existing open source software project, do this:

- Download and study the process flow chart
- Download the corresponding checklist, execute the steps and fill in the checklist as a form.
- Get in touch with the OSPO of Deutsche Telekom, hand over the form, and you will get the respective repository in an initialized version which already contains necessary and/or typically used compliance and health documents.

(4) Process in Detail

The following text is intended to be read by DT employees who need more (background) information. It follows the structure of the checklist mentioned above, but it additionally explains, why the single steps have to be taken. For those, who need a quick result, the documents mentioned under (3) could be more helpful.

Preparing the Contribution

- [ ] **S00**: Get the approval by your project and/or line manager.

> There are three reasons why you must contact your line manager:
>
> - To contribute your DT software to an existing FOSS project is encouraged by our company - if that steps supports your business needs. Wether that's true in your specific case cannot be decided by a general unit like the DT **O**pen **S**ource **P**rogram **O**ffice, but only by the project itself. So, the responsible managers must approve the publication.
> - If you as an employee of DT publishes DT software as open source software, you release a piece of intellectual property of DT. Hence, getting the approval of your line manager is a step of protecting yourself.

> - Most of us use their 'private' GitHub account also when they work as DT employee. To deal with two different accounts is very unhandy - as it has been turned out in some tests. Therefore, DT allows you to use your 'private' GitHub account. But it requires that you behave yourself in accordance with the DT Code of Conduct. To avoid misunderstandings your line manager should know that you are going to release DT work as open source software.

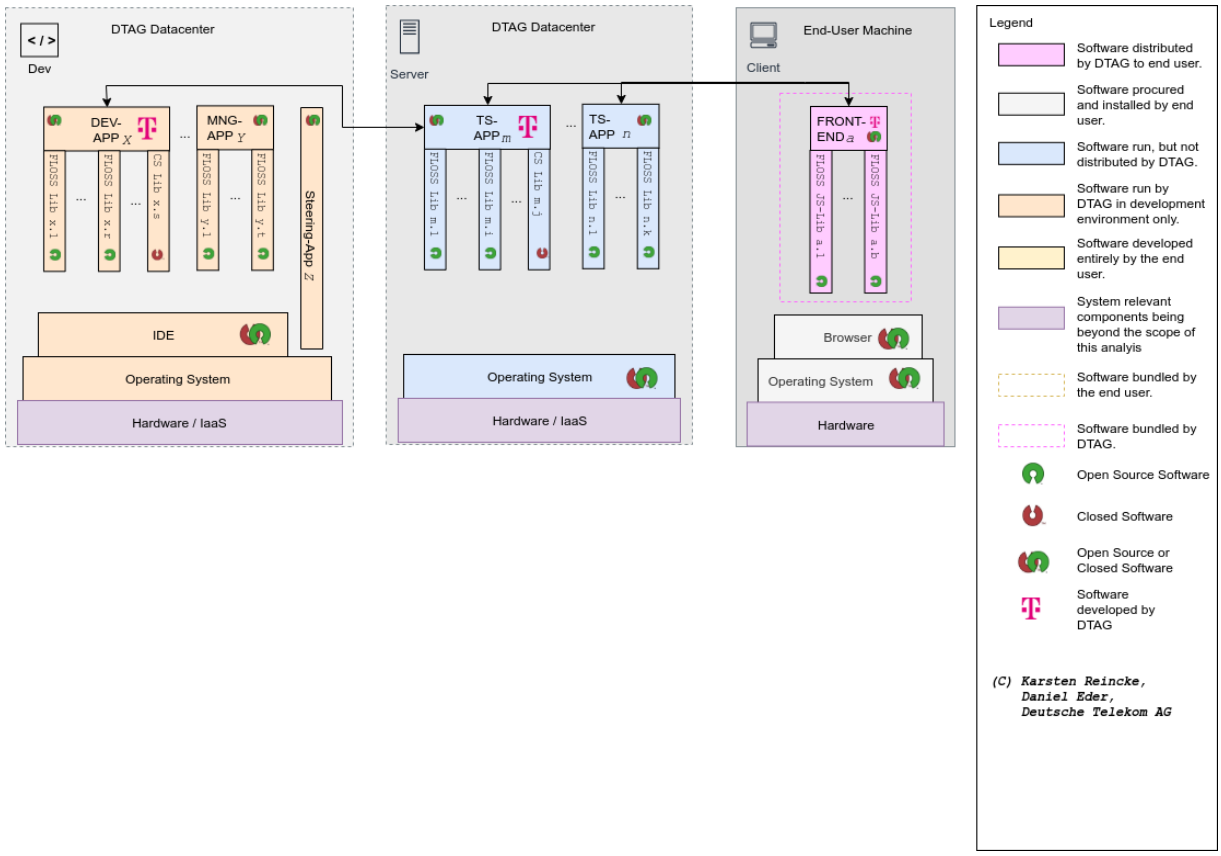
- [ ] **S01.***: Inaugurate an Open Source Supervisor as contact person for the Open Source Program Office (OSPO)

> To compliantly contribute to a FOSS project, you probably need help. The OSPO brings the respective legal and procedural knowledge into your project by training a contact person to do the right things or to support you to do so adequately. If you are the single one who contributes, you may become your own *Open Source Supervisor*, as we call such contact person. Howsoever, an *Open Source Supervisor* should fulfill a specific skill profile (see https://files.opensource.telekom.net/os-supervisor-profile.pdf). Feel free to either re-use / inaugurate a global OS-Supervisor for/of your department or to inaugurate your project specific OS-Supervisor. This supervisor is trained by the OSPO to support you practically.

- [ ] **S02**: Create your GitHub accounts in accordance to the guideline How to create Telekom fitting GitHub accounts

> Most of us use their private GitHub accounts also if the act as DT employee on GitHub. Therefore, these accounts should especially been protected from being abused. How to protect your accounts by some methods offered by GitHub is explained by this guideline.

- [ ] **S03**: Describe the architectural environment into which you want to contribute:



> This is the output of the drawio template DT offered for documenting software architectures.

- [ ] **S04**: Using the software architecture create a list all FOSS sub libs / components your part of the software requires as embedded (sub)sub-components.

> Contributing to an existing code can mean to modify an existing file or to create new files. In both cases you may have added code that requires specific sub-libraries. So, document in a **B**ill **O**f **Material** which sub-libs are required by the file you have modified or created

Avoiding unwanted IPR / patent side effects

- [ ] **S05.A**: Determine whether you have applied a patent of / for your software? `If NO -> S5C`.
- [ ] **S05.B**: Decide whether you want to set up a patent based business model `If YES do not publish your code as FOSS`.

> If you have applied a patent with your code and if you want to establish a patent based business model (= getting fees for permitting the use of the patent), then you should not release your software as open source software: Their exist some open source licenses with an explicit patent clause which permits the free use of your patent in the context of your software. And for the other open source licenses it is often legally argued that they contain an implicit patent clause. Hence, if you release your patented software you are about to subvert your intended business model

- [ ] **S05.C**: Are there any patents in the DT patent pool which are necessary to use your part of the software or any other part of the system into which you want to contribute `If NO -> S06`.

- [ ] **S05.D**: Obtain the approval for the respective patents to be used free of charge in the context of your software. `If you don't get such an approval do not publish your code as FOSS`.

> It is not necessarily harmful for DT if your software touches a patent of its patent pool. DT owns a lot of patents by which it wants to protect itself against patent trolls etc. Due to the fact that an open source license (implicitly) permits the use of the patent only in the context of the licensed software, the protecting effect is not negated generally. But DT also owns some (telecommunication) patents it does not want to permit to be used under any circumstances. Thus, you have to make sure that the DT patents touched by your software do not belong to those DT never wants to be used for free.

Choosing a FOSS License by Avoiding copyleft conflicts

- [ ] **S06A**: Determine whether elements of the specific BOM of your contribution are distributed under a strong copyleft license?
  - [ ] `IF several of them` **S06.B**: Manage the license conflict. `RESTART!`
  - [ ] `If one of them` **S06.C**: consider whether that is the same license under which that part of the project is released to which you want to contribute. `If NO > S06.B ELSE > S07`
  - [ ] `If none of them > S07`
- [ ] **S07.A**: Contribute your software under the same license under which that part of the project is released to which you want to contribute

> If two sub components are licensed under different licenses with strong copyleft effect, then both licenses require, that the software using the sub components must be licensed under the same license as the sub component. This condition cannot be fulfilled by the using main component. Hence, you have to solve the conflict.

Gathering the Repository Data

- [ ] **S08.A**: Become familiar with the project specific contribution guideline

> Contributing to an existing open source project requires that the contributor follows the contributing guideline of that project.

- [ ] **S08.B**: Determine whether the project requires to sign a Contributor License Agreement (CLA)?
  - [ ] `If NO > S09.A`
- [ ] **S08.C**: Sign the required CLA in accordance with the project specific contribution guideline and with the help of the OSPO

> DT offers an own process for signing contributor license agreements.

- [ ] **S09.A**: Let all newly created files of your contribution start with the specific DT file header

> All files of your contributions package you have initially created must contain file header indicating
>
> - that the file is part of the respective project
> - the copyright owners who have created it

> - under which license they have released the file
>   So, the file should have a file header like this
>
> `{PROJECT-NAME}`
>
> `Copyright (c) {YEAR} {NameOfTheDevloper}, Deutsche Telekom AG`
>
> 'This file is made available under the terms of the license {LicenseName}'
>
> `SPDX-License-Identifier: {SpdxIdentifier}`

- [ ] **S09.B**: Insert a DT copyright line and a comment in all files that you have modified by a sufficiently weighty contribution.

> A DT copyright line must contain the name of the (initial) developer and a DT reference:
>
> `Copyright (c) {YEAR} {NameOfTheDevloper}, Deutsche Telekom AG`
>
> It is necessary to insert both parts, because an employee of DT has generally transferred his rights to DT by his contract. So DT is the copytright holder of the software and must release it. But there is one right, which cannot be transferred to any third party: the right to be named as author. Therefore, the name of the initial developer must also be added to the Copyright line.

- [ ] **S10**: Contribute your work in accordance with the project specific contribution guideline

---