

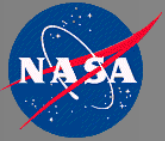
Open Source Breakout

Chris Mattmann, ESDS Software Reuse WG Chair

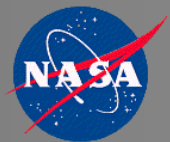
9th Earth Science Data Systems Working Group Meeting

New Orleans, LA

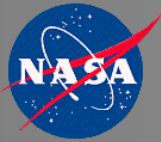
October 20–22, 2010



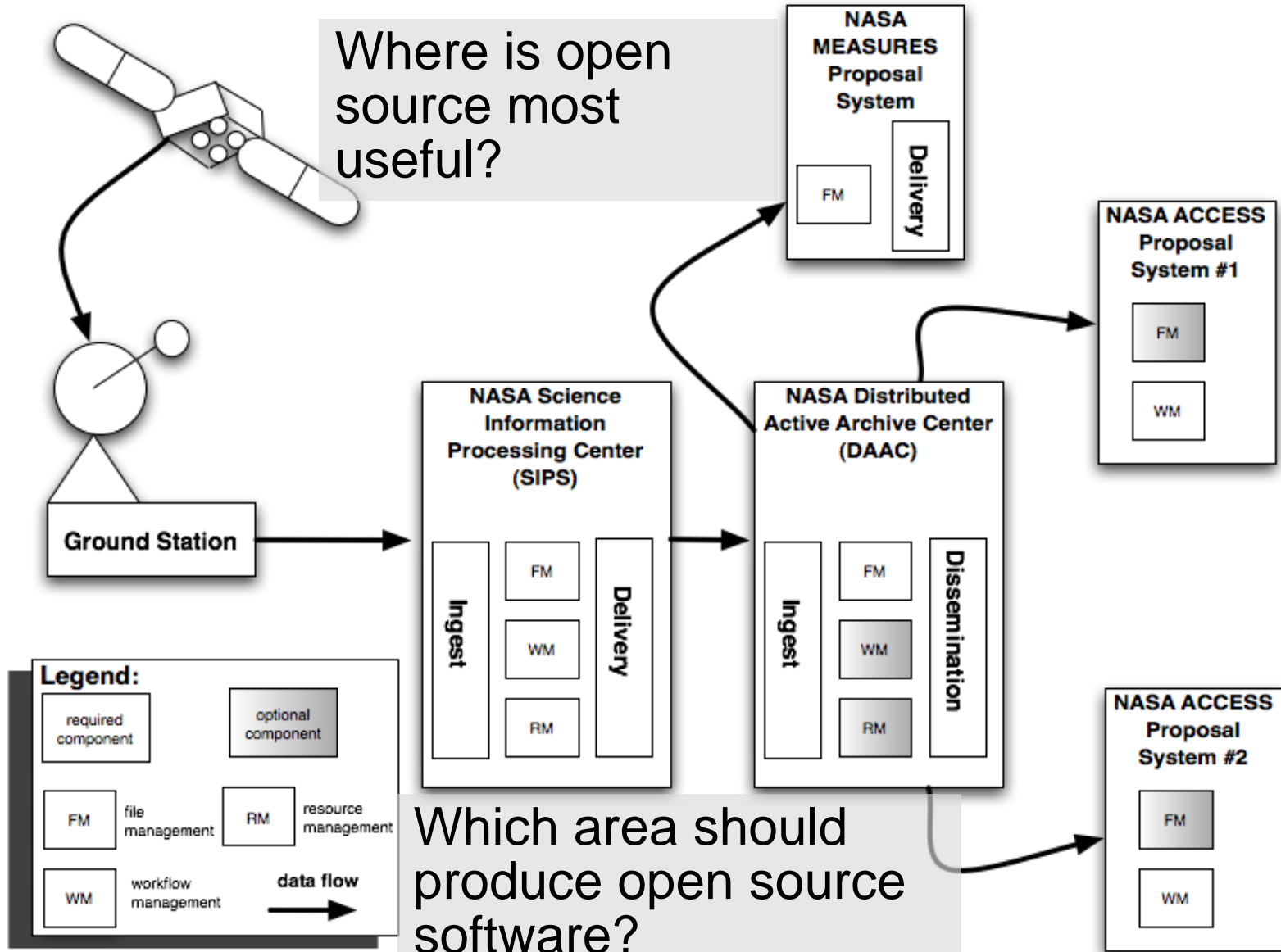
-
- **NASA ESDSWG 9th Meeting: Open Source Breakout**
 - *Successful Architectures*
 - *Licensing Strategies*
 - *Infusion of FOSS into NASA Earth Science Context*
 - *How to identify the “right” FOSS community*
 - *Contributing back to FOSS*
 - *Implementation Strategies*
 - *But wait...there's MORE!*
 - **Guest Speaker:**
 - **Dan Crichton, Program Manager, Earth and Planetary Data Systems, NASA Jet Propulsion Laboratory**
 - **Be ready to discuss and participate in the conversation!**

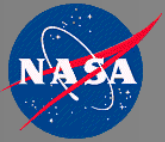


-
- **Why Open Source?**
 - - *Cost Savings*
 - - *Technology Innovation*
 - - *Stability*
 - - *Redistribution and Reusability*
 - **Why NASA Earth Science Data Systems?**
 - **Significantly rich context for software development**
 - **Distinct areas with lots of innovation, but different funding, schedules, concerns make (re-)use of open source hard**

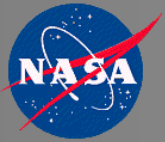


The NASA ESDS Context





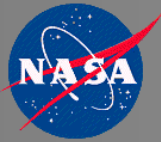
- Licensing
 - GPL(v2, v3?), LGPL(v?), BSD, MIT, ASLv2
 - Your own custom license approved by OSS
 - NASA OSS license?
 - Caltech license?
 - Copy-left versus Copy-right
- Redistribution
 - Can you take open source product X and use it in your commercially interested software Y?
 - If so, do you have to pay for it?
 - Should others pay for your open source product if they use it in their commercial application?
- Open Source “Help Desk” Syndrome versus Community
 - Are you trying to simply make your open source software (releases) available for distribution (aka *help desk*)?
 - Are you trying to get others to “buy in” to your open source software?



-
- Intellectual Property
 - Who owns it?
 - How does the Open Source Software affect your IP?
 - Open Source Ecosystems
 - Where can you find the “killer app” you need?
 - Which communities are conducive for longevity?
 - How relevant are “generic” open source software communities to NASA Earth Science Data Systems?
 - Contributing
 - *Are you even allowed* to contribute to a OSS community?
 - Can you do it on “company” time?
 - What’s required?
 - What’s the governance?
 - Responsiveness
 - How response is the OSS community to your projects’ needs?

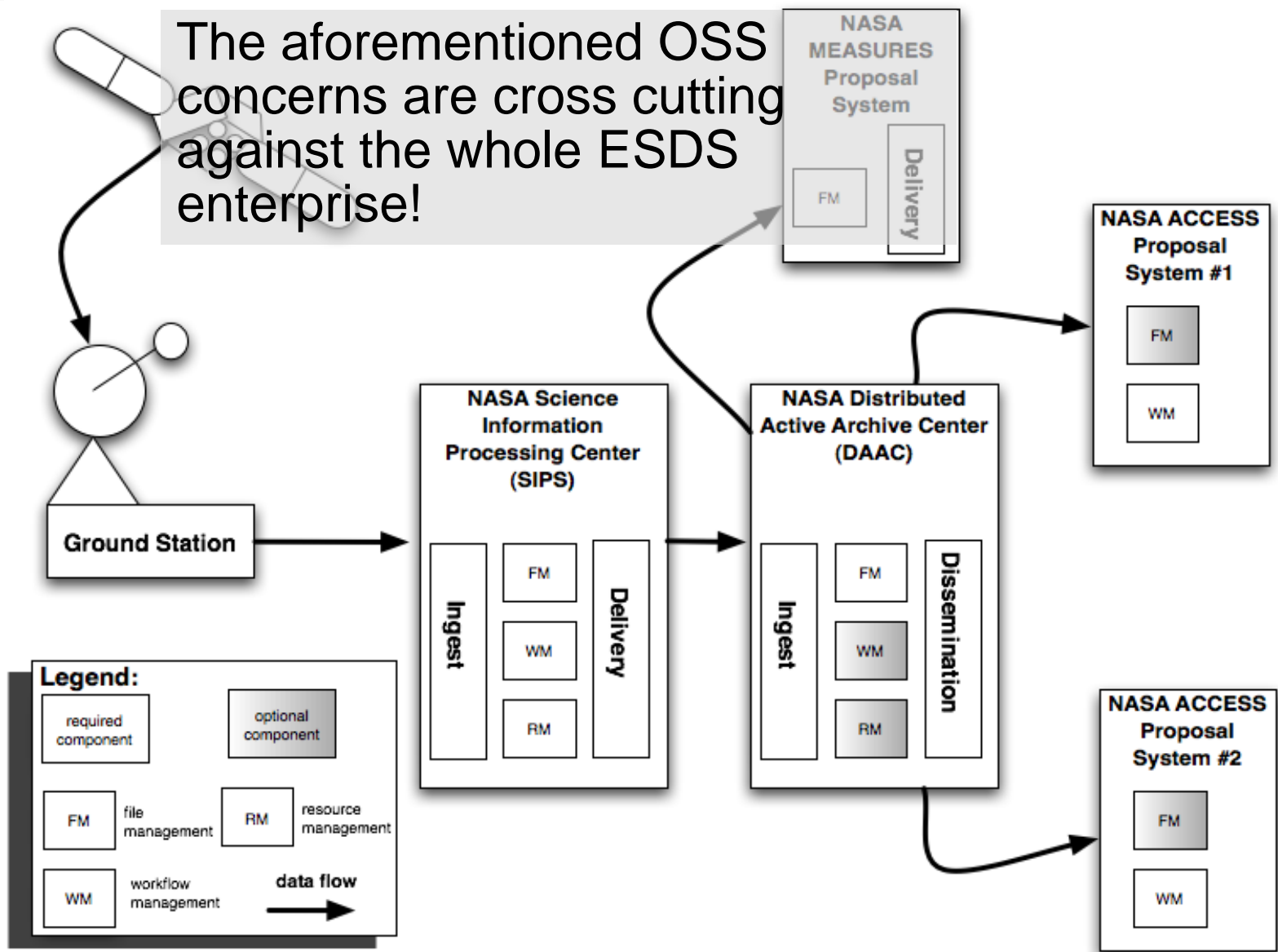


-
- **Help/Guidance**
 - Is the OSS community/project alive?
 - How can you tell whether the project is alive?
 - How can you “follow the rainbow” to the OSS pot of gold?
 - **Interaction**
 - What are the best practices for interacting with OSS communities?
 - **Implementation strategies**
 - Insulation: how do you insulate your project from OSS change?
 - Configuration Management: what are the important CM issues when using Open Source Software in NASA ESDS?
 - **Architectural strategies**
 - How to design your system to take advantage of OSS?
 - **Legal strategies**
 - How to avoid getting sued by Oracle some huge tech company?



The NASA ESDS Context

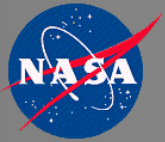
The aforementioned OSS concerns are cross cutting against the whole ESDS enterprise!



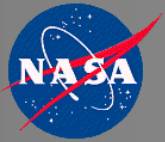


-
- *Please come to the Open Source Breakout!*

 - *Please listen and attend Dan Crichton's talk!*



-
- *Relates to: redistribution, intellectual property, contributing, legal strategies*
 - There are tons of OSS approved licenses
 - What's the difference between them?
 - The difference mostly has to do with
 - Commercialization
 - Redistribution
 - Attribution
 - Let's take a few examples



- Apache License V2

- Allows (unrestricted):
 - Redistribution (or not)
 - Commercialization
 - Must keep ASL headers and NOTICE file bearing ASL attribution
 - Contributors to ASF sign Apache CLAs or CCLAs

- BSD License

- Allows
 - Redistribution (or not)
 - Commercialization
 - Must include header in code or in NOTICE
 - 0 contrib. restrictions

Apache License
Version 2.0, January 2004
<http://www.apache.org/licenses/>

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the license.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

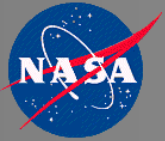
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of the authors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL JCRAFT, INC. OR ANY CONTRIBUTORS TO THIS SOFTWARE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING



- GPL(v2, v3), LGPL
 - Allows (restricted):
 - Redistribution
 - Based on implied commerciality and copyleft
 - Commercialization
 - May need to pay license fees
 - Standard attribution requirements (NOTICE or in source code headers) based on copyleft
 - “Copy left” Syndrome
 - Software must be redistributed under the *same* original terms of the any upstream author.
 - Author is not free to decide redistribution terms, or even source code inclusion terms (except under fair use which supersedes)

-
- “War Story”
 - Amazon EC2, S3
 - Johnson and Johnson Pharm. R&D
 - At a recent conference I met the director of R&D for J&J. He presented a story wherein which J&J needed *large* bursting processing and limited data storage for some drug tests they were conducted. They decided to use Amazon EC2. After reviewing Amazon’s licensing policy for EC2 J&J’s lawyers determined that Amazon claimed IP for *any data* or *computational* results produced on its cloud. Since the need for Amazon’s processing and cloud was limited to a few trials, and since the costs were so outrageous to stand up its own cluster for these experiments, J&J decided to forge ahead with Amazon with the understanding that its lawyers would “duke it out.” should the need arise with Amazon’s lawyers based on the FOU restrictions and IP claims induced by Amazon EC2.

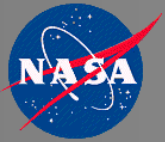


Why licenses are important

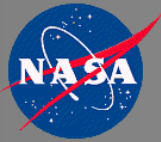
- “War Story 2”
 - Oracle versus Google
- Is there really free Java with your free lunch?
 - Although there is no definitive answer besides papers filed in court, Oracle’s claims in its lawsuit are based on perceived patents for the Java Virtual Machine and its associated IP. Sun originally filed patents on the Java Virtual Machine and its translation of programming language code into runtime executables. In order for a JVM to be a “certified” (read: trademark) JVM, the JVM must pass a “Test Compatibility Kit” (TCK), which Oracle/Sun license at a cost to JVM vendors. By purchasing the TCK, a JVM builder is given IP knowledge of the JVM patents. If a company builds a JVM but does *not* purchase the TCK, Oracle/Sun loses licensing dollars.



-
- Awareness
 - Know your distribution requirements
 - Know your contributor expectations
 - Know your field of use expectations
 - Know your commercialization expectations
 - Leverage licenses that give the most leeway in all important categories above
 - Allows the decision on some of the above aspects to be delayed without fear of penalty or prejudice
 - Friendly licenses: ASLv2, BSD, MIT
 - Know what you are signing *before* you sign it (or use it)



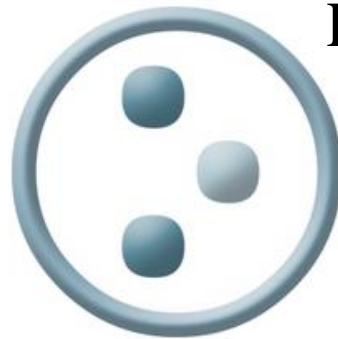
-
- So you've built some awesome piece of NASA ESDS Software
 - And you're wondering
 - What are my options for distributing it to
 - Other DAACs?
 - Other SIPS?
 - Other funded proposal efforts we're collaborating with (ACCESS, MEASURES)
 - Any other collaborators?
 - Question: what license are you going to choose?
 - *Hopefully one that supports redistribution under your own terms!*
 - How to redistribute the software?
 - Requires infrastructure
 - Who's going to set it up?



Issue
tracking



Source code
repository



Portals

Plone



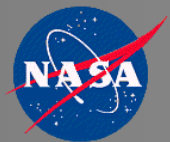
Repository browsing

Acceptance testing



Mailing lists





What options are there?

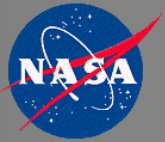
-
- **Sourceforge**
 - Provides redistribution infrastructure
 - No VOTE'ing requirements
 - No requirements on signing
 - **Apache**
 - Provides redistribution infrastructure
 - Releases must be VOTE'd on
 - Releases must be signed by releaser
 - **Github**
 - Provides redistribution infrastructure
 - Not set up for releases but for active development
 - Branching and forking of existing software is encouraged
 - **Google Code**
 - Provides redistribution infrastructure
 - No VOTE'ing requirements
 - No requirements on signing



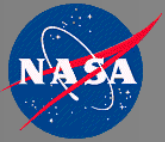
Community versus “Help Desk”

- How do you want to have your open source software project run in the open?
 - Do you expect folks to come to you and *only* file bugs?
 - Then you and your team are the only ones who can fix them?
 - Then you and your team are the only ones who can release updates?
 - → “Help Desk” open source project
 - Examples: Sourceforge.net, Google Code, etc.
 - Do you expect to grow a community of interest where volunteers actively engage in software development?
 - Are volunteers empowered to pick up a shovel and help dig the hole?
 - Can volunteers (including your own paid employees) file issues?
 - Do you want to give the community a stake/vote in the overall process?
 - → “Community Building” open source project
 - Examples: Apache Software Foundation, Eclipse Foundation, etc.





-
- Working on common software
 - Measured not in terms of what center contributor works for, but in terms of
 - Number of patches contributed (high quality)
 - Mailing list questions answered
 - # of releases made, or helped with
 - Tests written
 - Documentation added
 - Take the politics out of it and just work on “core” common code of mutual interest
 - Deciding on the right redistribution mechanism and license
 - Apache Software Foundation and ASLv2 provide openness and ability for center-local redistribution, commerciality and other decisions (for internal distributions and beyond)



- Heavily influenced by the underlying
 - License used
 - OSS redistribution infrastructure
 - What “it” is?
- Case Study: Apache Hadoop

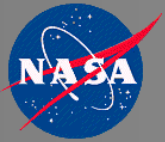


- Who owns it?
- Apache owns the Hadoop trademark
- What about?

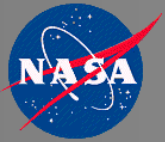
Cloudera's Distribution for Hadoop		
UI Framework		SDK
Workflow	Scheduling	Metadata
Data Integration	Languages, Compilers 	Fast read/write access
Coordination		

THE **YAHOO!** DISTRIBUTION OF





- Centers can similarly (if they so choose) have their own customizations/distributions of OSS
 - NASA Langley DAAC's FooBar powered by Apache Hadoop
 - NASA AIRS SIPS's Baz built on Apache Pivot powered by Linux
- NASA protects its marks through the New Technology Report process
 - Uncover marks
 - Uncover potential patents
 - License/etc., as appropriate
 - We see this less on the ground side than we do with the flight
 - Mostly NASA wide, but some center specifics (e.g., Caltech)
- **Takeaway:** the answer to “who owns it” is still “the Center” or “the project” who initiated the software development, with or without OSS. OSS may have its own restrictions/rules, so need to be wary of that (recall *copyleft* syndrome)
- **Advice:** Stay away from copyleft here.



- Where should you go to for your open source project?



Code

Results 1 - 10 of about 25,300. (2.62 seconds)

[adb/btree.h](#) - 5 identical

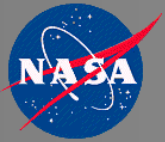
```
110: class btree {
```

```
114: public:
    btree(cbdispatch cb);
    virtual ~btree();
```

[cvs.fs.net/cvs/sfs1](#) - Unknown License - C++



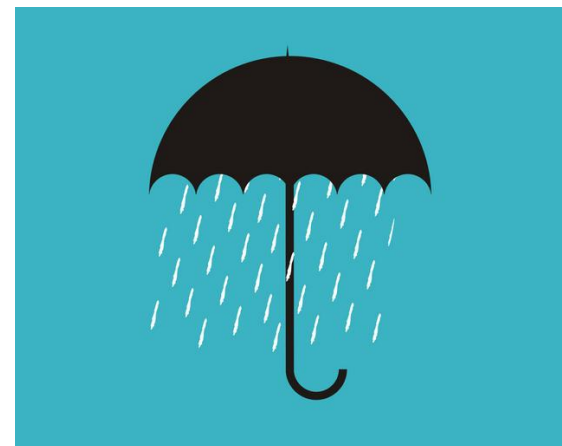
- Should NASA have its own?
- Should your project (SIPS, DAAC, proposal) have its own?

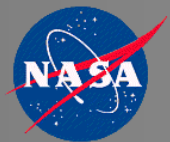


- FTR, there are tons of concerns here
 - Should the ecosystem impose license restrictions?
 - Only support license X or Y?
 - What are the redistribution policies?
 - What's the community?
 - What's the IP?
 - What's the infrastructure support?
 - Is it a foundation with rules, or just free for all?
- Elephant in the room: What is the exposure? How many people are going to community C and are going to see your software and perhaps want to use it, improve it, file bugs against it, file patches, etc.?
- Where/how does this matter to NASA ESDS?
 - Standing up our own OSS ecosystem may make sense for large, coarse-grained federations
 - A LOT more difficult to justify OTOH, for fine grained components, and module reuse

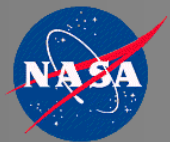


- Other valid concerns
 - What's the longevity?
 - Is this a brand new foundation?
 - A code dumping ground?
 - Does this community have the potential for “umbrella projects”?
- Case study: Umbrella OSS Project
 - Project code name “Umbrella” lives in OSS Foundation FooBar. Umbrella has a number of sub-modules to it, and a diverse community of code maintainers that are partitioned off to which modules that they can commit/contribute to. When Umbrella wants to make a release of sub-module X, it must gain consensus of the entire project and its contributor base, even though many of the existing project contributors only have knowledge of their particular sub-module of interest.

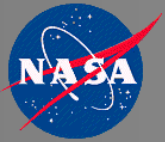




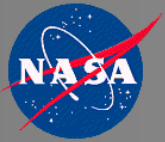
-
- OSS communities are built around “contributions”
 - But what does it mean to contribute?
 - Mailing list help/discussions
 - *Yes those are contributions*
 - Reporting/filing bugs and new feature improvements
 - *Yes those are contributions*
 - Writing documentation and user guides
 - *Yes those are contributions*
 - Volunteering to make releases
 - *Yes those are contributions*
 - Positively contributing towards the evolution of the project with your thoughts and ideas
 - *Yes those are contributions*
 - **NOTE: What did I leave out?**



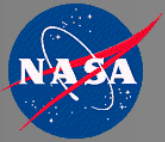
-
- Yes yes, code is important (but not the only contribution)
 - How should you go about your code contributing process to open source?
 - *Relates to: “Help Desk” versus Community*
 - Do you want the members of your project to be the only folks who can actual touch the source code of your OSS project?
 - Only bug reports?
 - How “open” are you?
 - How “open” do you want to be?
 - Do you want to accept code contributions from the community?
 - Patches?
 - Allow community members to become “committers” (assist in the development of the code?)
 - RTC versus CTR



- As soon as you start accepting OSS code contributions you have to decide between the following options
 - “Review then Commit”
 - Contributions come in the form of patches, which are reviewed by the project committers, adapted as necessary, feedback provided, iterated upon and then eventually committed in the final form
 - Issue tracking systems used heavily in this process
 - Committers have to be willing and contributors have to be willing
 - Egos thrown out the door b/c RTC can take a LONG time
 - May take longer to get new committers to the project
 - “Commit then Review”
 - Just commit the code, and use the CM system to track its evolution
 - Can introduce “feature creep”
 - Gives the contributor a quick “pat on the back”
 - Committers must be willing and contributors must be willing to address any feature creep or bugs
 - Promotes new committers fairly quickly



- “War Story 3”
 - Majority Rules
- The tyranny of the majority?
 - Company X decides to put a project Foo into open source. They go through the OSS Y Foundation which encourages community building versus “Help Desk” syndrome. Project Foo decides to employ RTC practices to accept code contributions from the community, and chooses to weight code contributions higher than any other form of contribution. Since Project Foo is managed by a majority of people within Company X, and since those people have Company X’s best interests in mind (versus the OSS project Foo or its community), Project Foo only elects “friends” who are subordinate to their ideals or members of Company X, as new code “committers”. When outside contributors provide patches, they sit in an issue tracking system for years, and are eventually turned into new project Foo features in Company X’s “distribution of project Foo”, a repackaging of open source Foo for Company X. After a few years working in this model, the non Company X members of project Foo stop participating on the mailing lists, and leave.



-
- Measure of a “healthy” community
 - Does your patch sit?
 - Do your mailing list questions go unanswered?
 - Is the project electing new “committers” (if it’s in community mode)?

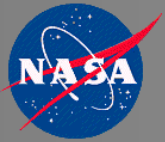
 - How should your NASA center/group decide whether or not an OSS project you want to use is responsive?

 - How should your NASA center/group decide how responsive *it will be*?

 - Heavily influenced by:
 - Community mode: “Help desk” versus “Community”

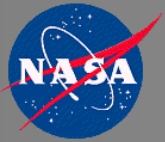


-
- Using Open Source Software is *different* than producing software intended for distribution in open source
 - Either way: what's your strategy for getting help?
 - It used to be: you can't rely on OSS for any "real time" support
 - Not anymore: companies stood up "around" OSS technology, dedicated to providing support
 - Apache Hadoop: Cloudera
 - Apache Solr/Lucene: Lucid Imagination
 - Apache Cassandra: Riptano
 - Apache Maven: Sonatype
 - ...others?
 - How good is the documentation for the OSS Project?
 - Healthy wiki?
 - Healthy "cook books"?
 - Is the documentation baked in or separate?
 - Javadocs (or equivalent) on all public methods?
 - Use cases?



- Build expertise and intellectual capital in existing open source technology
 - If you are pulling it in and using it
 - Involves: “Community” model, but also works in “Help Desk”
 - Involves: active participation
 - Involves: friendly license (especially if you want to redistribute)
- If you are building your own technology that you want to put into open source
 - Others may “help” you
 - Solve challenges at ZERO cost to you (well not ZERO but minimal)
 - Free cycles of interest





- Case Study: Mailing lists communications*

A Bad Email

Subject: Problem with crawling

I am having a problem with crawling the internet. It just seems that it is taking a long time. Does anyone know why crawling takes so long.

Me

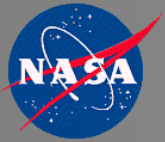
A Good Email

Subject: Crawl on 20K pages taking 4 hours

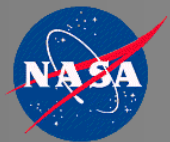
I am using Nutch .8 branch over a cluster of 3 machines each running redhat linux and java 1.5_10 with 500G hard drives, 2.8 Ghz processors, and 2G of ram. I am trying to fetch 20K urls and the fetch process completes fine but when it gets to the reduce process, the cpus go to 100% and the process seems to spin indefinitely. I did a kill -SIGQUIT on the process and it seems to be stuck on the Regex normalizer class. Has anyone experienced similar problems or know what might be causing this problem.

Me

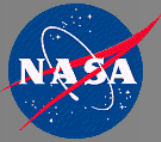
* Taken from: http://wiki.apache.org/nutch/Becoming_A_Nutch_Developer



- Mailing list interaction *very important*
 - Most immediate feedback you get when contributing *to* open source, and also when creating *your own* open source project
 - Is the biggest thing that turns people away to start out with
 - “who are these nerds and why don’t they have any manners?”
 - It’s all about “learning” their “language”
 - Sure, the cost may be high in terms of ego and attitude, but the reward (free work!) is well worth it
- Following and understanding the practices of the OSS project are *hugely* important
 - Know the license for an OSS product
 - Know whether they are “community” versus “help desk”
 - Know its history
 - Read its documentation
 - Be informed!
- Danger: thinking the OSS community is your personal helpdesk (or vice versa, you are its)

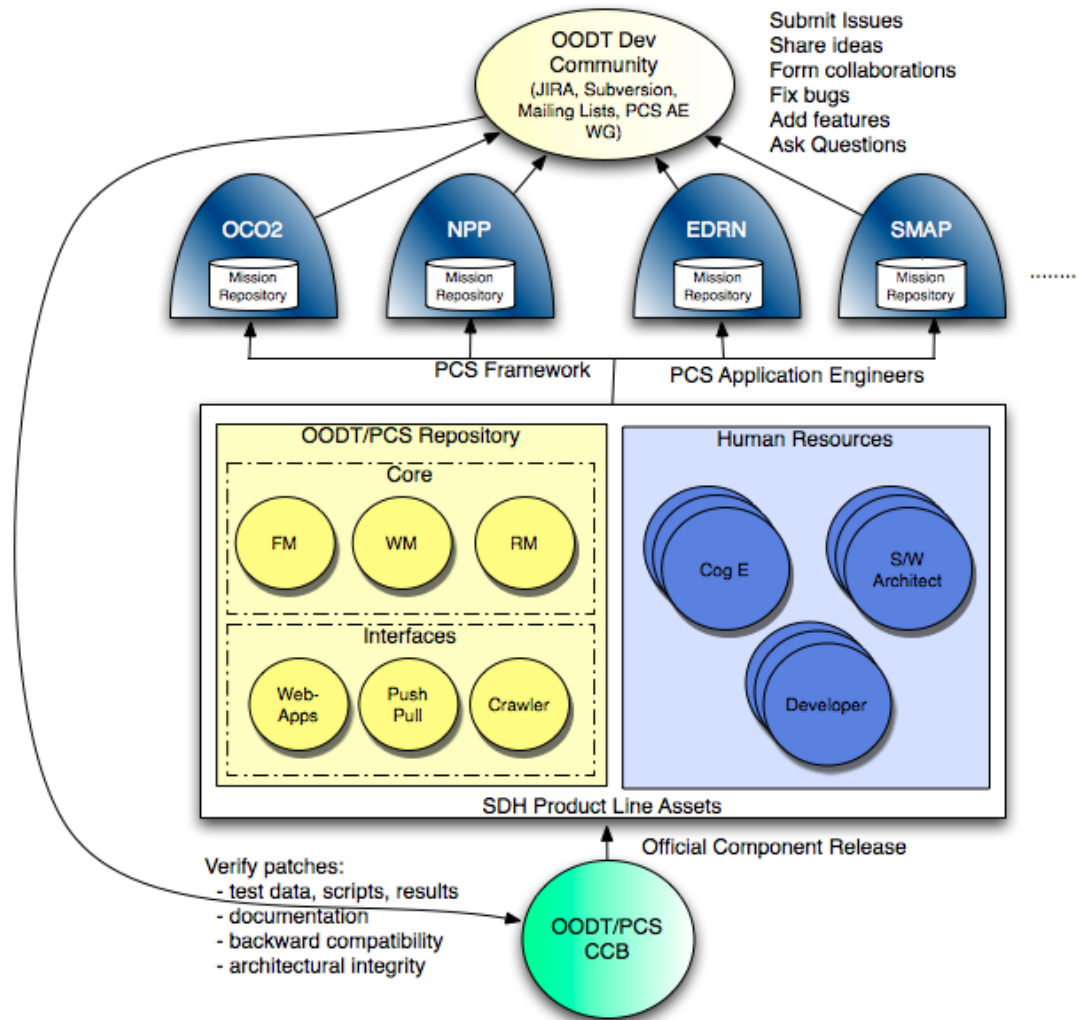


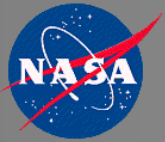
-
- Concerns
 - Insulation: how do I insulate my project from OSS change but still use OSS?
 - How do I make my project's OSS software available for redistribution?
 - What technologies are the “best” for my project?
 - What communities are the best?
 - Answers:
 - See next slide
 - It depends
 - It depends
 - It depends



Insulation: one strategy

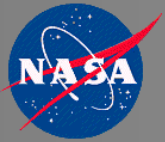
- Missions maintain their own local CMs
- Local mission CMs contain forks of existing OSS software
 - Forks can be patch based or CM based
- Changes found particularly effective are discussed within the comm. And eventually brought before a CCB that reviews their generality, etc.





- “War Story 4”
 - Sharing software across missions
- I’ll use your software if you use mine
 - NASA mission X makes a fork of OSS software S and adds a new great component W, and improvement over an existing component M, to it. NASA mission Y, started around a similar time also forks S, but uses M as is and doesn’t build a new component like W. NASA mission Z comes along and is interested in using OSS, but wants to make sure they are on the same baseline as missions X and Y. NASA missions X, Y and Z huddle up and try to decide how to sort out using OSS, and getting onto the same baseline.





- Option 1
 - Create a “mirror” of S shared by Missions X, Y and Z
 - Missions X, Y and Z may still maintain local repos
 - X checks new W in shared repo
 - Y ignores W and uses M from shared repo
 - Z uses shared repo to stay on common baseline
 - Pros:
 - Insulate X, Y and Z from changes to S
 - Z has 1 place to get mission forks
 - Requires coordination between X, Y and Z
 - Cons:
 - Requires coordination between X, Y and Z!
 - If W touches M, or changes M in any way, then Y is not insulated from X’s changes
 - Requires representatives from X, Y and Z to “take ownership” over shared repo’s maintenance

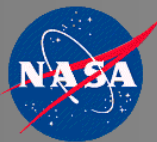
- Option 2
 - Missions X, Y and Z may still maintain local repos
 - X checks new W in its local mission X repo as patchset to S baseline
 - Y ignores Mission X's W and uses M from S
 - Z uses Mission Y's W patch to S to stay on common baseline
 - Pros:
 - Insulate X, Y and Z from changes to S
 - Z has 1 place to get mission forks
 - Requires **NO** coordination between X, Y and Z
 - Cons:
 - Requires understanding of patch maintenance, and patch merging into local repos

- Option 3
 - Missions X, Y and Z may still maintain local repos
 - X checks new W in its local mission X repo as branch of S baseline
 - Y ignores Mission X's W and uses M from S
 - Z uses Mission Y's W branch to S to stay on common baseline
 - Pros:
 - Insulate X, Y and Z from changes to S
 - Z has 1 place to get mission forks
 - Requires **NO** coordination between X, Y and Z
 - Patch maintained as branch using CM tool's super tools (history, changelog, etc., can *even generate patches* if desired)
 - Cons:
 - Requires understanding of branch maintenance, and branch merging into local repos



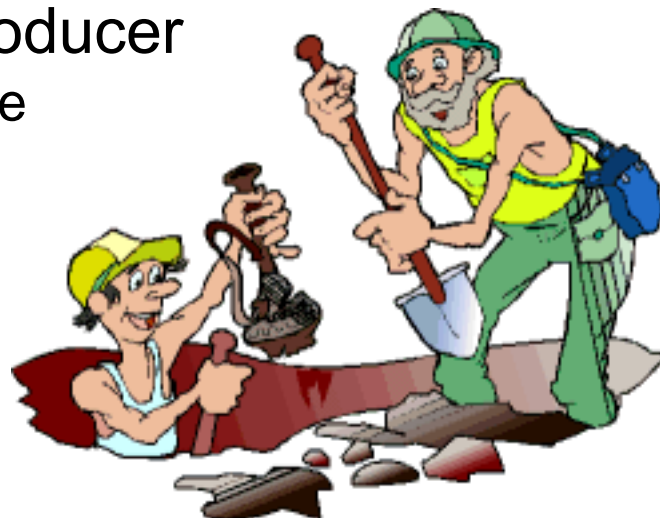
-
- How to *design* for open source?
 - Extension points
 - Places in your code for “plug ins”
 - Plugins should be well documented
 - Public facing “javadoc” (or similar)
 - “Cookbooks” for integrating
 - Wiki pages
 - Baked in documentation
 - Use of shared component marketplace
 - Many times PL specific
 - PyPI – Python
 - Maven Central/Ibiblio – Java
 - CPAN – Perl
 - STL, GNU, Autoconf, Make, etc. – C/C++
 - YMMV, some are
 - Webified, have package metadata, searchable, etc.

-
- Review architecture
 - Interrogate buy versus build
 - Why did you buy?
 - Why did you build?
 - WHY DID YOU USE OPEN SOURCE?
 - Or why didn't you?
 - Review regularly
 - Review it again
 - And again
 - Promote open source infusion
 - Promote construction of open source components inside of your data system implementation and their redistribution



Leverage active technologies

- Don't expect to use Haskell and have a huge OSS community there to assist you
- Some modern active OSS languages
 - Python
 - Perl
 - Ruby
 - Java (though Oracle some technology company is pushing the boundaries of that)
- Don't just be a consumer, be a producer
 - Pick up a shovel and help dig the hole
 - Instead of overlooking the OSS community and telling them how the hole should be dug to meet your needs
 - Will help you get needed OSS feedback for your technology

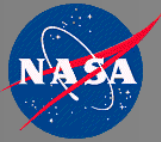




-
- Understand patent law
 - Or pay people to understand it for you
 - Understand OSS licenses and what you sign up for when you use relevant technologies and their licenses
 - Look for open APIs and open specs
 - Understand what “open” means
 - Vendor lock in
 - Try to avoid it through the use of all the techniques of OSS which we’ve discussed
 - Engage the community and understand what’s “really” free and what isn’t
 - Watch millions of dollars be spent arguing this currently in court by 2 unnamed companies
 - Get legal’s buy-in
 - Don’t insulate them



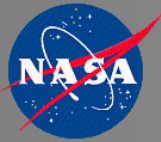
- Understand “marks”
 - Trademarks
 - Their implication to patents
 - What it means to say Java TM versus just “Java”
- What will NASA support in terms of licensing?
 - Center specific
 - Depends on IP, export control, ITAR, compliance, commercialization
 - If you make it through all of the above and the software is cleared for release, you really are in the driver’s seat
 - No NASA wide guidelines
 - Most or all communities allowed
 - Most or all OSS approved licenses allowed
- Is there a Contributor License Agreement (CLA) or Corporate Contributor License Agreement (CCLA) to sign?
 - What are its implications?



Let's bring it all together

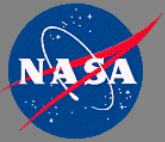
- Next slide includes case study that addresses many of these issues, let's discuss





- NASA's Object Oriented Data Technology (OODT)
 - Funded out of NASA's Office of Space Science in 1998
 - Designed, implemented, deployed, operationalized, and refined over the past 7 years across multiple scientific domains
 - Planetary Science, Earth Science, Cancer Research, Space Physics, Modeling and Simulation, Pediatric Intensive Care
 - Runner up NASA software of the year in 2003
- Meeting held June 15, 2007 at JPL with ASF President Justin Erenkrantz
 - Develop plan moving forward to bring first NASA project to Apache
 - Discuss obstacles, sponsorship
 - Discuss outlook



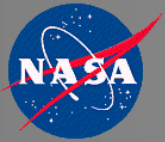


ASF: some fun facts

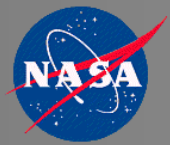
- Largest open source software development entity in the world
 - *Over 2300+ committers*
 - *Over 3500+ contributors*
 - *84 Top Level Projects*
 - *36 Incubating*
 - *30 Lab Projects*
 - *8 retired projects in the “Attic”*
 - *Over 1.2M revisions*



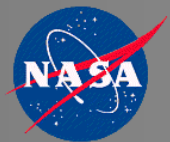
- Over 10M successful requests served a day across the world*
- HTTPD web server used on 100+ million web sites (52+% of the market)*



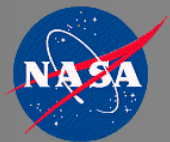
- Come up with incubation proposal
 - Chris Mattmann was one of the principal contributors to the proposal for the Tika project, and to other Incubation activities (Apache SIS)
 - Send out emails to the Incubator mailing list
 - Look for mentors
- Get sponsorship from ranking Apache PMC member or board member
 - Justin and others
- **Top-level** project versus sub project outlook heading out of incubation



-
- Monthly Updates (for first 3 months, then quarterly)
 - Status
 - Progress
 - Community
 - Acceptance
 - Plan for exiting incubation
 - How to have a solid user base
 - How to operate as a unit in the Apache way
 - Maintenance of user interest and community going forward



-
- **Varies**
 - Quickest graduation has been around 6 months
 - Some have been there for a year or two years
 - **No hard deadlines**
 - It'll be ready when it's ready

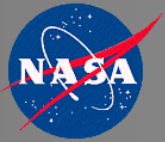


-
- How to deal with Open Channel distributor
 - Top level project versus sub-project
 - JPL/NASA legal issues

-
- JPL to tackle legal issues
 - Is OODT releasable as an Apache product
 - <http://www.apache.org/licenses/software-grant.txt>
 - This needs to be signed by parties that be by JPL
 - Contributor License Agreement
 - Do we need a corporate one?
 - In parallel to this
 - Draft OODT incubation proposal
 - Start identifying who would initially be interested
 - More external, non-JPL people who are interested, the better
 - Justin to get slides from other incubator people



- Worked it out with JPL legal
 - Turns out the ASLv2 license is extremely friendly and is something that JPL (note not all of NASA) was amenable to
- Developed OODT incubator proposal
 - <http://wiki.apache.org/incubator/OODTProposal>
- Found willing Apache mentors besides Justin
 - Jean-Frederic Clere, Ross Gardler, Ian Holsman
- ...Put OODT at Apache!



-
- What do y'all think?
 - Potentially work this within the SRWG
 - High end versus Low End version of these circumstances for OSS?
 - Lots here that may apply to low end and immediately help address problems that people like John are dealing with in his long term archive scenario
 - What are the criteria for when to start with open source
 - Cost of the infrastructure to support it
 - Cost for data center to run an OSS
 - Closed situation for infra
 - Aspect of having to start up each data center's *own* infrastructure
 - Cost of using a commercial OSS provider
 - External agencies, etc.
 - What does it mean if software is in Sourceforge
 - For preservation?



-
- As a community (NASA ESDS)?
 - SRWG interested in this
 - But also cross cutting to other groups (TWIG, SPG)
 - As NASA?
 - Best Practices?
 - Federations?
 - What are other govt' agencies doing?
 - Use DOE as a model?
 - GO-ESSP?
 - Do our own thing?
 - Who cares?



-
- THANKS FOR ATTENDING!
 - Feedback welcomed!
 - Software Reuse Working Group
 - *We are interested in these topics!*



http://twitter.com/esdswg_reuse



<http://www.facebook.com/group.php?gid=117453644936920>



<http://www.linkedin.com/groups?gid=2964349>

- **Contacts:**
 - Chris Mattmann (chris.a.mattmann@nasa.gov)
 - Robert Downs (rdowns@ciesin.columbia.edu)
 - James Marshall (james.j.marshall@nasa.gov)