Respecting Open Source Licenses !

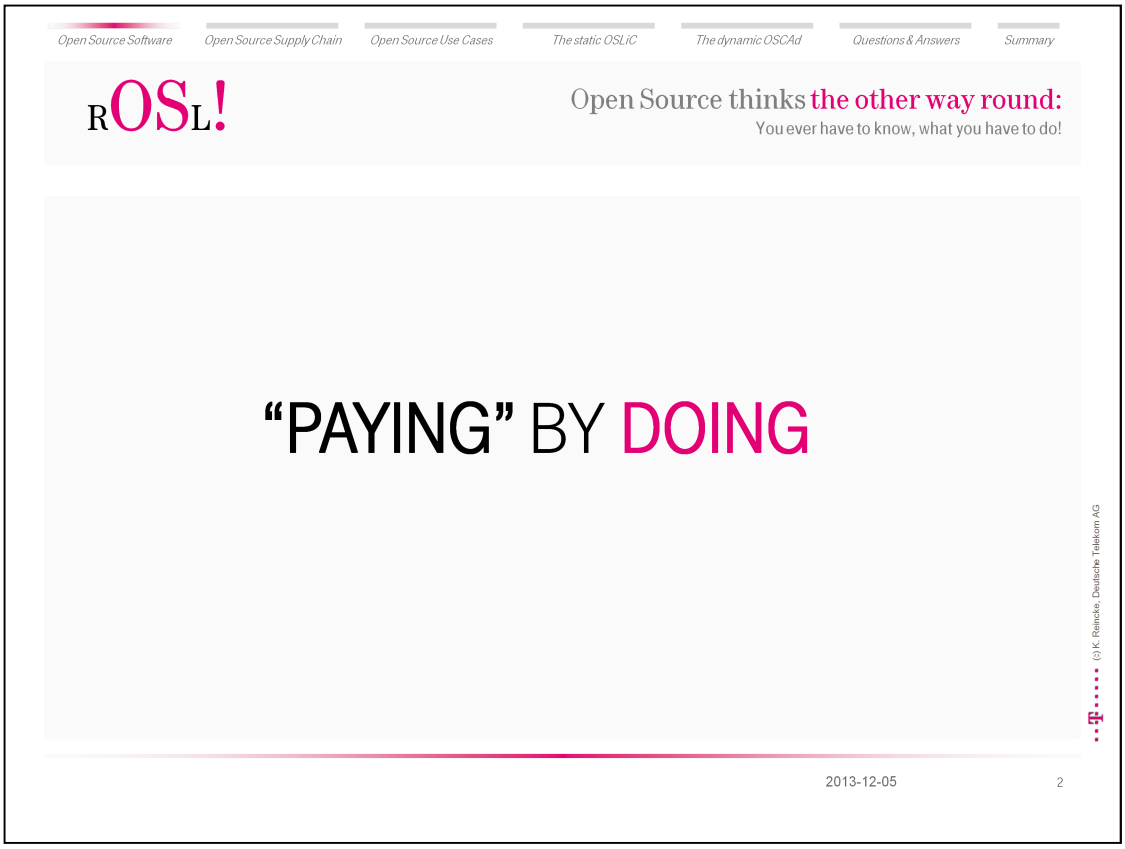
OSCAd & OSLiC

Dealing with the open source use cases based on the free tools
Open Source License Compendium & Open Source Compliance Advisor

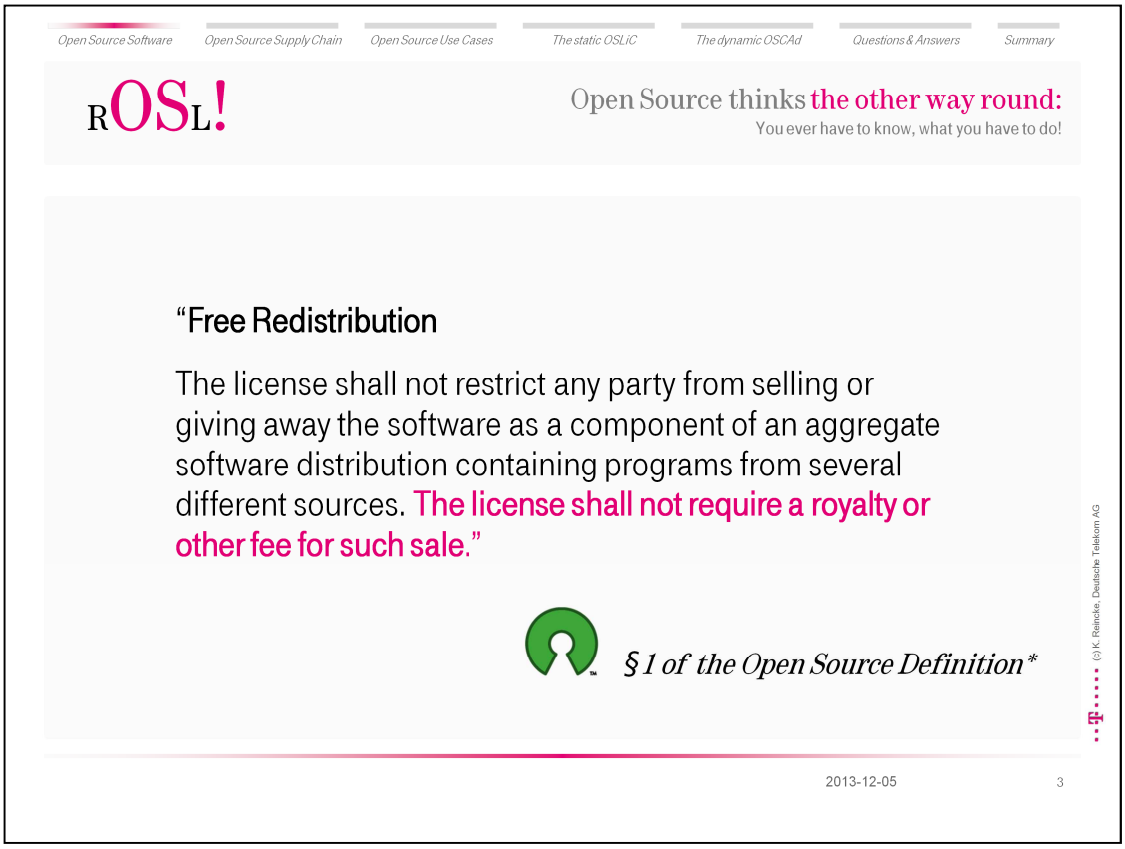
Open Source CompLianCe in Deutsche Telekom

Karsten Reincke

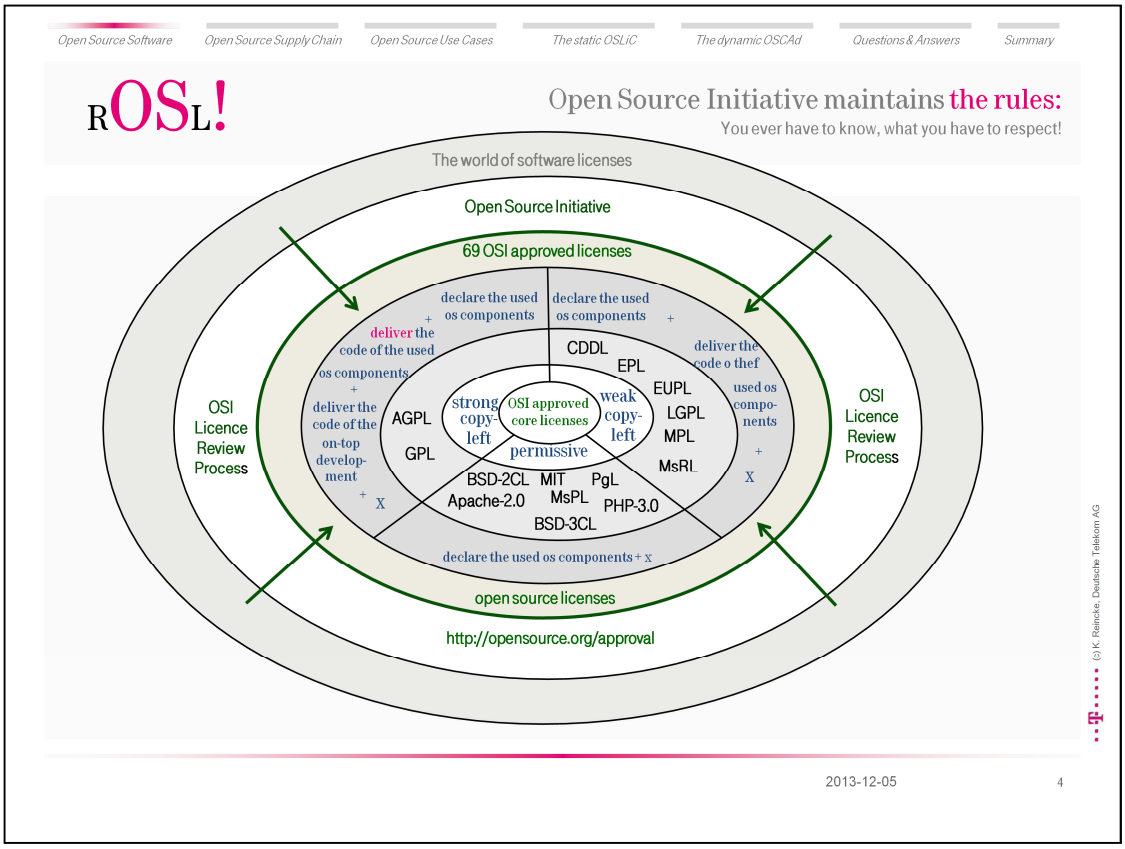
Erleben, was verbindet.



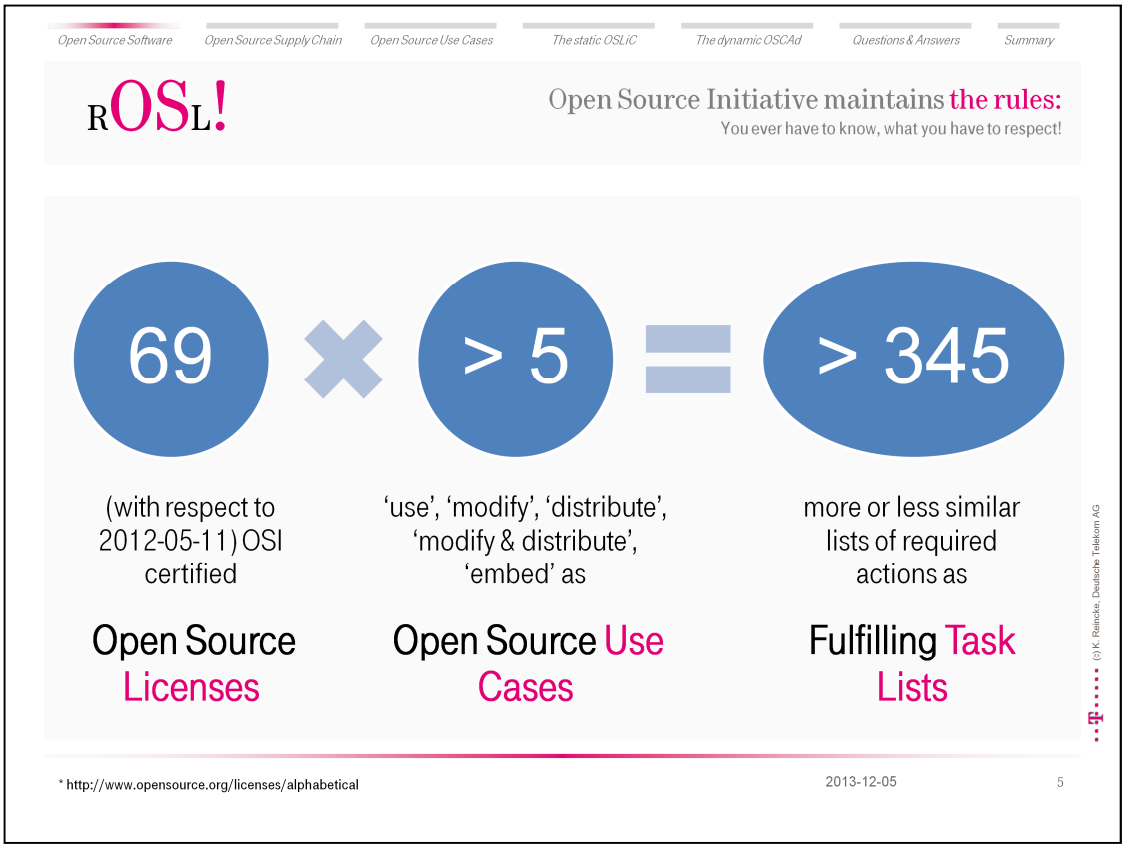
- First, let us start our analysis with the well known simple statement: We can’t buy the right to use open source software. The open source software thinks the other way round: you get the right to use, to modify and/or to redistribute the software by doing what is required by the open source licenses. Thus, the management of companies using open source software has to know: open source software follows the principle ‚paying‘ by doing.



- Second, let us remind ourselves that this fact refers to the OSD the ‚Open Source Definition' - hosted and maintained by the Open Source Initiative. The first paragraph of the OSD forbids to request for a license fee. Hence, we can't buy any open source software.

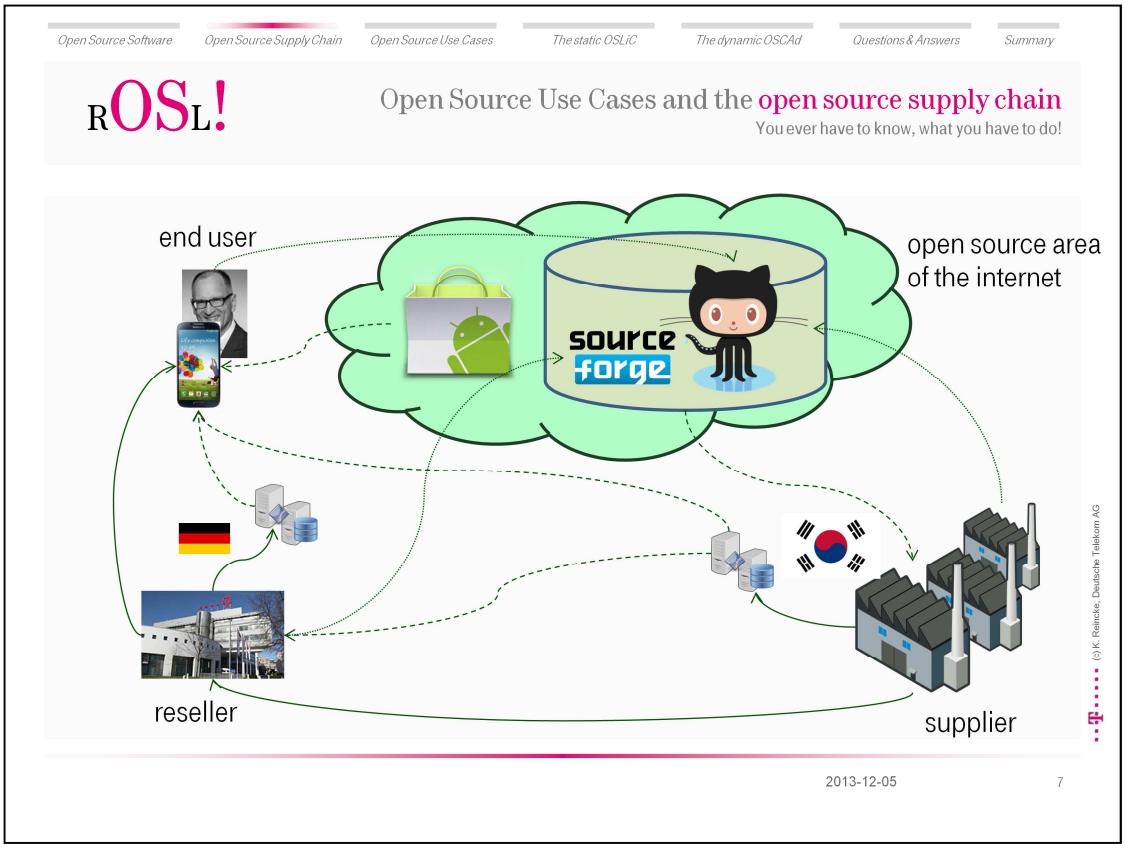


- Third, let us shortly consider, that the Open Source Initiative separates the open source licenses from the other software licenses by reviewing the licenses on the base of the Open Source Definition. This OSD contains 10 criteria, which all open source licenses have to fulfill. But on top of the 10 necessary conditions, the licenses may offer and require additionally aspects. And the may differ in the way they implement these criteria. So, inside of the set of OSI approved licenses, we can cluster these licenses by some of their other features. One of the most known method to group the open source licenses internally is a taxonomy based on the categories ‘permissive licenses’, ‘weak copy left licenses’, and ‘strong copy left licenses’.
- But consider also, that there exist many licenses which seem to be open source licenses, but which (still) are not approved by the OSI. Moreover, many of them can never be approved because they implicitly violate one of the 10 necessary conditions. A famous example is the requirement, that the licensed software may only be used for good purposes.
- Additionally, one should also consider, that some licenses like the BSD and the MIT license are license templates. Therefore, sometimes a license seems to be not approved by the OSI because it is not listed. But in fact, it is ‘only’ an instance of the BSD which got an own name and which therefore should and could be treated as BSD license.



- Based on these initial statements, we can point out to a specific challenge: There are so many open source licenses (69) which require different tasks for acting in accordance to the licenses. And these fulfilling working steps must consider the different contexts (>5): do I only use the software for myself. Or do I hand over it to any other person? And have I modified it? Hence we get more than 345 possible fulfilling task lists.

- So, we know, what an Open Source License is. But for determining the real meaning of an 'Open Source Use Case' we have to consider the open source supply chain?



- The typical open source supply chain contains several nodes, moreover, it seems to be not a single chain, but supply clews, a bundle of supply ways:

  LOOP:

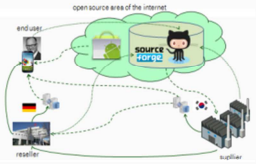
  - The true supplier receives his open source applications and source from open source repositories or from any other point of the net.
  - This supplier may install the software on one of his internet servers.
  - Or he me install it into pieces of hardware which he delivers to the reselling company.
  - The reseller my install the software on of his internet servers.
  - Or he may deliver the hardware containing open source components to his customer, the end user.
  - Additionally, this end user may mostly improve this bought open source based product by other open source based application offered by the application market place.
  - Finally, each of these players may also expand the open source internet by new or improved open source software which then GOTO LOOP

ROSL!

Defining the open source use cases

You ever have to know, what you have to do!

TYPE → proapse
TYPE → snimoli



a) independently executable program, application, or server

or

b) snippet, module or library which must be embedded

At least one license (GPL) imposes different conditions for the compliant use of embedded not autonomous software components.

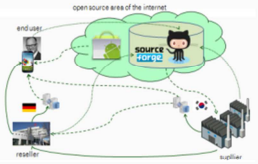
- (Open source) software can either be an independently executable program, application, or server. Or it can be an embeddable snippet, module or library, which itself needs a program, application, or server into which it must be embedded for being executed.
- This This dimension of distinctive features is relevant for fulfilling the license conditions because all licenses with strong copy left effect require that the overarching software complex must be licensed under the same license as the software which is embedded into the larger software complex.
- Note: Even scripting languages offer main clauses on the one side and on the other side methods to include files which can be viewed as libraries or modules even if the process of including is nothing more than a textual inclusion. Hence, this classification is applicable to traditional programming languages like C, C++, to pure interpreter languages like php or python, and to mixed languages like java.

rOSL!

Defining the open source use cases

You ever have to know, what you have to do!

STATE → unmodified
STATE → modified



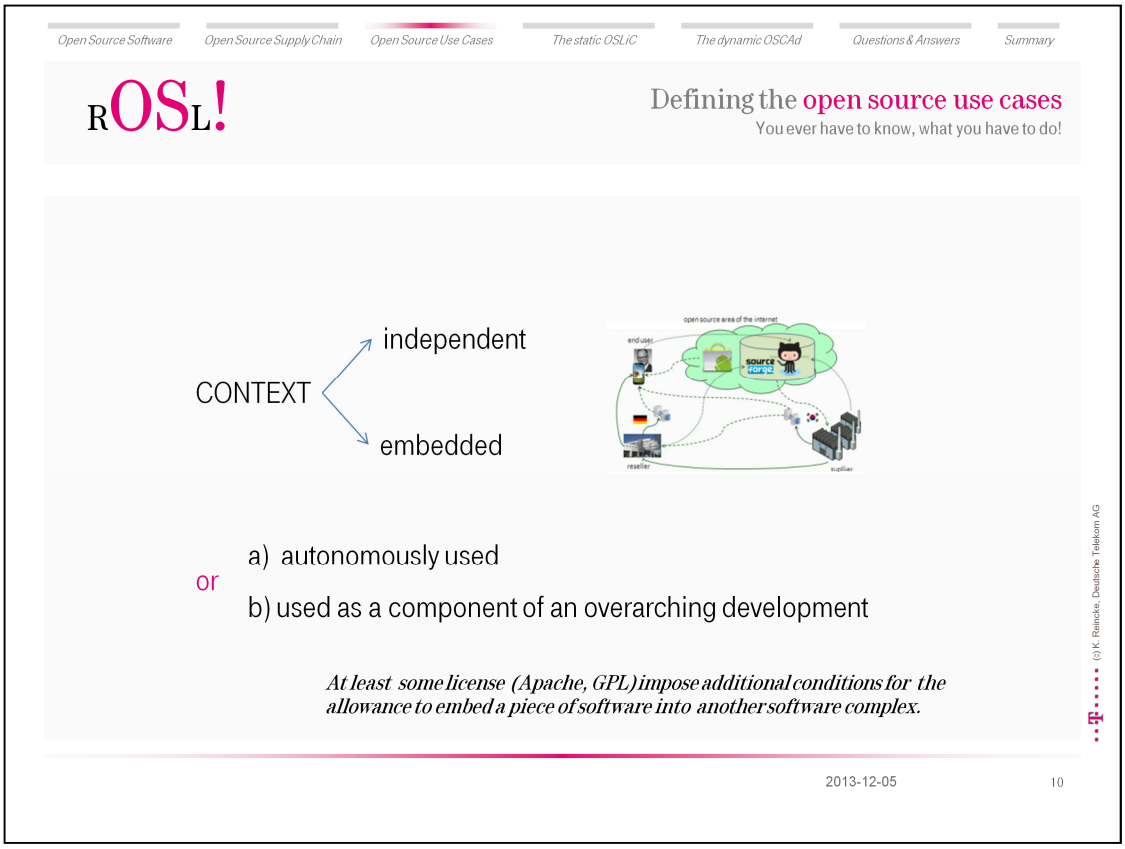
or

a) use the software in the form you received it

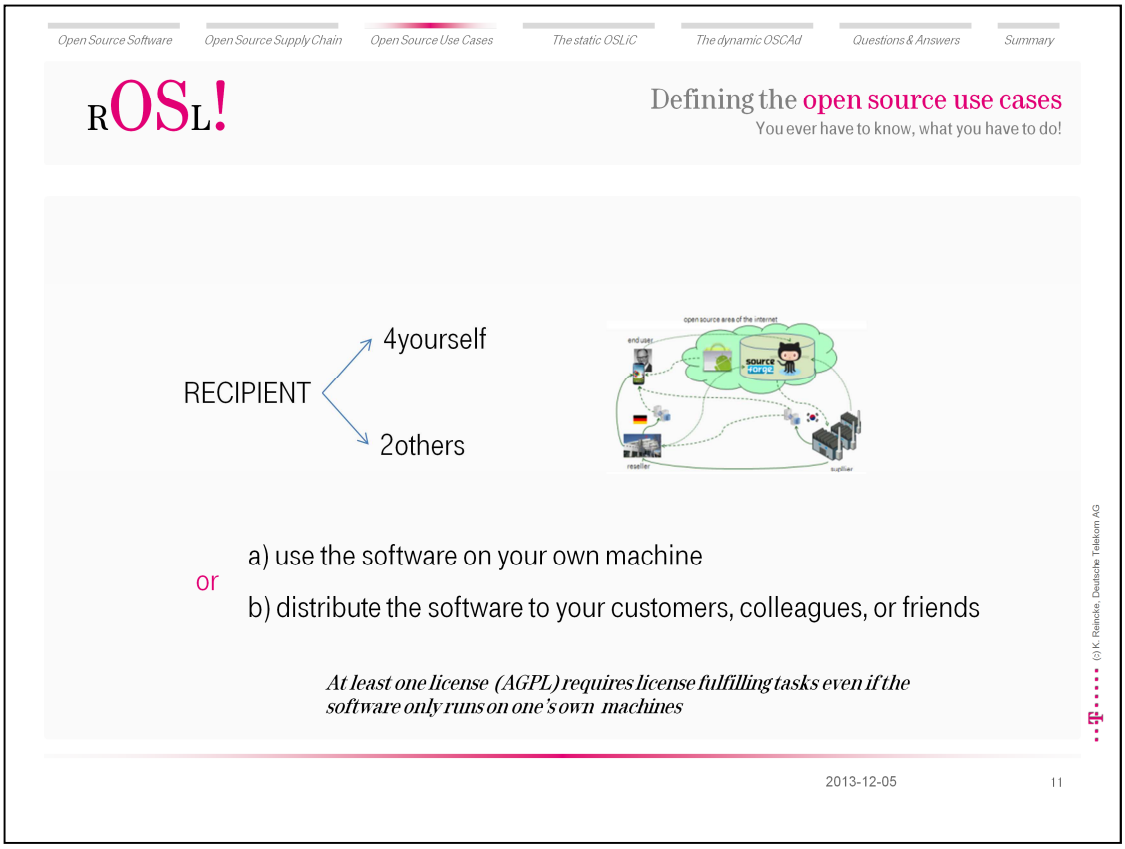
b) improve the software before you are using it

At least some licenses (e.g. EPL, MPL) want the modified open source software to be treated differently: the modifications shall be indicated

- Then we can use and/or distribute open source software in the form we got the original or we can modify that software before we are using and/or distributing it.
- This dimension of distinctive features is relevant, because many licenses require that all modifications must be indicated – a condition, which can be ignored if one has not modified the received and (re-) distributed software.



- On the one side, we can use the received open source software as an autonomous unit without taking it as a basis for an overarching work of software development. Or, on the other side, we can 'insert' the received piece of open source software into our development as one of its components.
- This dimension of distinctive features is relevant because at some licenses require different conditions if the software is embedded into a larger overarching complex or not.
- Note (1): There is a link between the difference of being a program, application, or server and of being a snippet, module, or library on the one side, and the difference of being used independently or being embedded on the other side – even if there does not exist an 1:1 relation:
  - On the one side an independently executable program, application, or server can not be integrated into an overarching program as one of its component – because such a complex would contain two main clauses.
  - On the other side, a snippet, module, or library can be embedded into such an overarching program as one of its component. But such a snippet, module, or library can also be distributed to third parties without being embedded into programs. Nearly each GNU/Linux distribution like UBUNTU, Red Hat, SuSE contains such 'embedded' elements for enabling the receivers of the distribution to start a development bases on these 'snimolis'. And distributing open source software is a kind of use. So, snippets, modules, or libraries can also be used independently, without being embedded into an overarching programming work.
- Note (2): Even a permissive license like the Apache license wants a piece of software, which is embedded, to be treated in another way than the same piece of software shall be treated if it is not embedded. On the one side, a program, which is using an Apache licensed library, must indicate that it is using the Apache licensed library and it must show the notice file the library contains. On the other side, if one distributes an Apache licensed library and a program which does not use



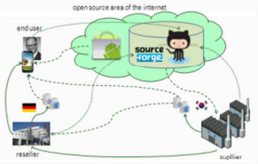
- On the one side, we can distribute open source software to third parties – like our end users or like our resellers. On the other hand, we can install and use the software only on our own computers positioned in our own data centers and maintained by our own employees.
- This dimension of distinctive features is relevant because at least one license requires license fulfilling tasks even if the open source software is only installed on our own machines and not distributed to third parties.
- Note:
  - at least in Germany, it is not finally determined whether handing over a piece of open source software to external data center operator which works on the base of a contract with the handing over company is nevertheless a kind of software distribution (2others) or not (4yourself).
  - But it should be uncontroversial, that if an employee (of the 'handing over' company) himself pushes the software into the cloud (offered by an external data center) and if no other people can access to maintain the running instance, then one could reasonably say that the company of the administrating employee uses the software only for itself.

ROSL!

Defining the open source use cases

You ever have to know, what you have to do!

FORM → binaries
FORM → sources



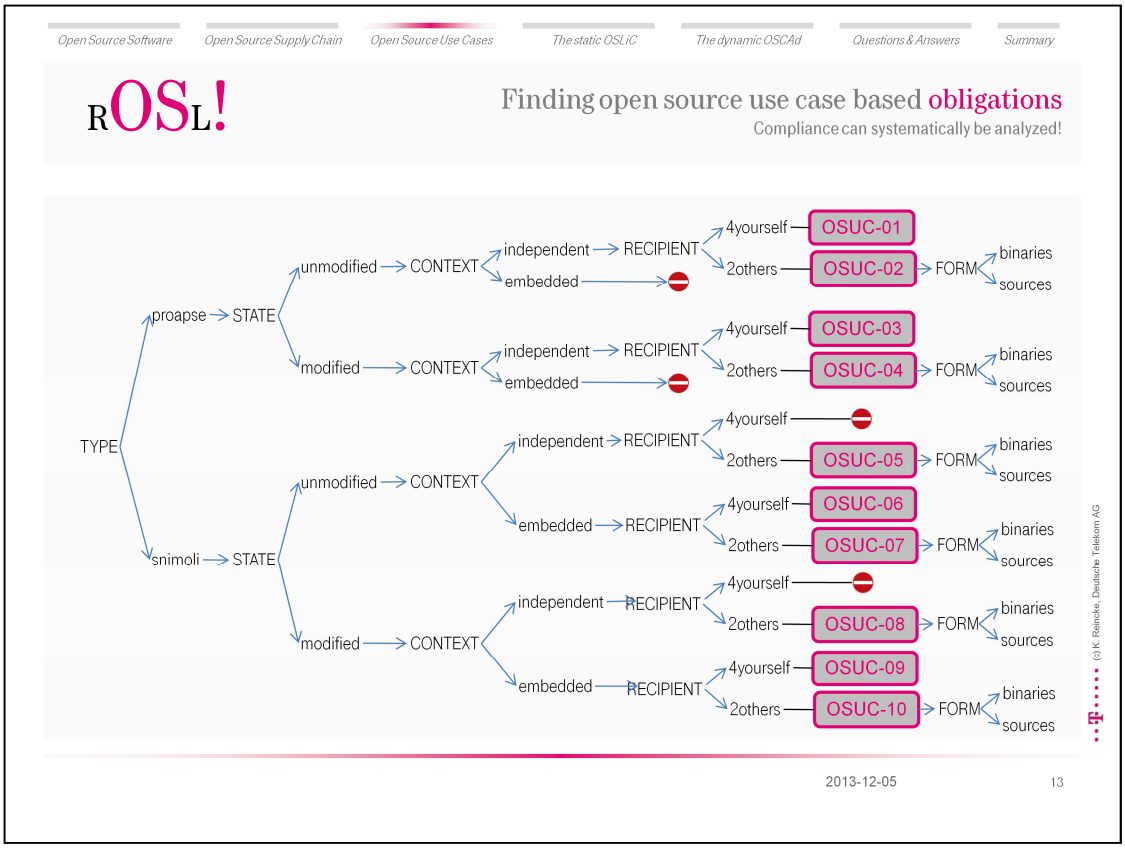
or

a) distributes compiled binaries

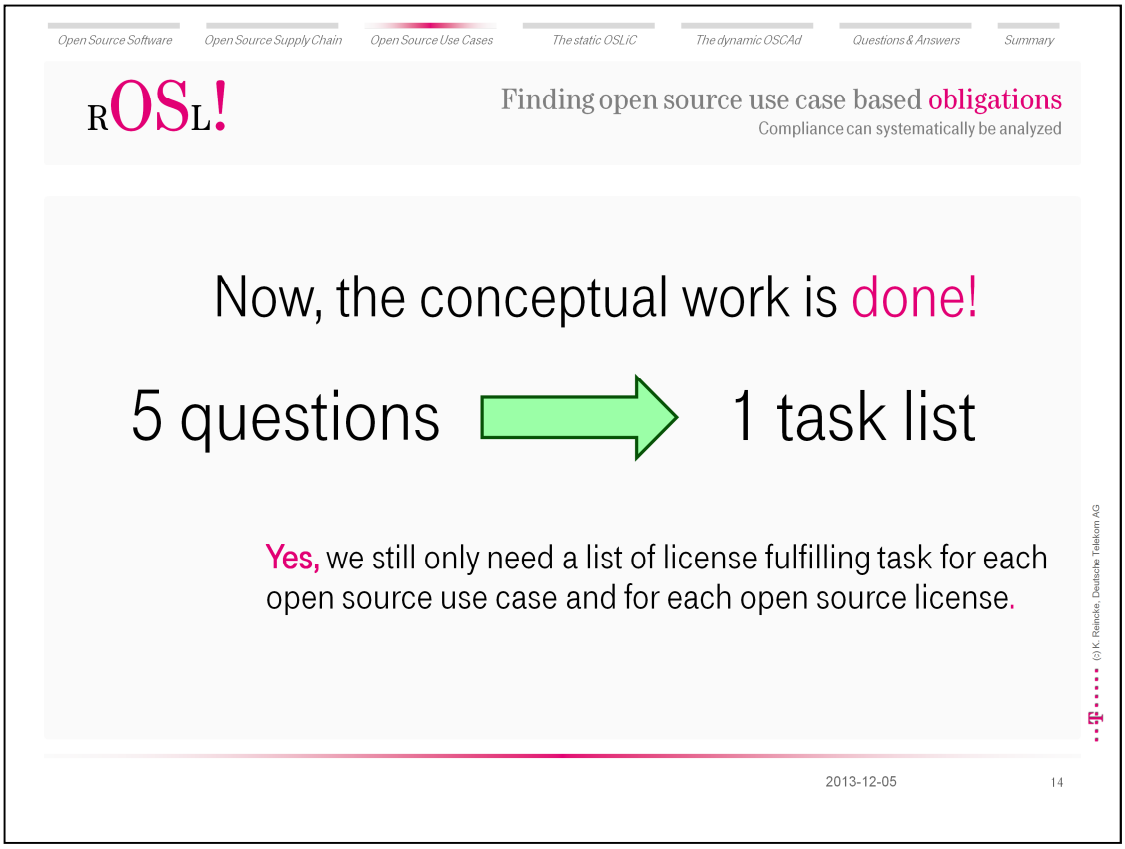
b) distribute pure sources

Many licenses (e.g. weak and strong copy left) impose different conditions for distributing the two forms compliantly

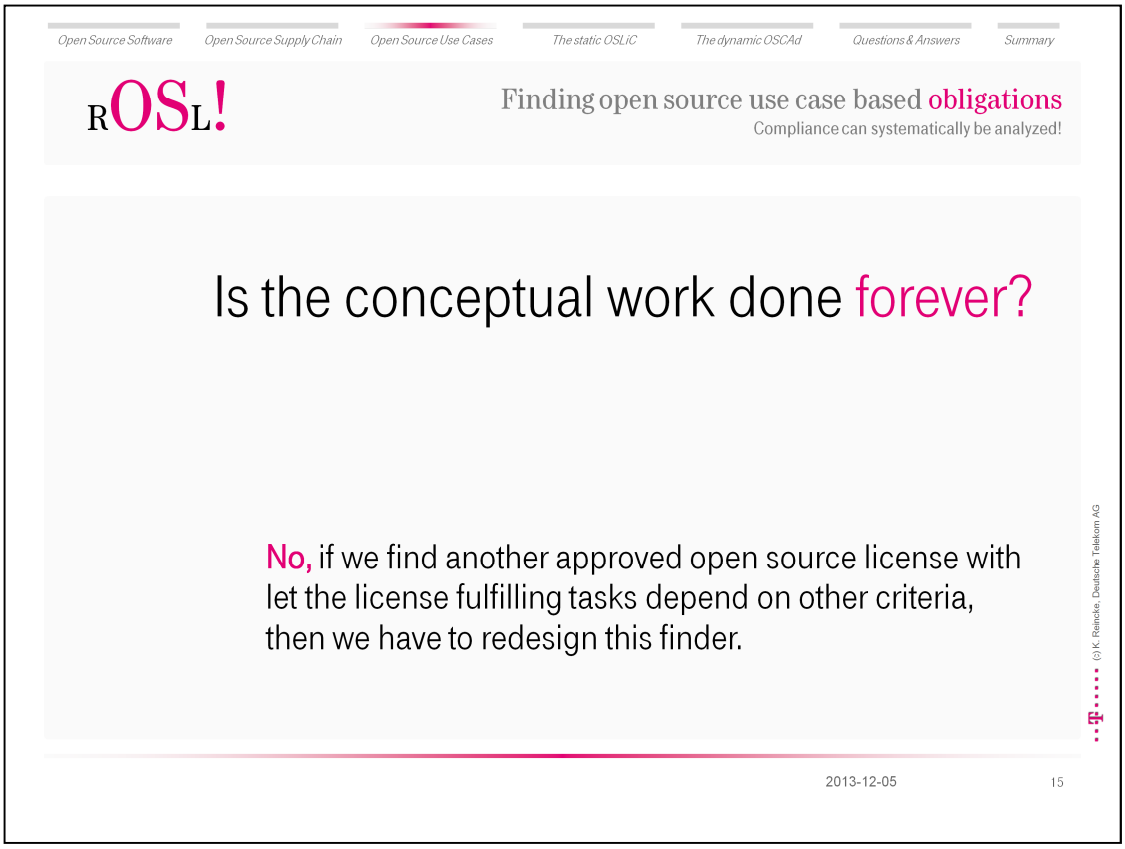
- Finally, we are able to distribute the open source software as a package of binaries or as a package of sources.
- This dimension of distinctive features is relevant because nearly all licenses treat the distribution of the source code in another way than distributing the binaries. Even many permissive licenses want to preserve the act of licensing if one distributes the source code.



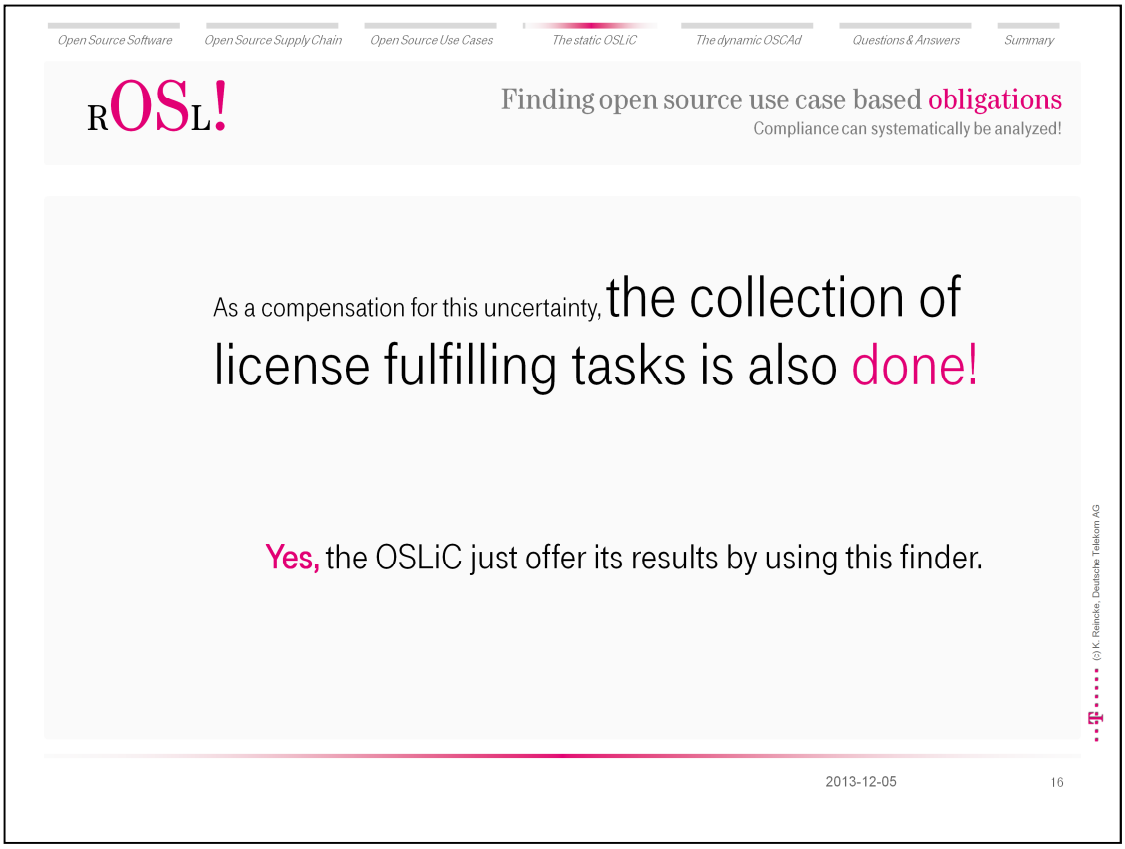
- If one now combines these features (criteria) into a tree, then one creates something like a finder for sets of features. And some of these sets might get a name, for example the name ‘open source use case + number’, or shortly: OSUC-01 to OSUC-10.
- As one can see, there exist some combinations of features which do not have an extension:
  - As we noted above, an autonomous program, application, or server can not be used as embedded component, because a software complex may not have more than one main clause.
  - On the other hand, a snippet, module, or library, which is used only for yourself and therefore which is not distributed to third parties, can not reasonably be used independently. If one ‘uses’ a snippet, module, or library without distributing it and without embedding it into an overarching software unit, then indeed this library is nothing else then an unused file on the file system. So, one does not use this library, snippet ore module.



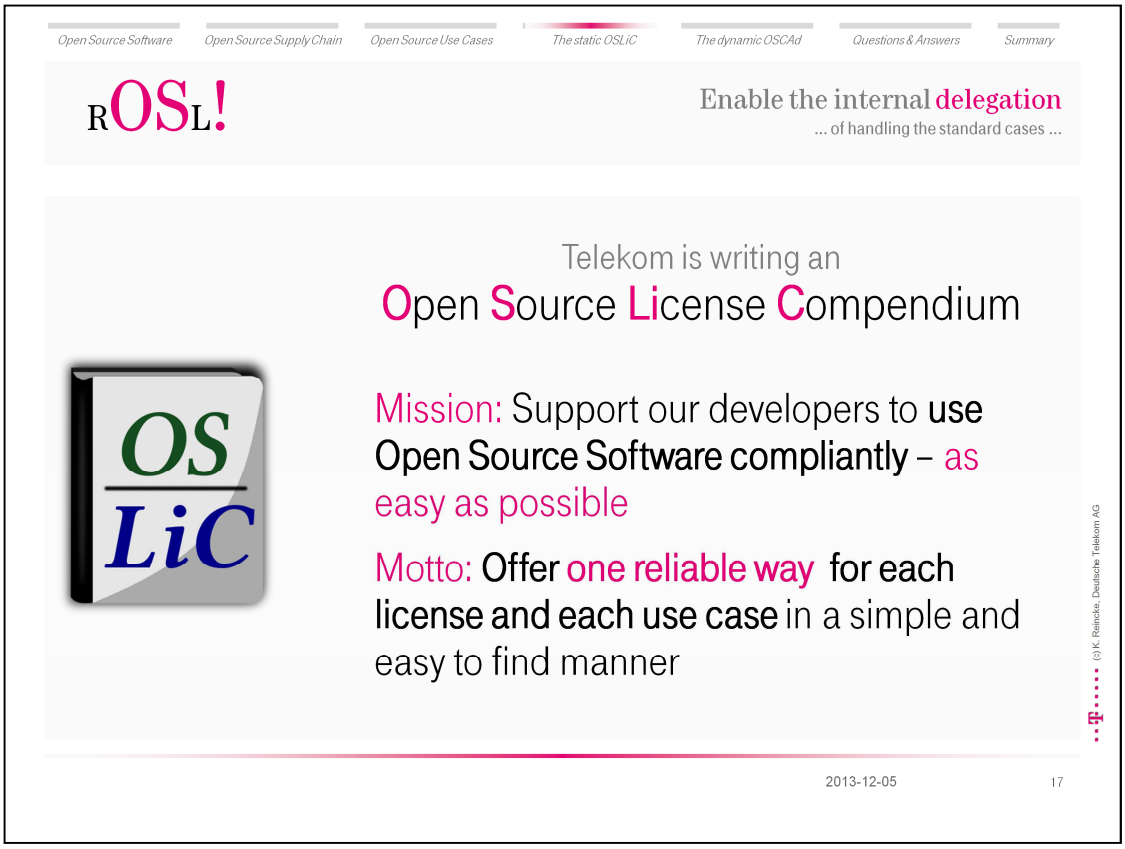
- Based on this taxonomy one has only to point to the open source license and to answer 5 questions and for being able to determine the use case specific set of license fulfilling tasks – if one has such a list of lists of license fulfilling tasks.



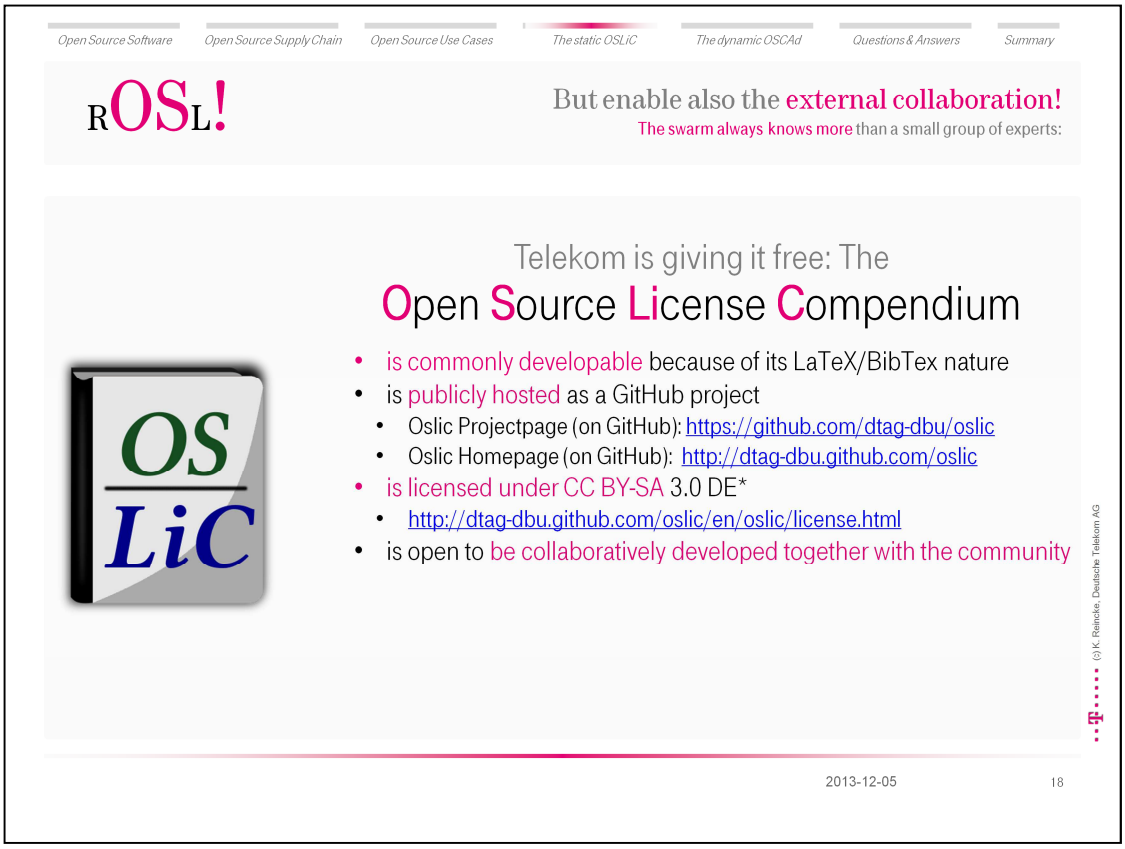
- The 'bad message' is, that this system of classification might be volatile. It depends of the regarded licenses and the features which they use as trigger for the activities they require. If – for example – at some point the OSI approves an open source license which dictates that in case of a distribution of more than 5 MBs the distribution must be executed on USB memory sticks, then we have a new dimensions of distinctive features which has to be incorporated into that system.
- The 'good message' that – as far as we can see – all existing open source licenses can be managed by this finder system.



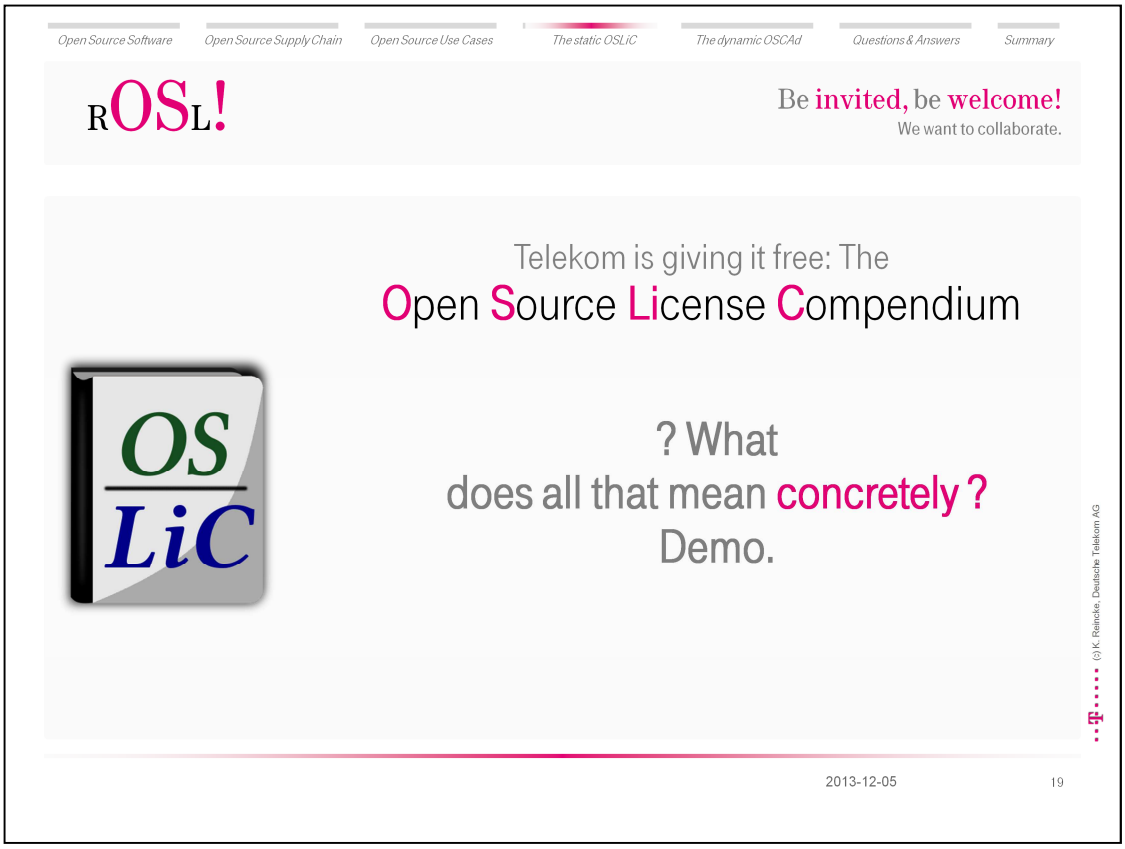
- So, the other good message is, that the OSLiC, the (Telekom) Open Source License Compendium, already offers license fulfilling to-do lists for the mainly used open source licenses and all its' open source use cases.



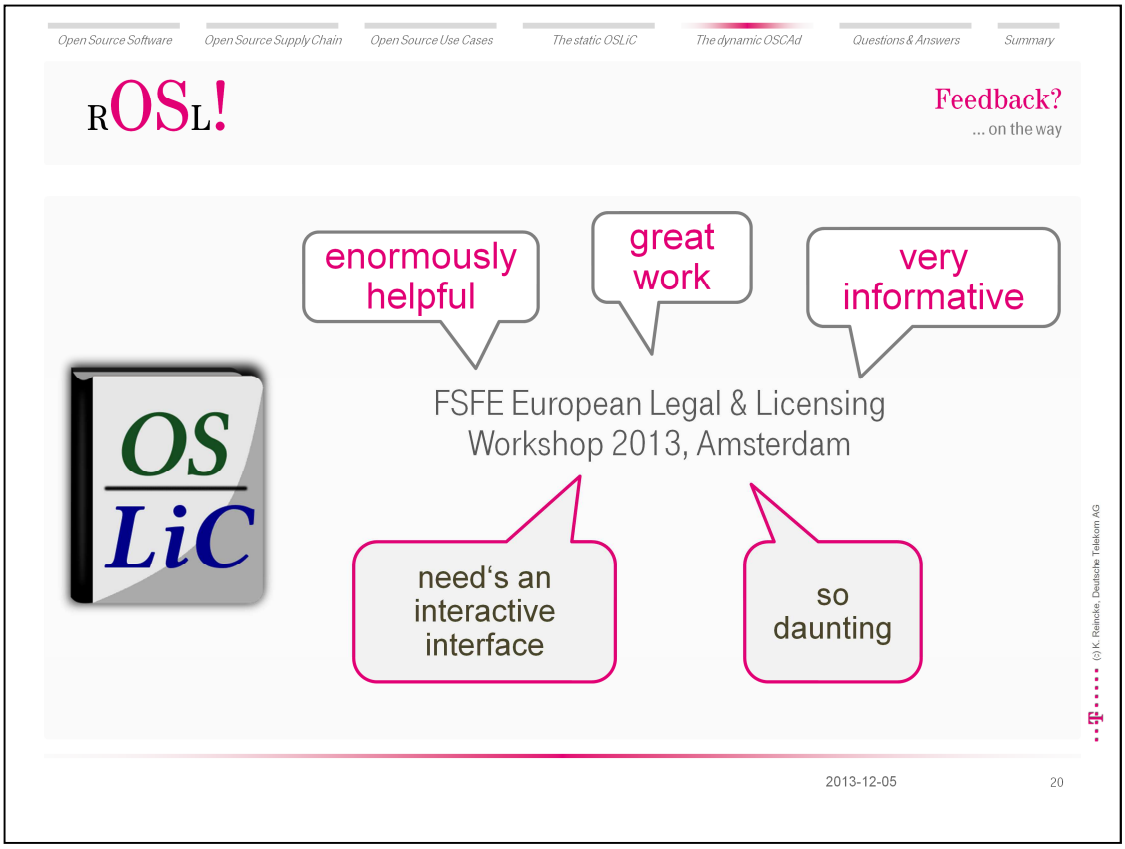
- The Open Source License Compendium is created on the base of a mission statement and a motto:
  - the mission: Support the developers to use Open Source Software compliantly – as easy as possible
  - the motto: Offer only one reliable way for each license and each use case in a simple and easy to find manner



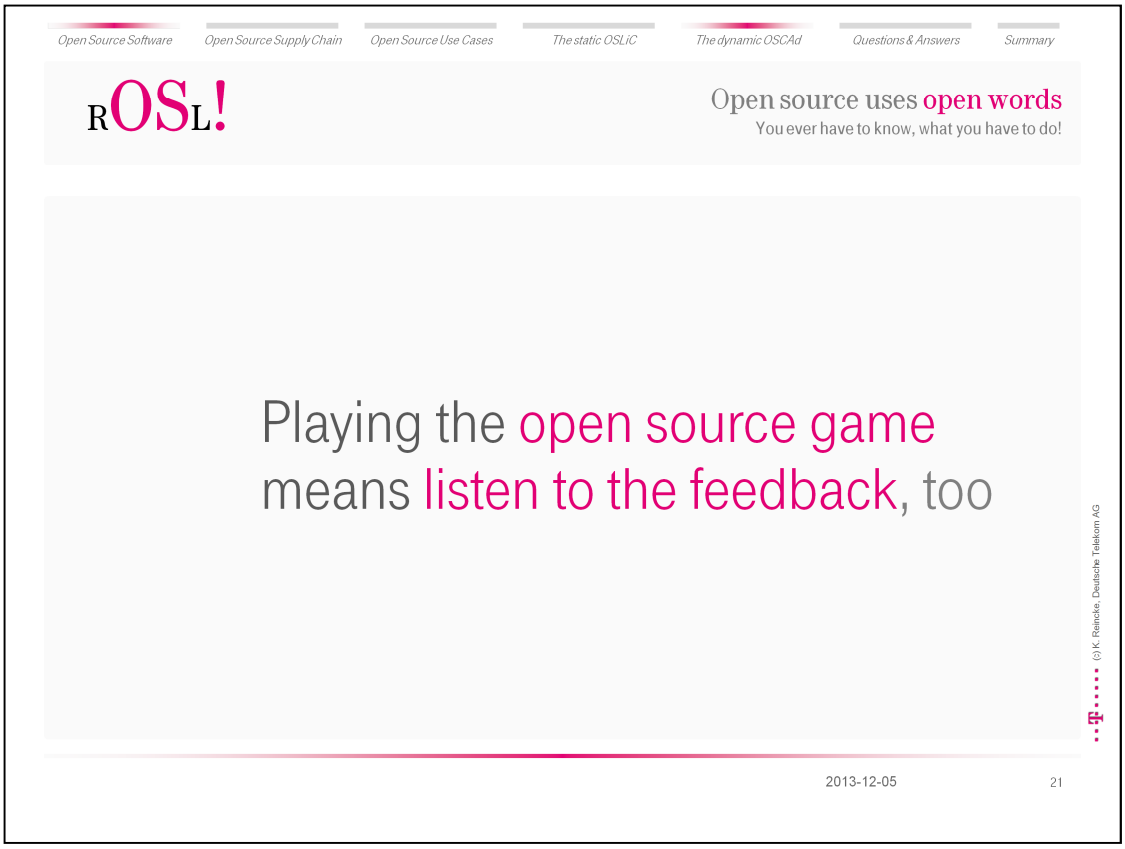
- The OSLiC itself is a special kin of open source ‘software: it is published under the Creative Commons License Share Alike – a license which might be viewed as GPL for documents – including all sources to modify and recreate the pdf document.


Open Source Software
Open Source Supply Chain
Open Source Use Cases
The static OSLiC
The dynamic OSCAd
Questions & Answers
Summary
rOSL!
Be invited, be welcome!
We want to collaborate.
Telekom is giving it free: The
Open Source License Compendium
OS
LiC
? What
does all that mean concretely ?
Demo.
(c) K. Reincke, Deutsche Telekom AG
2013-12-05
19

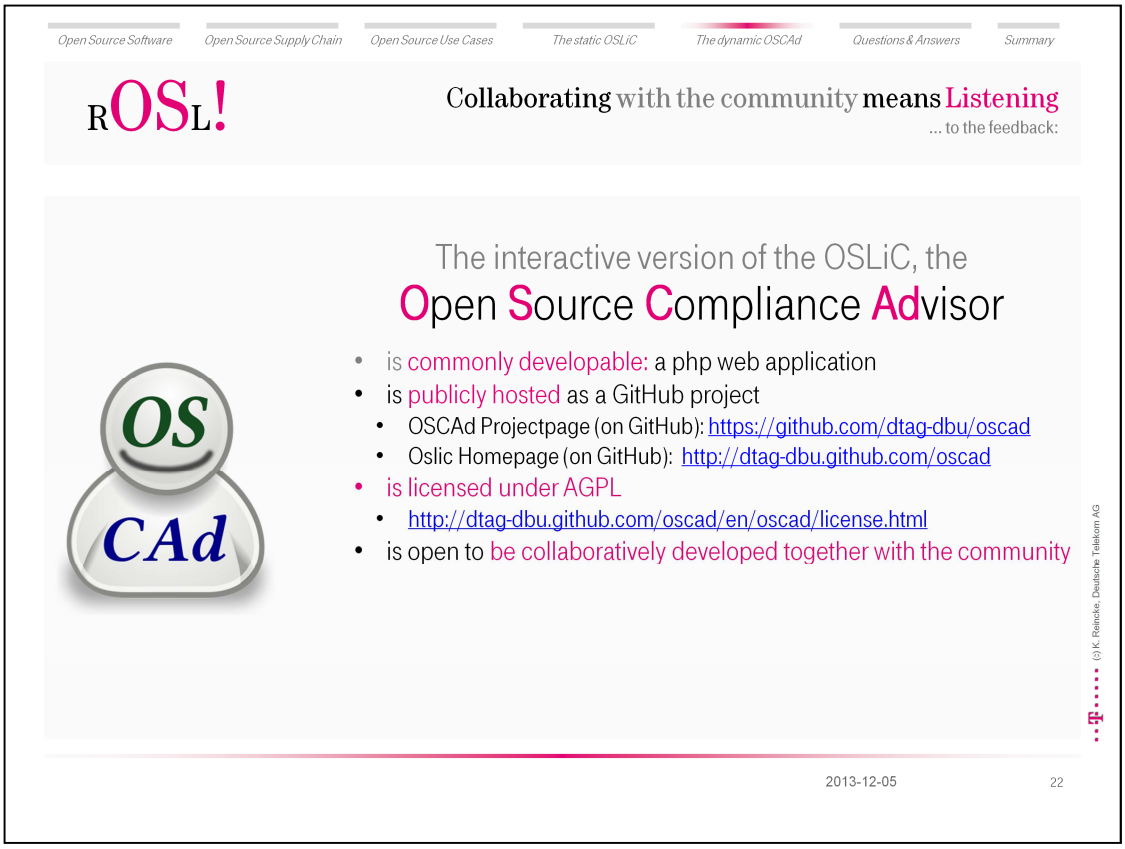
Demo OSLiC.



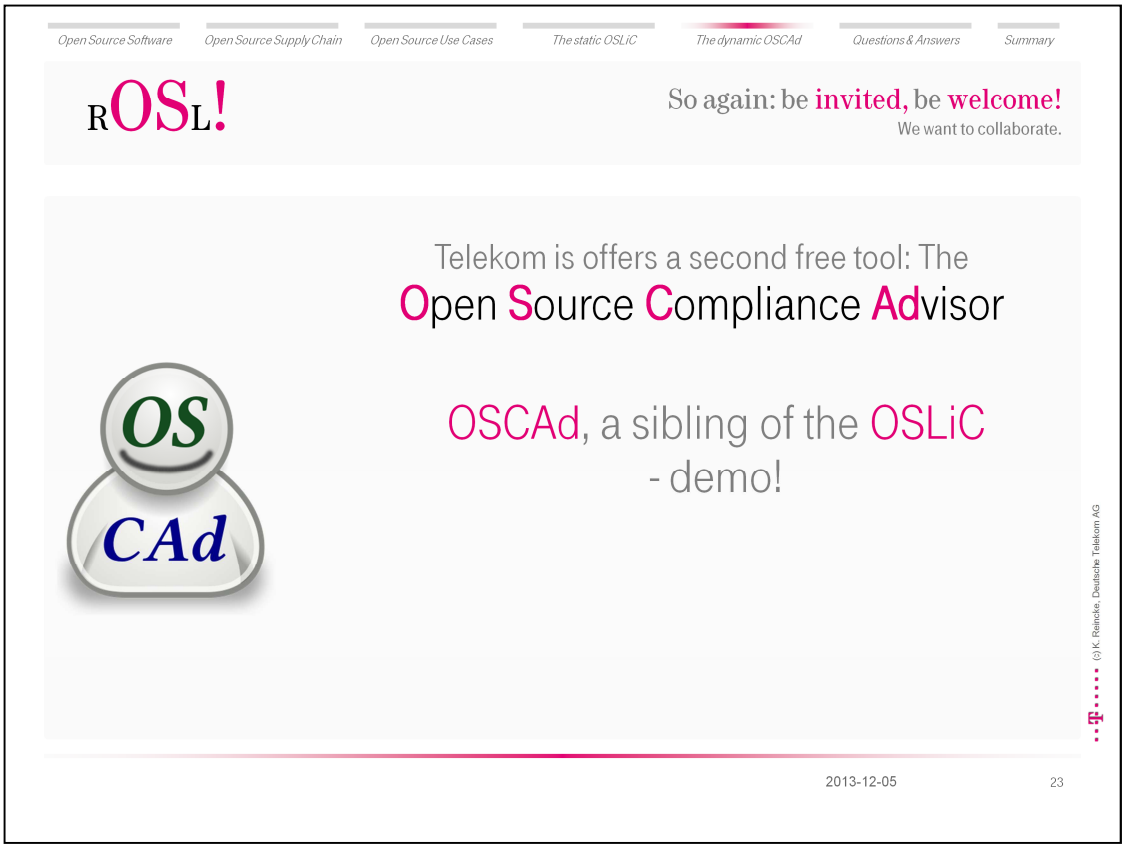
- In April, 2013, Deutsche Telekom presented her work on the FSFE Legal and Licensing workshop in Amsterdam. The feedback was a split vote:
  - Big encouragement, many thanks – especially of larger companies, and great admiration for having structured this topic successfully.
  - But also the message,
    - that the OSLiC is daunting,
    - that it would miss the intended auditorium, t
    - hat programmers and project managers never would read a 'book' of about 300 pages mainly containing lists and tables,
    - and that it should exist an interactive version.



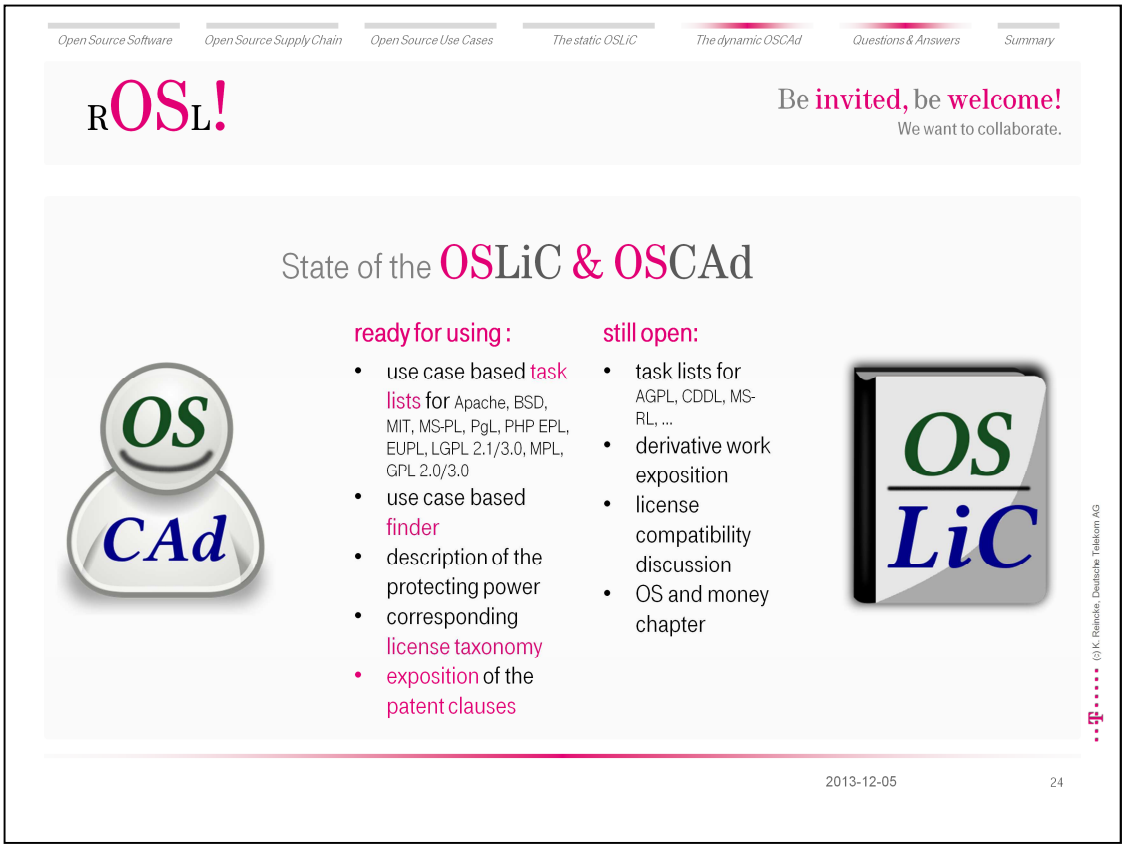
- First, we were a little disappointed and frustrated. But really soon, already on the way home from Amsterdam we had to concede: The feedback is right. With open eyes, we would have been able to derive this result ourselves.



- So DTAG to republish this compendium as interactive web application, licensed under the AGPL, and also hosted on github, but with its own name: „the Open Source Compliance Advisor“..



Demo of OSCAd.

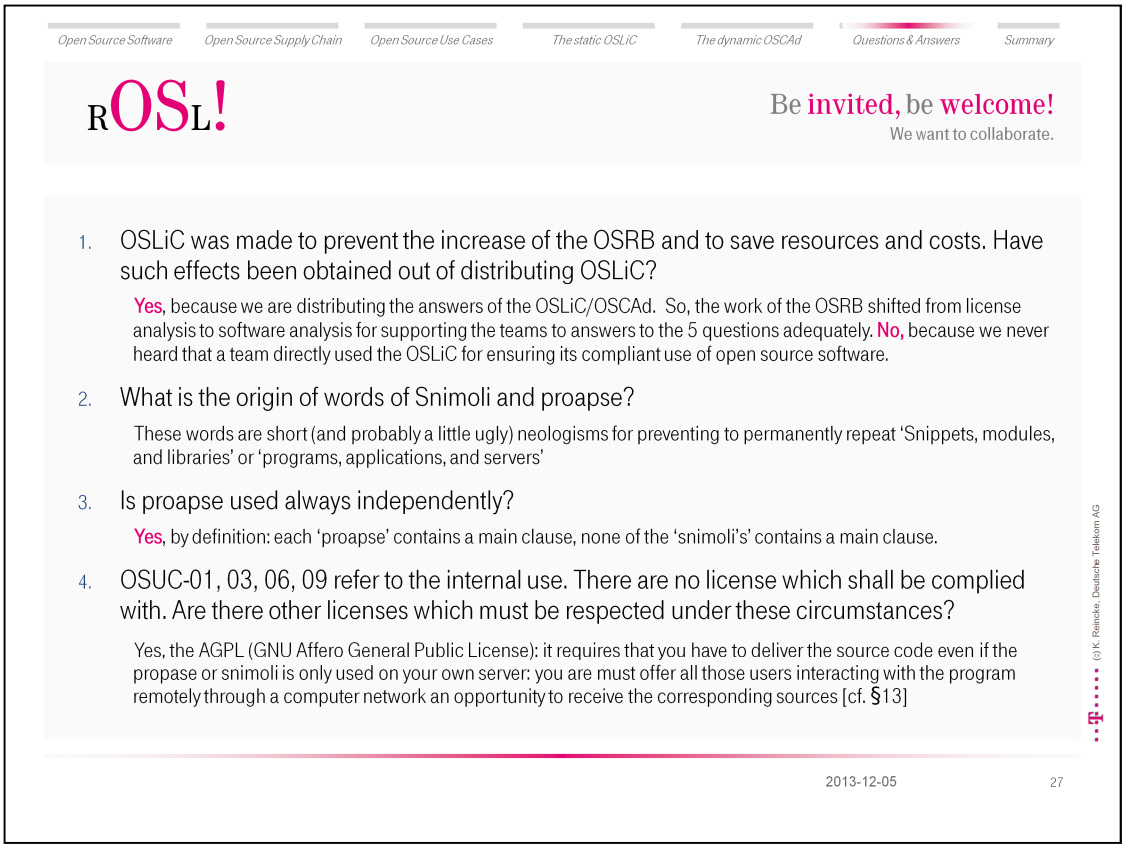


The current state of the OSLiC and OSCAd.

The future of the OSLiC and the OSCAd is the already cooperating community:

- The Korean translation of the OSLiC can also be regarded as a prominent example,
- as the OSLiC maintenance, executed by Deutsche Telekom AG, a well known European telecommunication company or
- as the just started re-implementation of the OSCAd by the company Amadeus, the leading provider of IT solutions to your tourism and travel industry: This coming version can be customized in a more easy way. And it will be able to handle translations in a more appropriate way. Amadeus is doing a great job and wants to publish first results on the European FOSDEM in Bruxelles. As a result of this cooperation, Deutsche Telekom will focus on the maintenance of the OSLiC and the integration of new licenses. And Amadeus takes over the job to maintain the OSCAd and to integrate the results of the OSLiC maintenance.

- So, for summarizing the result:
- We, DTAG, need a supporting tool for acting according to the Open Source licenses - but without investing to much manpower.
- First, we set up the internal Telekom Open Source Review Board
- Second, we are going to reduce the work of our OSRB by writing a simple to use compendium covering to-do-lists for the standard case.
- Third – as a giving back to the community (and as process of a public review) – we‘ve published this compendium in the spirit of open source
- Fourth, we listen to the feedback and published also a free interactive version of the OSLiC, the Open Source License Advisor OSCAD



- Some additional remarks:
  - >1) But we do not know whether they simply do not know the OSLiC, whether they have not told us that they used it, or whether they do not like to use the OSLiC. We assume the latter. The OSCAd has recently been communicated into the company. First reactions were promi sing.
  - >4) For being honest: the existence of the AGPL has influenced the structure of the OSLiC f inder even if we are still working on ist integration into the OSLiC and the OSCAd.

rOSL!

Be invited, be welcome!
We want to collaborate.

1. Are all kinds of use of software are covered by OSUC-01 und OSCU-10B? Is the taxonomy complete and stable?

   Unfortunately No! This system of classification might be volatile. It depends of the set of analyzed licenses and the features which these licenses are using as trigger for the activities they require. Luckily, from a specific point of view it is nevertheless stable: It covers the main licenses already integrated into the OSLiC / OSCAd. And we have never heard about other triggering criteria.

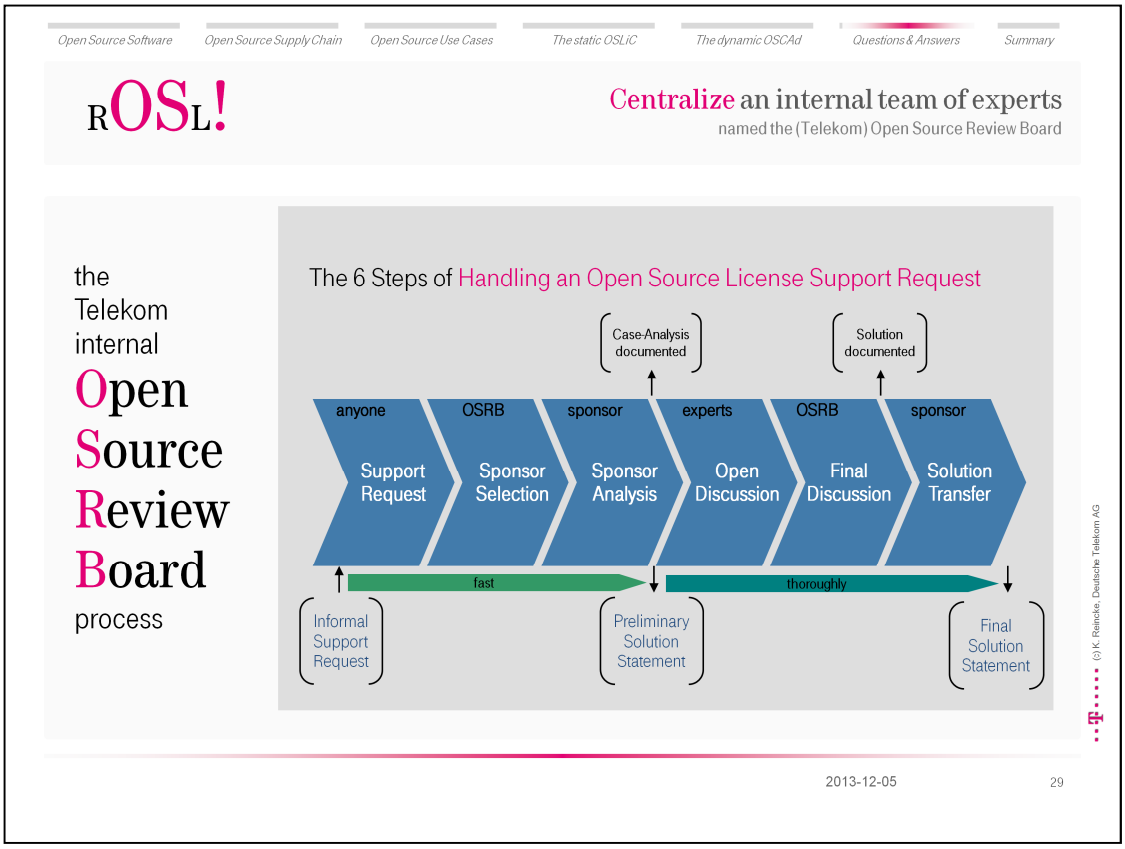
2. Should there exist a more efficient way to use OSLiC, such that OSLiC is useful to search each use cases?

   Yes: There was undeniably a need to have an interactive version (the OSCAd): searching the use cases manually in the OSLiC is a little annoying. Yes: The OSLiC should be improved if we meet other open source licenses using other triggers. Currently No: for the moment OSLiC and OSCAd are adequate tools with respect to the pareto principle.

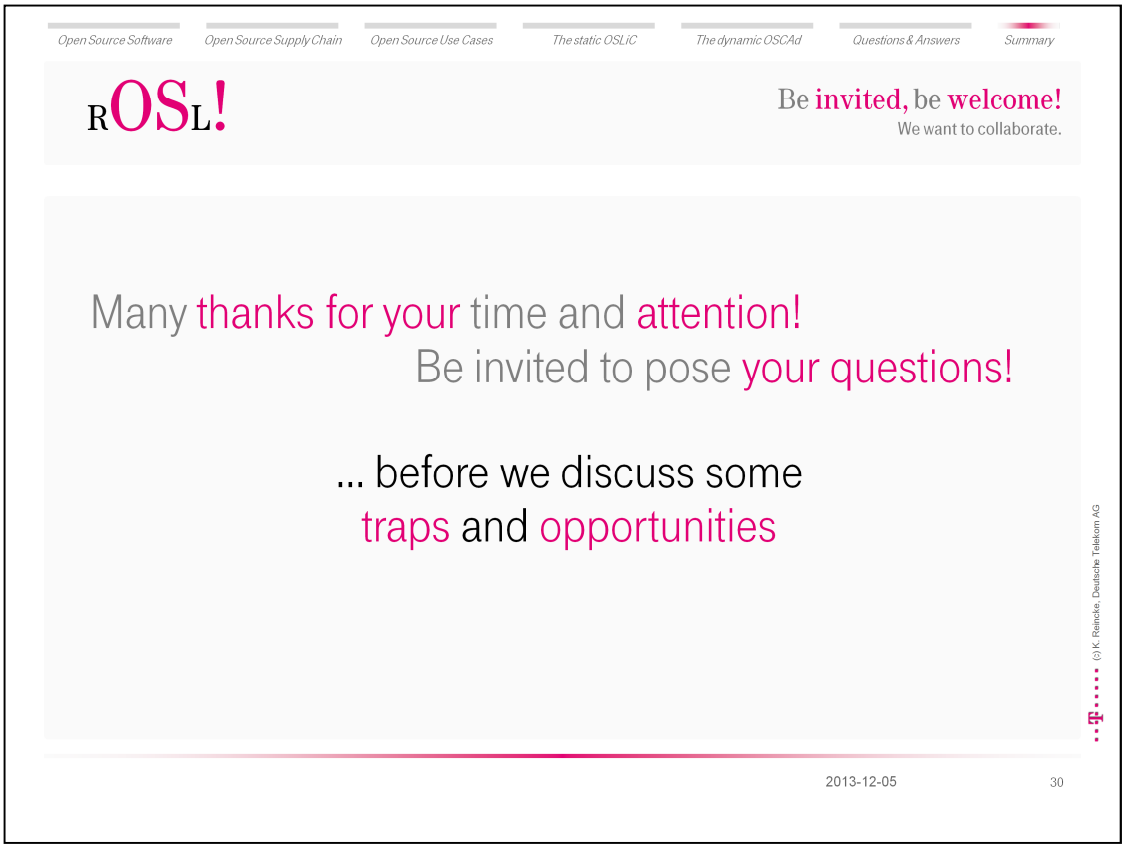
1. What is OSCAD, and how to use it?

   OSCAd, is the online version of the OSLiC. So, we in DTAG are now supporting our teams on 3 levels:
   - First they can inform themselves about their obligations without becoming a license expert (use OSCAd)
   - Second they can verify their obligations by checking the reasons (use OSLiC)
   - Third they can ask the OSRB for supporting their work.

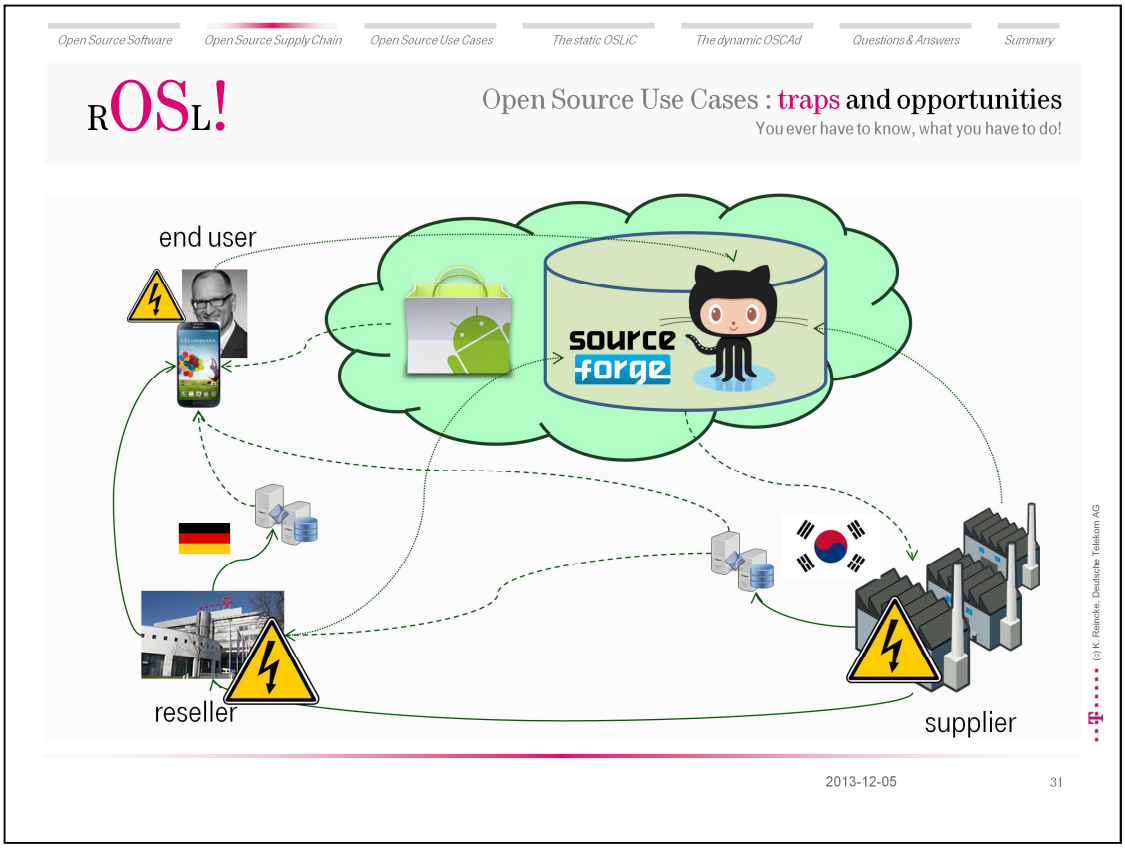
3. How is the Telekom Open Source Review Board staffed and which process does it execute ?



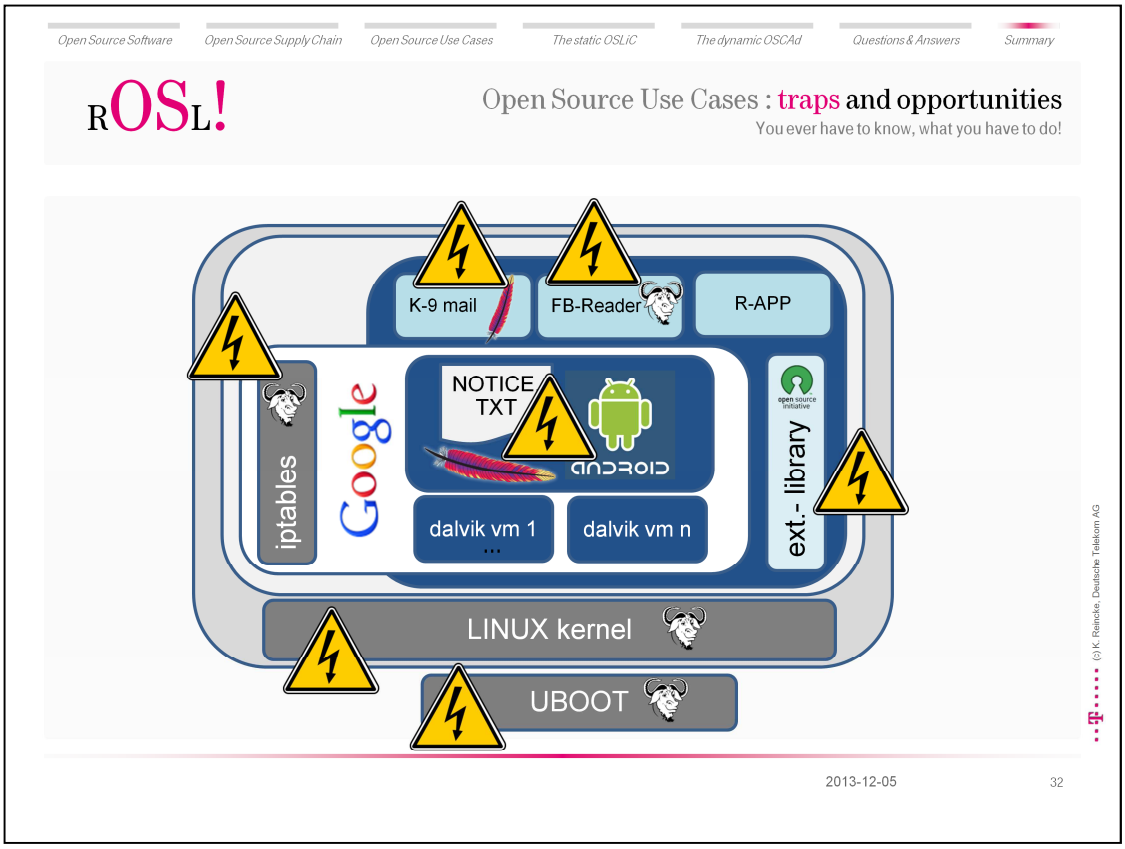
- Firstly we established indeed a team of experts, internally known as the Telekom Open Source Review Board. This team was a self organizing team. Until now it is only a small team containing
  - a lawyer as legal expert,
  - a software developer for mobile environment, which is also a dedicated open source software expert
  - a software architect being also a java programming expert and open source apologist
  - a project manager and open source software expert on many different aspects.
  - two ipr and patent experts
  - two associated members
- This team acts according to this process: any one in the company can ask any one on the board for a support. The board determines a sponsor for the request which firstly and very quickly analyzes and describes the case and gives a first preliminary solution statement. Then this statement is discussed and finally closed by the OSRB as whole.



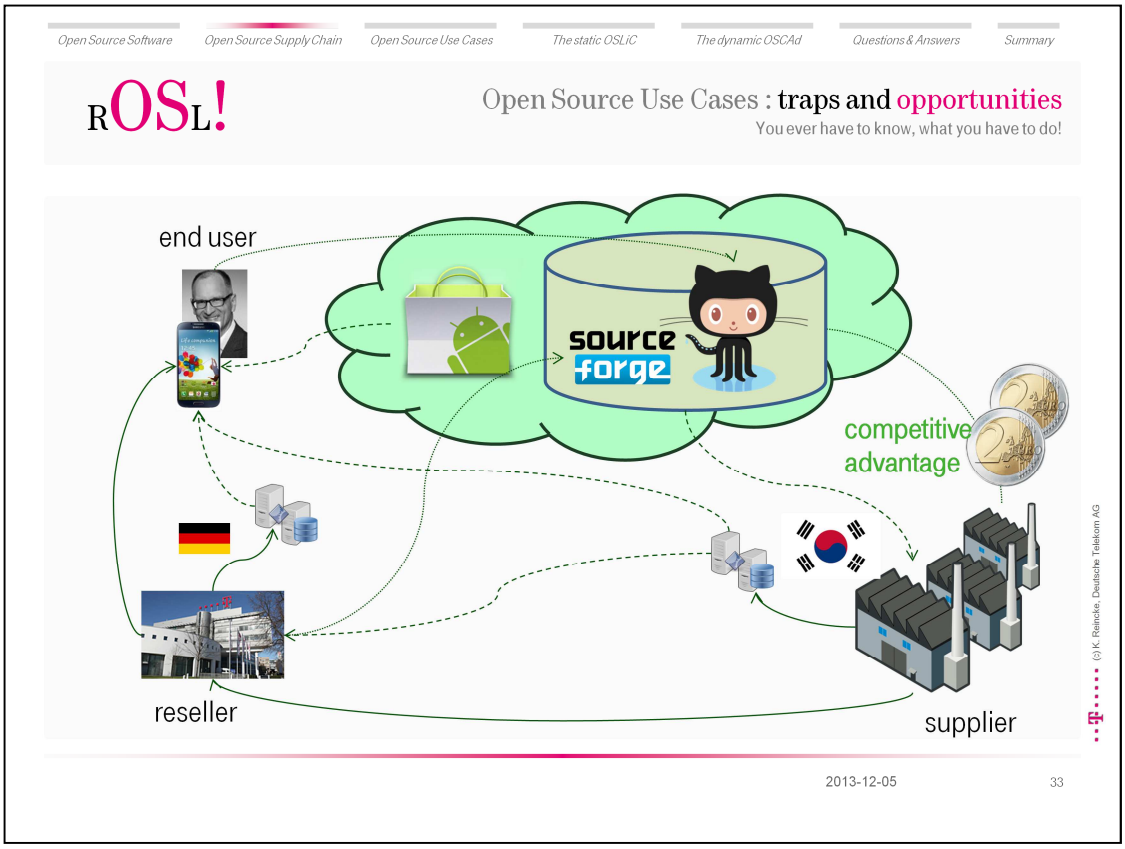
- So, I am Karsten Reincke – many thanks for your attention. Let us answer your questions – before we jump to some aspects of the praxis I additionally prepared for being to considered too.



- An interesting question concerning the supply chain is the question who has to focus on acting in accordance to the open source licenses:
  - Firstly, it is the supplier itself.
  - Secondly – but often forgotten – also the reseller has to fulfill the license requirements. Fulfilling the open source licenses can not be delegated. All licenses seems to be very clear: the one who uses / distributes the software as to fulfill the conditions.
    - Of course, he might reused prepared tasks – like lists of announcements or elaborated copyright dialogs.
    - But, there are some tasks for which he can't reused items prepared by his supplier. One of the famous points are the promises to deliver the source code as they are allowed by the GPL. The GPL only allow non commercial organizations to redistribute such promises.
  - Thirdly, also the end user has also to fulfill the requirements of open source licenses. He is using the software too. But because if you are using the software only for yourself and if you do not distribute it to third parties, concretely he has nothing to do. But if he distributes his mobile phone – especially if he resels it – then he has to fulfill the conditions. In which way and in which extent: that is a good question for all the lawyers.



- Here are listed some challenges concerning the 'granularity' problem – exemplified by the android system:
  - Even a smartphone needs a bootloader. And this bootloader can also be free/open software – like uboot!
  - The linux kernel is often changed. Bit even if not, the sources of all GPL software must be made receivable by each distributor!
  - The Android system contains also other GPL licensed applications running in the user space.
  - Google 'offers' Android as a unit. So one can assume, that Google deals all open source software correctly. Apache software requires each distributor to present the content of the notice file. So one can assume that Google fulfills all its obligations concerning the announcements by writing an elaborated notice.text file. Nevertheless, it is not sufficient to re-distribute this file. For all software with strong or weak copy left the distributor himself has also to offer the code.
  - Resellers may install additional components above of the Android basic system:
    - If it is a piece of Apache licensed software (like K9), the resellers themselves have to annouce the use and to distribute the K9 notice file. It is not automatically incorporated into the Android notice text file.
    - If it is a piece of a weak or strong copy left licensed software, the resellers must also offer to hand over the source code.
  - Resellers may also (develop and) install the own reseller apps. In this case, the reseller has to ennure that this development lists all other external open source libraries



- Enabling the reseller to fulfill the license requirements can be a great advantage for the supplier in the world of competitors.
  - If the reseller himself has to do all the investigation work for finding the relevant open source entities and all the collecting work for getting all the information which must be distributed, then the resellers has to spend a lot of resources an money for acting in accordance to the open source software licenses.
  - Mostly, for the supplier it is much easier to get, to collect, and to distribute the information and the source codes together with its product.
  - A reseller should pay for such a preparing work, but not as much as for the work he has to do himself.

A typic win-win situation

- So, I am Karsten Reincke – and again: many thanks for your attention.