TU Delft Guidelines on Research Software

Licensing, Registration and Commercialisation

TU Delft

TU Delft Guidelines on Research Software V 1.0

Licensing, Registration and Commercialisation

16 February 2021

Name of this document	TU Delft Gudelines on Research Software Licensing, Registation and Commercialisation
Date approved	16 February 2021
Version	1.0 - http://doi.org/10.5281/zenodo.4629635
Previous versions	
Person responsible	Head of Intellectual Property, TU Delft Valorisation Centre
Location of the Guidelines and accompanying Policy on TU Delft website	https://www.tudelft.nl/en/library/research-data-tamanagement/r/policies/tu-delft-faculty-policies

Table of contents

1 Executive summary

When dealing with software, the following considerations need to be taken into account:

- Research software falls under copyright law
- Software can be patented or licensed
- If a patent is considered, report your invention using an Invention Disclosure Form (IDF) at TU Delft's inventor portal: https://inventions.tudelft.nl/inventor*
- Software licences can be proprietary or open source
- Follow the decision tree (Fig. 1) in Chapter 4 to determine if an open source software (OSS) licence can be applied
- Eight types of open source licences are pre-approved by TU Delft: Apache, MIT, BSD, EUPL, AGPL, LGPL, GPL, CC0
- A description of the software and the licence must be registered in PURE [1]
- In order to apply an OSS licence, the software must be published
- Contact the data steward at your Faculty for questions about research software

**To access this link, the user must be connected to the TU Delft Intranet using an eduVPN connection.*

2 Introduction

Good software is essential for twenty-first century research. An overwhelming majority of researchers make use of software as part of the research process. Consequently, software is increasingly recognised as an independent research output, varying from a single line of code to complete software packages comprising several interacting programs. As such, software should be well documented, preserved and whenever applicable the FAIR principles (Findable, Accessible, Interoperable, Reusable) should be followed. Wherever possible, TU Delft encourages its researchers and software developers to make their software available Open Source, in the spirit of our Open Science programme. [2] There are different ways in which software can be shared with society. Software can be made freely available to a large audience through a repository such as GitHub or shared peer-to-peer under restricted licences. Alternatively, it is possible to run programs on dedicated servers, allowing only access to the outcome of the programs, while keeping the source code confidential. Yet, when making software available to others there should always be a balance between the level of openness and access granted, possible commercial exploitation of software and any (legal) restrictions that prevent sharing it as open source.

This document is written in addition to the TU Delft Research Software Policy. The policy sets out the high-level requirements for how software should be managed;

the responsibilities of the different stakeholders involved in software development and describes the global workflows for sharing software openly. The guidelines presented here are the 'how-to' and go into more detail on the prescribed workflow for sharing software either under an open or proprietary licence, providing more background and considerations, additional definitions and references.

It should be appreciated that these guidelines will require some added effort from all involved compared to the current state of affairs. Please note that under the current collective labour agreement software developers and researchers are actually obliged to report their software. This requirement is hardly ever met and is currently not enforced within academia. However, with the abovementioned change in recognition of the importance of software contributions and in the scope of fairly rewarding people for their contributions to the Open Science programme of TU Delft, the need for reporting gains a new dimension. The main goal of this document is to provide a pragmatic, workable solution to the need to register, document and report our collective software output. The document is also intended as a "living" document, allowing for input, improvements and updates wherever it does not meet these goals.

2.1 Target audience

The TU Delft Research Software Policy and these guidelines apply to anyone at TU Delft who is involved in software development (incl. researchers, software developers and other employees) that will be shared through either an open licence, proprietary licence or a patent.

Please note that both bachelor's and master's students at TU Delft in principle hold their own right to their software unless they have made use of university resources and/or received intellectual support from university researchers, software developers and/or staff for its inception. In that case, it is a "joint" result with the rights shared between the student and TU Delft. When students hold their own rights, the guidelines are explicitly not applicable to them, although these guidelines may naturally be used as a way marker for students looking for information.

The document also is not directly applicable when 'merely' contributing to the work of others, for instance, when a developer is responding to a pull request on GitHub. However, when the contribution being made is substantial, it may still be prudent to register it in PURE. This is voluntary, but may be of interest to the developer, especially when in the foreseeable future attribution and contribution to Open Science becomes used as a metric of merit.

Please note that we often refer to "researchers" throughout the document, how-

ever this should be understood to encompass every employee at TU Delft writing, dealing with and creating software (including support staff for example, Education & Student Affairs, TU Delft Library, etc.). Concomitantly, all the support staff and contact points described herein will be at your disposal as well.

2.2 Structure of this document

Chapter 3 provides some background on the copyright questions underpinning this document.

Chapter 4 describes the different aspects that need to be considered to determine if the software can be shared under an open source software (OSS) licence. A decision tree (Fig. 1) to guide software developers and researchers on when it is possible to apply an OSS licence to software is provided and the different considerations are explained. TU Delft has pre-approved a subset of licences that can directly be used by researchers when following these guidelines (Apache, MIT, BSD, EUPL, AGPL, LGPL, GPL, and CC0). These licences are described in detail in Appendix 2. The use of other licences is allowed upon agreement with the TU Delft Valorisation Centre.

Researchers and developers are encouraged to follow the decision tree presented in these guidelines. In case of questions and uncertainties on how to use it, researchers can contact the data steward at their faculties. It is strongly recommended to seek support at early stages in their project. When in doubt, reach out.

Chapter 5 describes the follow up steps that must be taken by staff, software developers and researchers when an OSS licence can be applied.

In addition, Chapter 6 describes the considerations that come into play when dealing with proprietary research software licences.

2.3 Support

TU Delft support offices are there to guide and assist software developers, researchers and staff through the process of creating and sharing software, but **it is the responsibility of each employee to seek advice from support staff when in doubt** about the procedure. A few of the main support roles that come into play are:

The **faculty data stewards** provide advice to software developers and researchers on how data (including code) can be managed, stored and preserved in accordance with internal TU Delft guidelines. The data steward should be the first contact

point for the software developers, researchers and staff when dealing with software related questions. If further questions arise, they will be able to redirect software developers, researchers and staff to the appropriate contract managers, IP managers, Investment Managers of Delft Enterprises or other experts, when and where needed. It is prudent to connect with the data steward as early as possible in your software project.

The **contract managers** at the faculties need to be involved when dealing with commercialisation of software projects or when involving industrial partners in the project. They will ensure that the Dean and faculty are properly informed, secure compliance of all the licences with internal TU Delft guidelines, pre-existing contractual arrangements and (European) legislation. When a contract deviates from TU Delft general legal guidelines, the legal advice from **legal affairs** is mandatory. The contract manager will assist with securing this advice in a timely manner.

The **intellectual property managers from the Valorisation Centre** assist with all Intellectual Property (IP) related matters at TU Delft. They will provide support when filing and administering all forms of IPR (such as patents, licences, industrial designs, trade secrets) and will assist software developers, researchers and staff when needed in drafting and negotiating the IP section of licences and collaboration agreements with industrial partners. When a contract involving the generation or transfer of intellectual property deviates from TU Delft guidelines, the legal advice from IP legal (VC) is mandatory. The contract manager at the faculties will assist with securing this advice in a timely manner.

Lawyers of Legal Affairs are responsible for all legal matters, including IP and all kinds of agreements, including licence agreements, collaboration agreements, etc. In case of complex matters or whenever required according to the TU Delft Mandate Agreement (*Mandaatregeling TU Delft*), the contract manager at the faculties will contact one of the lawyers from Legal Affairs for (additional) advice.

3 Copyright and software – some background

Before starting to discuss the different paths to share software, it is very relevant to keep in mind that creation of software is protected under copyright law (for an explanation about the law see Box 1).

Why is this important?

- Based on the law, TU Delft owns the software that is written by employees.

- When copyright exists, others (i.e. general public or a commercial entity) are not free to reproduce, copy, adapt, amend, distribute, perform, display or make the software public without the explicit permission of the owner of that copyright.

In contrast to other types of intellectual property (like patents, trademarks or design rights), there is no registration of a copyright needed to claim the creator's right of software. Even when an explicit notification is absent in a program, the copyright exists and infringement of the copyright occurs when using the software without explicit authorisation of the creator.

Therefore:

- If software developers, researchers and staff creating software want to make

their software available for free, the permission of this intention must be clearly indicated to the public by using an appropriate open source software licence (OSS).[3]

- In order for the software developers, researchers and staff to use an OSS licence TU Delft is willing to waive its claim to the copyright provided the procedure described in these guidelines is followed. Please see Chapter 4 for all the necessary steps. Note that in this case the software developer or researcher establishes an own copyright: e.g. © (2020) John Doe, Delft, the Netherlands.

- When a proprietary software licence is intended and in all other cases, software developers, researchers and staff should assert copyright on behalf of TU Delft. This can be done by using the copyright symbol, followed by a date and an identifier: e.g. © (2020) Technische Universiteit Delft.

4 How to determine if an Open Source Software (OSS) licence can be applied?

4.1 Considerations and conversations

For convenience, the questions that need to be answered to determine if an open source software licence can be applied are depicted in the figure below (Fig. 1). Each step is also described in more detail below.

It is important that the questions be answered correctly. When registering the software in PURE these questions will also be listed there. Therefore, when in doubt please consult with the data steward at your faculty.

Step 1: Funding of the research

In order to assess the question whether TU Delft is the sole entity that can decide on the possible licensing of the work, it is essential to identify possible other Intellectual Property (IP) candidates.

If a project has been solely funded by the government through direct funding, TU Delft owns all of the IP of the project based on the collective labour agreement (CAO VSNU-2019). In that case, it is possible for TU Delft to waive the copyright.

When companies are involved, there are practically always clauses involved that secure certain claims on the results and clauses that prohibit the use of (certain types of) open source software licences. Researchers, software developers and/or data stewards should contact the contract manager at their faculty to identify the contracts associated with the research project and closely follow the clauses provided.

If a project was funded by government agencies, including the European Union, it is unclear if the researcher and staff have the freedom to operate. In most grants and subsidies, the IPR is left to the inventor(s). However, this is not always the case. When companies are involved in the project (e.g. NWO-demonstrator call), they have a right-of-first negotiation and/or a royalty-free, non-exclusive licence to exploit the software. Therefore, it is important that software developers and researchers contact the corresponding contract manager to retrieve the contracts associated with their projects.

Box 1. Copyright and a software example

Copyright is a type of intellectual property right (IPR) that automatically comes into being the minute developers or researchers create a piece of software. In order for copyright to exist, only two requirements must be met:

- **The creation must have an original character, meaning there must have been a creative process (without a creative or intellectual effort a copyright does not exist).**
- **The creation must bear a personal mark of the creator: there must have been an identifiable involvement of a human mind guiding and steering the process of creation.**

Note that the copyright law deals with 'works' (e.g. software) and not ideas or concepts of nature. Therefore, data, a formula, an algorithm or a gene sequence are not subject to copyright as they are not 'works'.

Another point is that copyright sees to the form of the works. In software terms: the code as such is copyrighted, the functionality behind the code is not. This is in contrast to patent rights, which deal with the exact opposite. Patents protect the functionality. So when a researcher converts a piece of code from Python to C++, copyright is not infringed. However, if there is a patent on the (algorithms behind the) Python code, patent law is infringed when transcoding it into C++.

Another caveat of copyright is that although it exists "by creation", when challenged, the responsibility is on the creator to prove that he/she created the software in a legally binding and unambiguous matter. For this reason, it is very important to have your software deposited and registered, especially if the software is not published immediately in an online repository like GitHub.

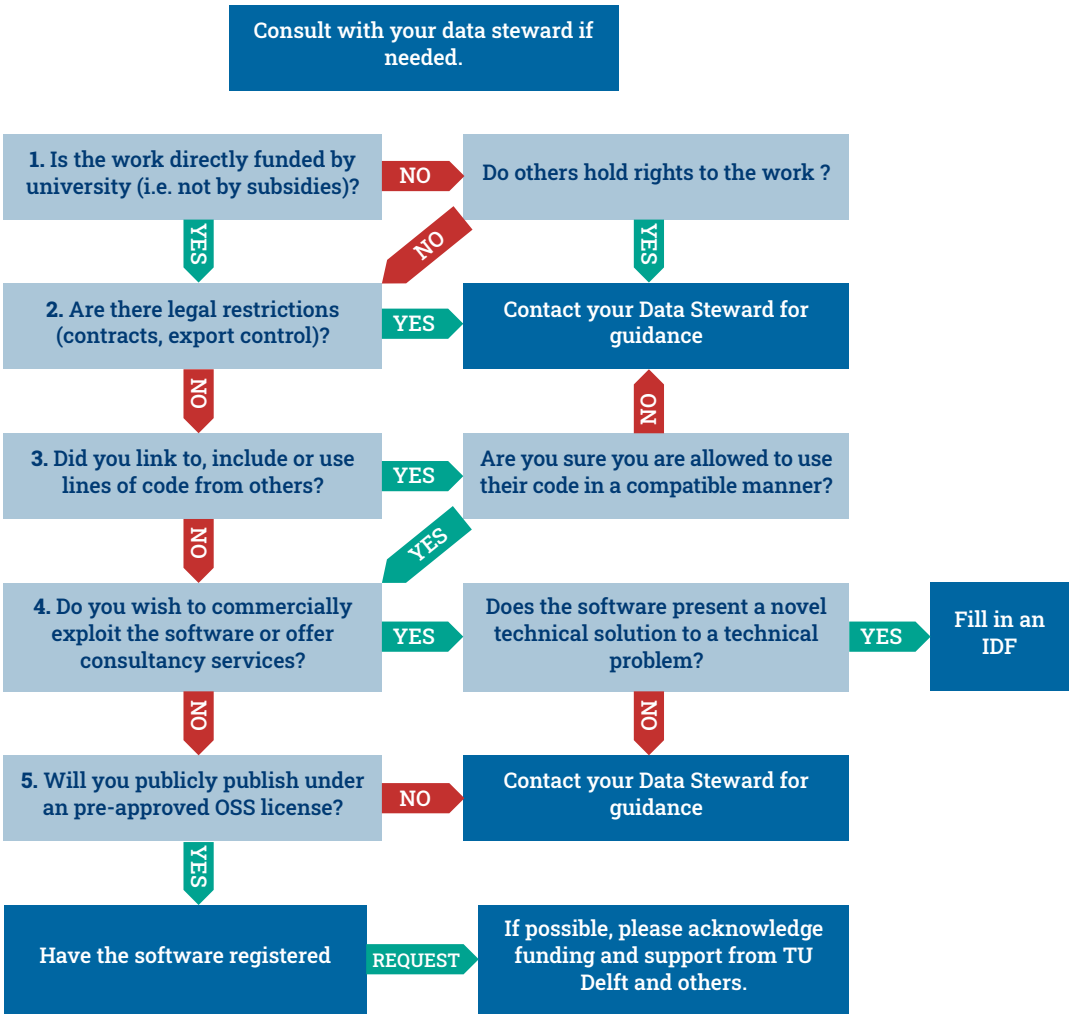

Figure 1. Decision tree to guide software developers, researchers and staff on when they can apply an open source licence to their software. OSS: Open Source Software, IDF: Invention Disclosure Form

Step 2: Legal restrictions

In a sense, when applying an open source licence on software, then developers, researchers and staff are exporting it to all countries across the globe simultaneously. This brings some responsibilities and considerations.

An important consideration is covered by the “dual use” regulations. Dual use concerns programs that have a beneficial application at heart, but that can be easily turned into something more dangerous or malicious. To give an example: software for the automated use of drones will generally be used to the benefit of society, but it may be used to disrupt air traffic. That is why these types of programs cannot be simply shared online.

There is a non-exhaustive list of “dual-use” goods. [4] The list gives a good indica-

tion of what items and programs are considered as “dual use”. If something is on the list, then by definition it is considered a “dual-use” good and cannot be freely/openly shared online. Note: the aforementioned example regarding the drone is indeed listed under 9E102 - Technology for the use of Unmanned Aerial Vehicles.

Other grounds for export control are economic considerations. Currently, multiple countries in the world are in a race to develop quantum computing. The competitive economic benefit of the first to succeed is considered so high that all European funds that facilitate research in the area of quantum computing have clauses that prohibit the export (and thus open publication) of all results (including software), that has been developed using European funding.

A last legal restriction is provided by the European privacy laws. Anything dealing with storing or handling personally identifiable data is subject to strict regulations provided under the GDPR (General Data Protection Regulation). In case of any questions, contact the data steward at your faculty.

Step 3: Does the software contain lines of code from others?

It is important and good practice to keep track of the origin of all code included in the software. This is not only needed for proper attribution [5], but also to know what type of licence can be used when reusing code from others.

All lines of code have their own copyright associated with them and might have a licence. If that is the case and if software developers and researchers would like to build upon existing software (or lines of code), it is important to consider the compatibility of software licences when thinking of making software openly available (Table 1). Not all open source licences are compatible with each other, meaning you cannot use them interchangeably.

As described above, not all types of open source software licences are compatible. Therefore, it is best practice (and sometimes, even compulsory) to simply follow the licences already in place when using the work of others. Naturally, this consideration also implies that when the researcher asks permission to use a licence outside the TU Delft pre-approved list by reason of it being the licence already in place, then the permission to do so is practically a given. The researcher is still requested to make a formal request to the Valorisation Centre however, for the purpose of monitoring and to facilitate updates of the pre-approved list contained in this document (e.g. if there is an unlisted licence that turns out to be in particular demand).

Proprietary software cannot be used without explicit written permission from the owner, either in the licence terms or in a written permission from a person authorised to assign a licence on copyright and/or ownership. When using an open source software licence, the clauses of the corresponding licences automatically provide grounds for such a permission and assignment.

Original Combine with?	CCO	MIT	BSD	Apache	EUPL	GPL, AGPL or LGPL	Proprietary
CC0	YES	YES	YES	YES	YES	NO	NO
MIT	YES	YES	YES	YES	YES	NO	NO
BSD	YES	YES	YES	YES	YES	NO	NO
Apache	YES	YES	YES	YES	YES	NO	NO
EUPL	YES	YES	YES	YES	YES	NO	NO
GPL, AGPL or LGPL	YES	YES	YES	NO	NO	YES	NO
Proprietary	YES	YES	YES	Claused	NO	NO	Depends on licence

Table 1. Scheme depicting the compatibility of the OSS licences described in this document

Step 4: Commercial exploitation

There are several ways to commercially exploit Open Source Software (OSS). Some examples are:

- Provide the program as open source, but allow clients to hire your services as a consultant in implementing, maintaining or even upgrading the software.

- Provide licences with free use of a basic model (like Adobe Acrobat), or proprietary licence with cheaper versions for dedicated purposes (like Microsoft – research and education purposes).

Note that each faculty also has their own guidelines and regulations in dealing with consultancy arrangements or the establishment of spin-offs. In addition, third party funders might have specific rules concerning research outputs, including software. For example, many grant schemes from NWO or the European Commission would expect software developers and researchers to make research outputs resulting from such grant funding as open as possible, which might preclude commercialisation possibilities. Therefore, you should consult the data steward at your faculty about commercialisation plans at an early stage of the project.

When the decision has been made to commercially exploit the software, a proprietary licence or even a patent application is the way to go. For this, the interested reader is referred to Chapter 6.

Step 5: Choose an OSS licence from the 'TU Delft pre-approved' list

TU Delft has defined a list of pre-approved software licences that software developers, researchers and staff can choose from when sharing software openly. Currently, these are Apache, MIT, BSD, EUPL, AGPL, LGPL, GPL, and CC0. Please note that it is a 'living' list; based on the use this list will be modified and appended in the future.

Some general recommendations when choosing an OSS licence:

- Generally, when you are working in a community it is best to use the licence that is associated with that community (i.e. GNU recommends GNU GPLv3 for most programs whereas Cloud Native Computing Foundation prefers to use the Apache License 2.0 by default).

- If at all feasible, it is also best to try to maintain coherence. Hence, if the work you used was all derived from MIT-licensed material it is preferable to release the resulting software under an MIT-licence too.

- If there is absolute freedom to choose and the researcher has no personal preference for either licence, some considerations come into play that are best discussed with the data steward. A major distinction is between the so-called "permissive" and "restrictive" open-source licence. The characteristics of these two groups will be explained in more detail in chapter 4.2.

In general, at the Valorisation Centre -in dealing with start-ups and corporately funded research- we observe that the Apache licence works well for all the different interests at play between university and corporate headquarters (or garage-box in case of a start-up). It tends not to raise a red flag from companies investing in R&D

projects at the universities and does protect the “open” character of the software. Apache was the first to embed a “patent-retaliation” clause that allows the software to stay open, even when under a patent. The main drawback associated with Apache is that it does not enforce openness on derivative programs.

Another staple in academia is the GPL licence, for the simple reason that any software that uses code that has been published under a GPL licence must automatically be covered by a GPL licence itself. GPL is also one of the oldest open source licences and has a large and active community of users behind it. The main drawback associated with GPL is that it is often contractually prohibited in public-private partnering.

Whatever the decision is, software developers, researchers and staff are encouraged to read the description of the licences and the full licence text provided in Appendix 2 in order to understand the implications of their choice. For further input, the software developer, researcher or staff member is also referred to https://choosealicense.com/ for information and guidance.

Step 6: Register your software

In accordance with the collective labour agreement, software developed by the researcher needs to be reported to the university. At TU Delft, we do this through registration. These registrations also serve several other purposes:

- Companies collaborating with TU Delft software developers and researchers on software development demand that TU Delft keeps track of the types of (open source) licences applied to different programs to prevent contamination.

- Society, being the main funding body of our software developers and researchers, has a right to be informed of what we do with their contributions. We can use these lists to report to the government, open science/open source interest groups and society.

- In the foreseeable future, open source sharing of software may be used as a metric to determine ‘scientific output’ and merit alongside peer-reviewed publications, data and teaching skills.

There are two different tracks for registration, depending on whether the researcher wants to have a proprietary licence (or a patent) on the software or an open source software licence.

If software developers, researchers and staff have decided to openly share their software, they must choose to either:

- Publish their software in the 4TU.ResearchData repository, providing one of the pre-approved licences listed and explained in this document. It is the task for the Library to make it possible for software developers, researchers and staff to submit software in a straightforward manner (e.g. facilitating the publication of GitHub repositories directly on 4TU.ResearchData) and to feed the information into PURE.[1]

- Publish their software in another suitable repository (e.g. GitHub, Zenodo). In this case, it is the researcher's responsibility to also register information about this software in PURE.

If software developers, researchers and staff have decided to establish a proprietary licence or a patent application, then the registration is embedded in the process (through submission of an IDF). It is the task of the support staff involved (mainly contract managers and legal affairs) to ensure this is done properly.

4.2 Public domain dedication (no licence), permissive and restrictive OSS licences

4.2.1 No Licence (CC0)

The list starts with CC0 from Creative Commons. Everything considered Creative Commons licences are not appropriate for software. The notable exception is CC0, which is commonly deployed to note something (including software) falls in the public domain and is freely available to all, without any requirement or need to attribute at all. [6] Note that The Boost software licence [7] and The Unlicense [8] also provide a proper public domain dedication, but the CC0 has as a benefit that the simple statement "© 2019 [NAME, PLACE]. *This work is licensed under a CC0 licence.*" suffices to cover the legal text of the licence in its entirety. The other two public domain dedications rely on extensive texts to be incorporated, which is rather awkward when your project consists of something akin to ten lines of code (or less).

1 It has been suggested to make this an automatic deposit linked to 4TU.ResearchData repository and the PURE system after filling in a questionnaire (along the lines of fig 1). When established, the text will be amended accordingly.

Even when the researcher does not want to claim or do anything with the software at all, it is not advisable to simply "put software online" without any licence [9]. After all, as explained above, copyright originates from the act of creating the software. Therefore, without notice, the copyright actually exists, but nobody knows how it will be applied or enforced and nobody else can copy, distribute, or modify the work. In addition, please do note, 'nobody' can include the actual software developer or researcher too. That is why it is important to use a "public domain dedication" instead of publishing the software with no licence.

4.2.2 Permissive software licences (MIT, BSD, Apache)

These licences allow users to do anything with the code as long as they properly attribute the original creator. In other words, when someone wants to place a proprietary licence on a piece of software that is derived from software under an MIT or a BSD licence, this is possible. BSD and MIT are the most open source software licences available. Another way of coining it, an MIT or BSD licence is always "eaten up" by the other licences.

The Apache licence is sterner in protecting the access rights. The licence agreement also contains a "retaliation" clause, which does not allow filing of patent infringement proceedings against people using the Apache covered code. It means that if someone files infringement proceedings (in court) on a software, which was itself a derivative of code under an Apache licence, then they infringe the licence itself. As a result, they automatically lose the licence and the right to use the software (and therefore their infringement claim is void and they become the perpetrator). The Apache licence is still considered a "permissive" licence because it can co-exist with other licences. For example, a basic package can be made freely available under Apache licence, whilst the add-ons and special features are made available (e.g. for a fee) under a proprietary licence.

4.2.3 Restrictive software licences (GPL, AGPL, LGPL, EUPL)

Note that all versions of GPL are actually considered compatible with these guidelines albeit with a recommendation for GPLv3 to be used for new programs and projects. This may be especially relevant for 'legacy' projects given that the GPLv2 and GPLv3 licences are incompatible with each other. In these situations, the software developer may therefore simply choose (or rather, is actually forced) to use the GPLv2 version.

The GPL (from the GNU General Public License suite), its derivative LGPL (L from 'Lesser'), another derivative AGPL (A from Affero) and the EUPL (European Union Public Licence) are all so-called 'copyleft' licences and thereby more restrictive than the two types of open source licences described above. A copyleft licence grants the right to freely distribute copies and modified versions of a software with the stipulation that the same rights be preserved in any derivative work [10].

From these two, the EUPL licence is slightly more permissive because it can co-exist with other open source software licences (including MIT, BSD and Apache) due to the existence of an 'exclusion and compatibility list' in the licence. For example, when a researcher incorporates a part of a software licensed under Apache it will stay Apache even when the project as a whole is under EUPL. Under GPL, the entire code would become GPL licensed. This runs counter to the express wish of the owner that enacted the Apache licence on her/his code and the clauses covering the Apache licence (which state that Apache covered stays Apache forever). That is why Apache and GPL are incompatible.

In the case of the GPL licence, all derivative work that uses software pieces under this licence automatically falls under the same licence no matter how small the GPL-licensed contribution is. The Free Software Foundation [8] behind the creation of the GPL licence even considers a link to a GPL-covered program as sufficient reason to have the linking program fall under the coverage of the GPL licence. However, this assertion has not yet been tested in court and is in the field generally deemed ill-advised and not enforceable.

For this reason, many companies sponsoring research at TU Delft often enact specific clauses in the collaboration agreements prohibiting the use of these copyleft licences for software to ameliorate the risk of having their in-house proprietary software "contaminated" with GPL (which would then result in them having to release their proprietary source code to the public).

When dealing with GPL, it is good to realise that open does not mean "for free" (gratis). It is possible to charge for the use of (copies of) GPL licensed software. However, due to the open nature of the licence, such business models will generally be based on "software as a service" or "maintenance and support" as sources of income.

Of all GPL licences, AGPL (Affero GPL) is the strictest of the group. It was designed specifically with server-based software in mind. In addition, it has a clause ensuring that if you run a modified program on a server and let other users communicate with it there, then your server must also allow them to download the source code corresponding to the modified version running there.

LGPL is the final family member to be discussed. LGPL is designed for library routines and is deemed a lighter version of GPL. When another person modifies software released under an LGPL licence the licence acts similar to GPL in that it automatically covers the new software too. However, when someone writes software that merely links and uses the library covered by the LGPL licence it does not automatically extend itself (whereas –as mentioned above- GPL does). This can broaden the potential uses of a library and makes it more accessible to the public at large (especially for those operating under proprietary, Apache or MIT covered works).

5 What to do to apply for an OSS licence?

It is important to remember that in principle TU Delft holds the rights to the software created by its employees (i.e. software developers, researchers and/or staff). So, some formal (legal) steps are needed to arrange matters properly.

When these guidelines are followed and when the software is published, TU Delft disclaims its copyright, allowing software developers, researchers and staff to hold the copyright to their software and thereby having the right to apply one of the pre-approved licences when sharing software. This is provided for in the following procedure:

1. Determine if it is possible to apply an Open Source Software licence to your project (see Chapter 4).

2. The waiver of TU Delft should be indicated in the software licence with the following text:

> **Technische Universiteit Delft hereby disclaims all copyright interest in the program "Name program" (one line description of the content or function) written by the Author(s).**
> **[Name Dean], Dean of [Name Faculty]**

3. Assert your own, personal copyright *(© YEAR, [NAME], [REFERENCE project, grant or study if desired]*.[2]

4. Apply one of the TU Delft pre-approved Open Source Software licences in the format and form described in the licence text after stating, *"This work is licensed under a [NAME and VERSION] OSS licence".*

5. Make the software openly available (for instance in an online repository such as GitHub).

6. Please consider acknowledging support from TU Delft and/or your funding provider.

7. Register the software either in 4TU.ResearchData or in PURE.

Please note that if the software is not published, and/or if the guidelines have not been followed correctly and/or if the software is not registered in PURE, then this 'agreement' is invalid and the software automatically falls under the legal copyright of TU Delft. This instantly nullifies the right of the software developer or researcher to apply for a licence and thus the open source software licence applied never came into existence. This works retroactively.

The pre-approved open source software licences at TU Delft are Apache, MIT, BSD, EUPL, AGPL, LGPL, GPL, and CC0. The licences are described in detail in Appendix 2 of this document.

2 The reason for waiving the copyright and having the software developers, researchers and staff file the copyright in their own name facilitates the use of copyleft licences.

6 Software commercialisation

If software developers, researchers and staff from TU Delft want to exploit their software commercially, there are several ways to do it. This chapter lists some things to take into consideration and discuss with the data steward. The data steward will then connect the software developer, researcher or staff member with the contract manager of the faculty involved.

A common option is to provide the program as open source, but to allow clients to hire your services as a consultant in implementing, maintaining or even upgrading the software.

An alternative option is by either providing licences, following the model of Adobe Acrobat (free use of a basic model, proprietary for advanced tools) or Microsoft (proprietary, but cheaper versions for research and education purposes). Generally, this is not done through the university as such directly, but to sell or licence the software to an existing outside company or to a spinoff established for this purpose. They then use the code to make their own code to be placed on the market. Universities rarely licence software to the market directly as they do not have the organisation for it (updates, maintenance, support, etc.). If the researcher considers the option of a spin-off, it is well worth noting this, as providing these obligatory services can be time-consuming.

Of note, these activities fall under the regulation for ancillary activities which need to be reported to HR and can only be engaged in when the researcher has received permission of the proper authority to do so (in most cases this will be the dean of the faculty).

Especially when commercialising software, it may be of interest to apply for a patent. The contract manager provides support for the researcher about commercialisation and is well positioned to discuss and provide guidance. The section below is intended to provide a brief primer to facilitate the discussions between research and support staff.

6.1 Proprietary licences and software

When the prime objective of the licence is to make the software available to a commercial party, potentially being a spin-off, a proprietary licence will in most cases be the best option. Drafting a proprietary licence is the exclusive domain of the contract manager (with necessary input from the researcher).

Before initiating talks with an interested party on a licence agreement, it is important to have two other documents in place: an establishment of ownership and a non-disclosure agreement. To start with the first document, it is important that the researcher has some sort of proof establishing ownership. This can be done either by a patent application or through a so-called “i-depot”. [11]

6.1.1 The Non-Disclosure Agreement

Under a Non-Disclosure Agreement (NDA), parties agree to keep everything they learn about each other confidential. This means that items shared under an NDA will not destroy a patent application, because the items were not made public.

The NDA can be unilateral (only one party sends information, the other simply receives) or reciprocal. In most cases, when initiating talks with a company, a unilateral NDA should be sufficient.

Generally, we insist on the use of our own template in these negotiations. Do not sign an NDA offered to you by the company without contacting the contract manager at your faculty first.

6.1.2 The i-depot

By filing an i-depot you will receive a certification by the government that on that particular date and time you disclosed your software to the government in a confidential manner. The certificate acts as legal proof that on that date and time you were in possession of the software (it does not establish you as the rightful

owner though). This depot is established by depositing one or more files (e.g. the source code, all manual(s) and hierarchy) to a maximum of 100 MB at the Benelux Office of Intellectual Property (BOIP). Just like a patent application, this process is initiated by filing an Invention Disclosure Form (IDF) at: https://inventions.tudelft.nl/inventor/invlogin.jsp (in this case, however, the researcher must tick the box “trade secret”).

6.1.3 Patents

When the software provides a novel technical solution to a technical problem in a technical field, it is possible to file a patent application on the software. As mentioned before, a patent will protect the function and as such, give stronger protection than copyright. If software developers and researchers want to file a patent, they have to fill in an IDF at TU Delft’s inventor portal: https://inventions.tudelft.nl/inventor/[3].

In order to be eligible for patent protection, software must provide a technical solution to a narrowly defined technical problem. For example, a self-learning algorithm that allows pattern recognition is not patentable. However, a self-learning algorithm that fully automatically recognises and brightly lights up malignant patterns in a breast-cancer scan can be patented with success.

It is also important to remember that only something that is “novel” can be patented. A patent application is no longer deemed novel if the software is already published, presented at a meeting, or even posted in a blog or on social media, before the patent application has been filed. Therefore, it is very important not to share your software online or with other people if a patent application is considered.

From a commercial perspective, the added advantages of a patent on software are:

- Not only the method (i.e. the algorithm) is covered by the patent, but also the product derived from that method (e.g. the diagnostic report).
- When the software is part of a high-tech component, a patent can be vested on the whole package (a “computer implemented-invention”).
- When further investment or co-development is needed for the software, a patent can be considered as an asset to facilitate attracting investors.

3 Please note that the portal is only accessible on campus, in a Citrix-environment, or through a secure eduVPN-connection.

There are also some disadvantages in patenting software to consider, in particular cost and time:

- Applying for a patent will come at an average cost of 35,000 EUR per patent. This money needs to be recouped when selling the software to a company or spin-off, along with the costs incurred by the university to “produce” the software (wages, experimental cost for validation)[4]. These costs tend to be challenging for many smaller companies. Therefore, in most cases a simple licence agreement will be preferable, leaving patents for the special cases (e.g. unique market, long-term benefit, large investors/companies).

- The length of time to obtain a patent (at least 30 months) is often too long for a high-paced environment such as the development of apps. During that time, the software itself may have become obsolete before the patent is even granted. In such cases, licensing might be a better option.

4 Costs relating to the acquisition and integration of the acquired business or operations into the Company.

Appendix 1. Definitions

In this document, “software” means a set of coded instructions designed to cause a computer, a machine or automatic data processing equipment to perform a task. For the purposes of these guidelines, the term software also includes (where relevant) the associated documentation (including but not limited to a manual), hierarchy, platforms, API’s and specific hardware (if any) required to run the aforementioned instructions.

Licence: “Licence”, “Export Licence” or “Software Licence” in this document mean a document that provides legally binding guidelines and requirements for the use and distribution of the software. Licences typically provide end users with the right to one or more copies of the software without violating copyright law. Open source licences typically grant this right to all of humankind rather than to a specified end-user.

Attribution: The requirement that the original authors must be credited for their work.

Derivative: In this document, “derivative” means a piece of code based upon one or more pre-existing software, such as a translation, an abridgement, a condensa-

tion, an expansion or any other form in which a code may be recast, transformed, or adapted. In order to constitute a derivative, the change needs to be significant. In general, in our view a dynamic link to another program, a database or library does not provide a derivative code, but a static link can (under circumstances) provide a derivative code (though the Open Source foundation advocates a stricter view– see the chapter covering GPL for a detailed discussion).

Dual-use goods: In this document, "dual-use goods" are goods, software and technology which are commonly used for civil purposes, but that can have military use or that can contribute to the production and deployment of weapons of mass destruction (WMD). For a non-exhaustive list see https://www.rijksoverheid.nl/onderwerpen/exportcontrole-strategische-goederen.

Inventor: A natural person who has made a substantial, significant intellectual contribution to the development of Intellectual Property is regarded as an inventor. To qualify as an inventor, an individual must have contributed sufficiently to the software to take public responsibility for its content and the individual's intellectual contribution must be critical to its main conclusions. When there is disagreement, an inventor can or should be able to identify, demonstrate and provide documented evidence of this inventive contribution to specific claims in a patent application.

Intellectual Property (IP): This means all protected forms of intellectual property rights including patents, utility models, trademarks, trade or business names, database rights, service marks (be it either registered or unregistered), copyright, right to mask work of integrated circuit, copyright of design, know-how and other proprietary knowledge and information, rights protecting goodwill and reputation and the applications of the protected forms of intellectual property rights or having equivalent effect and all rights of licence and consent in respect to any of these aforementioned rights.

Permissive (software) licences: They have minimal requirements about how the software can be redistributed. They do not block appropriation, nor do they force the redistributor to open the modified source code. By its nature, code written under a permissive licence can generally be incorporated in code written by a more restrictive licence, but not the other way around. Another caveat of permissive licences is that the party that originally made the licence can change a permissive licence into a non-permissive or even proprietary licence. Therefore, as a user of permissive licences it is important to check the current status when about to do something that uses a program offered under a permissive licence. Examples of permissive licences are MIT and BSD.

Appendix 2. Types of licences pre-approved by TU Delft

There are several types of licences made available to TU Delft employees for application without further approval or permission beyond the requirements embedded in these guidelines and described in detail below.

1. CC0 – "No rights reserved"

Creative Commons [12] licences are not appropriate for software. The notable exception is CC0, which is commonly deployed to note something (including software) falls in the public domain and is freely available to all.

To apply this, simply copy this image:

Please note that when your software contains other types of art (pictures, movies, excerpts from Wikipedia etc.) these may be covered by a CC licence. Please consult with your data steward.

The link to the legal text of the licence is here: https://creativecommons.org/publicdomain/zero/1.0/legalcode

2. MIT

The MIT License is a permissive free software licence originating at the Massachusetts Institute of Technology (MIT) in the late 1980s. As a permissive licence, it puts only very limited restriction on reuse and has, therefore, reasonable licence compatibility. The MIT licence permits reuse within proprietary software provided that all copies of the licensed software include a copy of the MIT License terms and the copyright notice. The MIT licence is also compatible with many copyleft licences, such as the GNU General Public License (GPL); MIT licensed software can be integrated into GPL software, but not the other way around.[13]

The MIT licence ensures that the original authors must be credited. Furthermore, the MIT licence ensures that the software is provided "as is", without any warranties or guarantees. As a consequence, the researcher cannot be held liable for any damage or claim regarding your software. These are the minimal requirements of any open source licence we can accept.

When the researchers wish to apply the MIT licence, the researcher must insert the full text of the licence found here: https://opensource.org/licenses/MIT.

3. BSD

BSD licences are a family of permissive free software licences, imposing minimal restrictions on the use and distribution of covered software. The BSD licence is a simple licence that merely requires that all code retain the BSD licence notice if redistributed in source code format, or reproduce the notice if redistributed in binary format. Like the MIT licence, the BSD licence does not require that source code be distributed at all. The BSD licence is also similar to the MIT licence in that attribution is secured and disclaimers of warranty are maintained.

There are three types of BSD licences (0, 2 and 3 –clause). ***TU Delft endorses the use of the 3-Clause BSD License***.

The 3-Clause BSD License is more detailed than the MIT licence and contains a clause restricting use of the names of contributors for endorsement of a derived code without specific permission.

When the researchers wish to apply the 3-Clause BSD License, the researcher must insert the full text of the licence found here: https://opensource.org/licenses/BSD-3-Clause.

4. Apache

The Apache License is a permissive free software licence written by the Apache Software Foundation (ASF). It does not require a derivative of the software, or modifications to the original, to be distributed using the same licence. But it still requires application of the same licence to all unmodified parts. In every licensed file, original copyright, patent, trademark and attribution notices must be preserved. Furthermore, in every licensed file that has been changed, a notification must be added stating that changes have been made to that file.

If a copyright notice is included as part of the original software (for instance, in a README text file), then derivative code based upon this software must include a readable copy of these notices within a notice text file either distributed as part of the derivative code, within the source code or documentation, or within a display generated by the derivative code (wherever such third-party notices normally appear).

The Apache License 2.0 [14] has a unique feature that makes sure that the user does not have to worry about infringing any patents by using the software. The user is granted a licence to any patent that covers the software. This licence is terminated if the user sues anyone over patent infringement related to this software. This condition is added in order to prevent patent litigation on software. That's why the Free Software Foundation recommends Apache License v2.0 over other permissive licences.

In the licence text these clauses are:

- Giving a free licence: "each Contributor hereby grants to The researcher a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent licence"

- Preventing patent litigation: "If The research institute patent litigation against any entity (…), then any patent licences granted to The researcher under this License for that Work shall terminate as of the date such litigation is filed"

In order to apply the Apache licence to software it is necessary to add the text found in the Appendix of the following link: https://opensource.org/licenses/Apache-2.0.

It is recommended that a file or class name and description of purpose of the software be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

5. GPL

The GNU General Public License (GPL) is a widely-used free software licence. It is a copyleft licence, which means that any derivative code must be open source and distributed under the same or equivalent licence terms. This is a significant difference with the permissive free software licences, which are discussed above. Any licensee who adheres to the terms and conditions is given permission to modify the software, as well as to copy and redistribute the software or any derivative version. The full text of the licence can be found at: https://opensource.org/licenses/GPL-3.0

Historically, the GPL licence family is the first copyleft licence and has been one of the most popular software licences in the open source software domain. In 2007, the third version of the licence (GPL-3.0) was released to address some perceived problems. In particular, the third version addresses issues in the area of 'tivoization', new legislation like the European Union Copyright Directive (which makes it a crime to write or share software that can break Digital Restrictions Management; DRM) and the advent of (discriminatory) patent deals in the open source software community. To keep the licence up to date, the GPL licence includes an optional "any later version" clause, allowing users to choose between the original terms or the terms in new versions as updated by the Free Software Foundation.

In order to apply the GPL-3.0 licence to software it is necessary to add the text found at the end of this page: https://opensource.org/licenses/GPL-3.0.

5.1 Incompatibility

Please note that although the Free Software Foundation states that the Apache licence v2.0 is compatible with GPL-3.0, in reality it is not. When the researcher applies an Apache licensed project in a GPL-3.0 licence software it becomes a derivative of some GPL-3.0. Thus, the Apache licensed software would have to be distributed under GPL-3.0. This, at the same time, is incompatible with the requirement that all Apache licensed software must be distributed under the Apache License 2.0. Similarly, GPL-3.0 is unilaterally compatible for materials under Creative Commons Attribution-ShareAlike (mostly drawings or clips) to be remixed into the GPL-licensed materials (mostly software), but again not *vice versa.* This is especially encountered in niche use cases like mixing a game engine (GPL) with game scripts (CC-BY-SA).

5.2 Dynamic linking

On linking with GPL, the Free Software Foundation asserts that an executable which uses a dynamically linked library is indeed a derivative. This does not however apply to separate programs communicating with one another. The mere act of communicating with other programs does not, by itself, require all software

to be GPL; nor does distributing GPL software with non-GPL software on a single information carrier provide such a link. However, minor conditions must be followed that ensure the rights of GPL software is not restricted. In the Frequently Asked Questions (FAQs) from the Free Software Foundation the researcher can find the answer to the question *"What is the difference between an "aggregate" and other kinds of "modified versions"?"* [15], which describes to what extent software is allowed to communicate with and be bundled with GPL licensed programs.

5.3 Commercial and private use

Software under the GPL may be run for all purposes, including commercial purposes or as a tool for creating proprietary software (GPL-licensed compilers). Users or companies who distribute GPL-licensed code (e.g. software), may charge a fee for copies or give them free of charge. This distinguishes the GPL from shareware software licences that allow copying for personal use but prohibit commercial distribution, or proprietary licences wherein copying is prohibited.

In purely private use (with no sales and no distribution) the software code may be modified and parts reused without requiring the source code to be released. For sales or distribution, the entire source code needs to be made available to end users, including any code changes and additions. The GPL additionally states that a distributor may not impose "further restrictions on the rights granted by the GPL". This forbids activities such as distributing the software under a non-disclosure agreement or contract.

However, software running as an application program under a GPL-licensed operating system such as Linux is not required to be licensed under GPL or to be distributed with source-code availability—the licensing depends only on the used libraries and software components and not on the underlying platform.

5.4 Linking to Libraries – FSO point of view

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, the researcher may consider it more useful to permit linking proprietary applications with the library. If this is what the researcher wants to do, use the GNU Lesser General Public License instead of this License. The GNU Lesser General Public License (LGPL) was created to have a weaker copyleft than the GPL, in that it does not require its own custom-developed source code (distinct from the LGPLed parts) to be made available under the same licence terms.

Therein the Free Software Organisation provides the following consideration: "Why the researcher shouldn't use the Lesser GPL for your next library" [16].

6. EUPL

The European Union Public Licence (EUPL) is a free software licence that has been created and approved by the European Commission. Software, mainly produced by European administrations, has been licenced under the EUPL since the launch of the European Open Source Observatory and Repository (OSOR) in October 2008, now part of Joinup collaborative platform. It is also for this specific reason that the EUPL is included in this list. Its latest version, EUPL v1.2, was published in May 2017. It is, incidentally, the only licence established by law (OJ 19/05/2017 L128 p. 59–64 to be precise).

Importantly, the licence is available in all 23 official languages of the European Union and all linguistic versions have the same legal validity. The licence is also vetted to conform with all existing (civil law) copyright laws of the Member States of the European Union. As a consequence, the limitations of liability and warranty have been precisely defined, rather than “to the extent allowed by law.

6.1 Coverage and Compatibility

EUPL v1.2 has broader coverage than the previous versions. Currently, it covers “the work” (meaning copyrighted code) and not exclusively “the software”. Therefore, it is easier to apply the EUPL v1.2 to related documentation, handbooks, standard specifications, etc.

It also has broader compatibility: the software itself (copies or modifications/improvements) will stay covered by the EUPL without possibilities of re-licensing by recipients, but it may also be merged into a new and larger code with other software components covered by compatible licences. When needed and for avoiding licence conflicts, this other derivative can then be distributed under the compatible licence. The list of compatible licences includes both the GPLv3 and the Creative Commons licence CC-BY-SA. The purpose of this compatibility list is not to endorse or recommend the listed licences: it is finding interoperable solutions to possible licence conflicts. This is the reason why the list includes SA (share alike) or copyleft licence, and not the most permissive ones (as these do not pose licensing conflicts).

EUPL v1.2 (article 9) provides more flexibility concerning the additional agreements: any additional provision that is not in contradiction with the licence is valid, including the selection of a specific applicable law, of a specific arbitration court, etc.

The purpose of the EUPL is to encourage European public administrations to embrace the Free/Open Source model to valorise their software and knowledge. Some applications developed in the framework of the IDABC programme, such as

Circabc, or Eusurvey have already been licensed under the EUPL in 2007. Other European Institutions are bound to follow under the new European Committee and the Horizon Europe programme.

In order to apply the EUPL licence to your software the application of this boiler-plate declaration is sufficient:

Copyright [yyyy] [name of copyright owner] Licensed under the EUPL

The full text of the EUPL licence can be found at https://opensource.org/licenses/EUPL-1.2.

7. GPL and EUPL: a comparison

For convenience, the table below lists some of the differences between the strong copyleft licences EUPL and GPL. This table is for information purposes only and is by no means exhaustive or to be taken as an endorsement of one over the other.

	EUPL	GPL
Type	Strong copyleft	Strong copyleft
OSI-approved	Yes	Yes
Legal language	22 linguistic versions Valid "as is" in any of the 22 languages	English Need "sworn translator" in court
Applicable law	Law of the European Union country of the Licensor, otherwise Belgium	Local law that most closely approximates an absolute waiver of all civil liability
Legal complexity	Based on civil law ("good faith principles") short text, provisions fix principles	Based on common law ("pacta sunt servanda") Long, complex text, provisions address specific and technical issues in detail
Liability & warranty	Fully in line with EU law: Consumer protection & information practices. Warrants ownership copyright. Liability in case of wilful misconduct and/or damages to persons.	Claims a general "catch-all" exclusion of warranty and liability. But not accepted by all jurisdictions, thus only "to the extent permitted by applicable law".
Competent court	Where the Licensor resides. Other agreements are permitted.	Court is not designated.
Licence conflicts	No, mostly not. Pre-defined list of "interoperable licences" (incl. GPL)	Yes.
Linking	Recital 15 specifically authorises linking without copyright infringement	"Linking statically or dynamically with other modules is making a combined code. Thus, the terms and conditions of the GNU General Public License cover the whole combination."
Specific items	Covers "Software as a Service" meaning that if an internet service provider modifies the licensed software to distribute online services (as Google does), this is "software distribution".	Covers "Tivoization" meaning that it prevents hardware providers from locking a protection which will not allow modified software to run on their hardware. Note: Many consider that "Tivoization" is related to hardware protection (i.e. against theft, counterfeiting) and not software appropriation!

Appendix 3. Relevant contact information

Intellectual Property contact information

All inventions should be reported at: https://inventions.tudelft.nl/inventor*

*To access this link, the user must be connected to the TU Delft Intranet using an eduVPN connection.

The IP team of TU Delft can be found at:
Valorisation Centre
Building 26, C tower, 3rd floor
Van der Burghweg 1, 2628 CS Delft
PO Box 5, 2600 AA Delft
Email: patent@tudelft.nl

Delft Enterprises

Ronald Gelderblom
Investment Director
Email: r.gelderblom@tudelft.nl
Phone: (+31) 015 278 4350

Legal Affairs (incl. contract management and privacy/GDPR)

Rianne van den Bogerd
TU Delft / Directie Legal Services
Postbus 5, 2600 AA Delft
Gebouw 23, Stevinweg 1, k. 5.24
Email: r.vandenbogerd@tudelft.nl
Phone: +31 (0)15 27 87003/06 41643772

ICT-related Horizon Europe funding

For questions concerning European funding of ICT-related research projects please contact:

Jan Schiereck
EU research grant advisor
TU Delft / Valorisation Centre / Section European Research Funding
Van der Burghweg 1 (building 26a) | 2628 CS, Delft | PO Box 5, 2600 AA Delft
Email: E j.d.schiereck@tudelft.nl
Phone: +31 (0) 6 810 618 64

Appendix 4. References

[1] https://pure.tudelft.nl/portal

[2] https://www.tudelft.nl/en/library/tu-delft-open-science/programme-open-science

[3] https://choosealicense.com/licenses

[4] https://eur-lex.europa.eu/legal-content/NL/TXT/PDF/?uri=O-J%3AL%3A2019%3A338%3AFULL&from=NL

[5] Katz DS, Chue Hong NP, Clark T et al. Recognizing the value of software: a software citation guide [version 2; peer review: 2 approved]. F1000Research 2021, 9:1257 (https://doi.org/10.12688/f1000research.26932.2)

[6] https://creativecommons.org/share-your-work/public-domain/cc0

[7] https://choosealicense.com/licenses/bsl-1.0

[8] https://choosealicense.com/licenses/unlicense

[9] https://choosealicense.com/no-permission

[10] https://en.wikipedia.org/wiki/Derivative_work

[11] https://www.boip.int/en/entrepreneurs/ideas/your-idea-in-an-i-depot

[12] https://creativecommons.org

[13] https://en.wikipedia.org/wiki/MIT_License

[14] https://www.apache.org/licenses/LICENSE-2.0

[15] https://www.gnu.org/licenses/gpl-faq.html#MereAggregation

[16] https://www.gnu.org/licenses/why-not-lgpl.html

Colophon

Credits:

These guidelines were originally written by Merlijn Bazuine, Valorisation Centre with contributions and feedback from (in alphabetical order):

Anton Akhmerov – Faculty of Applied Sciences

Julie Beardsell – Information & Communications Technologies (ICT)

Rianne van den Bogerd – Legal Services

Susan Branchett – Information & Communications Technologies (ICT)

Alastair Dunning – TU Delft | Library

Meta Keijzer-de Ruijter – Information & Communications Technologies (ICT)

Maria Marques de Barros Cruz – TU Delft | Library

Paula Martinez Lavanchy – TU Delft | Library

Margot Spaargaren – Legal Services

Marta Teperek – TU Delft | Library

TU Delft Data Stewards:

Heather Andrews, Faculty of Aerospace Engineering

Nicolas Dintzner, Faculty of Technology, Policy and Management

Esther Plomp, Faculty of Applied Sciences

Contact info:

Delft University of Technology

Intellectual Property, Valorisation Centre

patent@tudelft.nl

Design:

Carla Feijen

Photography:

Doris Aschenbrenner & Meng Li

Faculty of Industrial Design Engineering

Delft University of Technology