The Software management team strategic plan

1 Team Strategy

Problem: Currently rpm is threatened by various community groups that don't use it for software installation and management. If we don't change the way we deal with things, we risk getting out of the main area of interest.

Goal: Our strategic goal for the next 3-5 years is *for the rpm stack to once again become the central place where all the information about installed SW is kept.*

Strategy summary:

a) Don't break anything important for existing users
b) Become more attractive to OS users
   - Integrate well with Desktop SW management applications for non-experienced users
   - Offer better usability of yum and related tools for advanced users
c) Become more attractive for developers
   - Start thinking about integration/collaboration/coordination with as many of devops-related communities as possible to offer them simple and effective way how to reach users running Fedora, RHEL and derivatives.

1.1 Don't break anything important for existing users

This is the part that we have been doing all along. We still need to think about current users, even those using older releases of RHEL, CentOS and other more conservative distributions based on Fedora. However while doing this we need to deliver new features of rpm faster. If there are some roadblocks in other part of existing infrastructure, we need to escalate them to responsible people and try to find solutions for them.

1.2 Become more attractive to users

This is the part which covers integration with Richard Hughes' gnome-software and the new PackageKit backend. But there are also some problems on the yum side that we should look into and if we do so, we have a high chance to improve user experience significantly. The feature request (A1) can serve as an example of such a problem.

We should also think about connecting rpm with other packaging technologies. The goal of the new gnome-software is to connect with and cover as many existing software installation methods as possible. We should follow that example – while our target area is a bit different, the idea is valid for us as well.

1.3 Become more attractive for developers

There are two groups of developers in this context: traditional developers and devops. In every group there are some users we can target. In the case of traditional developers we can target for beginners or developers without any deeper knowledge about rpm packages. The area we want to focus on here is a simple creation of quick & dirty packages.

In the case of the devops we are talking about groups that are usually used to their way of packaging because it works the same on other platforms. In some cases the difficult rpm repackaging of their packages is the roadblock for wider adoption. In other cases it's a fundamental incompatibility between the two ecosystems. These are some of the areas we might want to focus on:

- SCLs
- OpenShift use cases
- Python, Perl, Ruby modules
- Java (similar situation as Ruby but still a bit different)

2 Future feature plan

Please note that this is just an overview, if you need to know more information about specific feature, don't hesitate to ask. We also plan to start a separate discussion on some topics in the list when the time is appropriate.

2.1 In progress (0-1 years) + prioritized (A):

(A1) Delta metadata in yum and dnf

- this is in late stages of development for yum, also a long requested feature
- aligns with strategy goal 1.2

(A2) DNF: Python 3 support and Anaconda integration

- without these two the dnf can't be tested

(A3) New format of rpm payload

- this is in late stages of development and it doesn't have any dependencies to other features
- aligns with strategy goal 1.3, usable especially for cloud and virtualized environment

(A4) RPM plugin interface

- there are many possible application of this, also some other features will benefit greatly if the interface is finished

2.2 Near future (1-2 years) & prioritized (B1):

(B1-1) DNF finalization

- aligns with strategy goal 1.2

(B1-2) Improved SELinux support

- this depends on (A4)
- aligns with strategy goal 1.2

(B1-3) Rich dependencies

- at this point we don't have to have an implementation, just a design will do to calculate with it when adjusting spec files and/or rpmdb
- will align with almost every level of better user experience with RPM

(B1-4) New format of rpmdb

- in progress but deferred due to its relation to (B1-3) and (B1-5)
- loosely aligns with strategy goal 1.2

(B1-5) 2B. Separation of yumdb

- related to (B1-3) and (B1-4)
- loosely aligns with strategy goal 1.2

(B1-6) 2C. File scanner

- discussed in the context of OpenShift file categories
- the implementation might need to be discussed again
- loosely aligns with strategy goal 1.3

(B1-7) 3. Simplification of spec files

- declarative rpm macros, %user, %group
- Better git integration (rpm)
- Checksums in spec file (rpm)
- some simplifications derived from the file scanning engine
  - automatic subpackages
  - semi-automatic %pre and/or %post sections
  - …
- aligns with strategy goal 1.3

(B1-8) Retired packages

- createrepo, dnf
- aligns with strategy goal 1.2, retired packages can cause a lot of pain in the current state

2.3 Mid-term future (2-5 years), high priority (C1):

(C1-1) Repository tracking

- dnf
- aligns with strategy goals 1.2 and 1.3

(C1-2) rpmbuild should fail on non-utf8 specs

- will improve the quality of spec files by raising a bar a bit

(C1-3) Simple rpm creation:

- integration of tools like gem2spec and similar
- automatical suggestion of BuildRequires?
- aligns with strategy goal 1.3

(C1-4) Simple SCL creation:

- spec2scl
- aligns with strategy goal 1.3

2.4 Mid-term future (2-5 years), medium priority (C2):

(C2-1) Allow logging mechanisms directly in rpm

- most of this is done so we might get it sooner
- aligns with strategy goal 1.3 by allowing progress of scriptlets to be logged

(C2-2) Hardware specific installation (dnf, plugin, lspci)

- aligns with strategy goal 1.3

(C2-3) Non-locking operations (dnf only)

- aligns with strategy goal 1.3

(C2-4) Instance tracking

- for OpenShift and other container-based technologies if they keep being in the middle of interest
- aligns with strategy goal 1.3

(C2-5) Tracking of the network-mounted software

- this should be implemented separate of the main rpm code base
- aligns with strategy goal 1.2 in enterprise environments

2.5 Mid-term future (2-5 years), low priority (C3):

(C3-1) RPM thread safety

- in progress, moved here just to signal its low priority for the moment
- aligns with the strategy goal 1.3 but only for a very specific group of developers, that's why it's been de-prioritized

(C3-2) Tracking updated daemons (rpm plugin)

- only very limited usage of this RFE (the functionality is already present, this will just move it to the right place), therefore it's low on our list

(C3-3) Improved RPM verification (rpm)

- only very limited usage of this RFE, therefore it's low on our list

(C3-4) Configuration tracking

- only very limited usage of this RFE (the functionality is already present, this will just move it to the right place), therefore it's low on our list

2.6 Long-term future (D)

(D-1) Modular rpmbuild

- the goal here should be to enable different formats of input than spec files
- some example conversations:
  - https://twitter.com/stahnma/status/378036237409345536
  - https://twitter.com/samkottler/status/377977813900746752