High-level vision of SW Management

Manifest:
To achieve an evolution that makes sense, we have to stop thinking about *Package management* and start thinking about more generic *Software management*.

A SWM needs to provide mechanisms to handle all general tasks involved in ensuring how applications are made available on a system, how they are taken care of over their lifetime, how applications can be run and finally how applications are removed.

Overview:
What does it mean to do Software management? When you manage your software, you have to take care of three different areas:

- software installation
  - basically exactly the thing that rpm stack does today (install, update and erase sw)
  - means to centrally manage all SW installed on the computer (including that not installed via SW management stack)
- software configuration
  - configuration profiles, default configuration, guided configuration
- software runtime
  - how does the software run - standard application, daemon, containerized application, containerized daemon, ...

We realize that Software management can't do all of this itself but it has to offer solid base platform for all this functionality. Package maintaners then can utilize this base to provide the functionality itself. The same applies for other projects that are involved in SW management, solving for examle cases that are out of scope of SW management. Our stack shall provide interfaces for these projects.

Software installation/update/removal

- installation of software
  - standard installation tasks - install / uninstall / update / downgrade software
  - installation of sw by non-root users
    - try to optimize disk space by not keeping multiple copies of that sw
    - try to maximize user security by having a web of trusted distributors of sw (some sort of signing mechanism, issuing certificates for trusted upstreams??)
    - try to further improve security by being tightly connected with the runtime part - run user-installed sw from untrusted sources only in containers
  - conditional installation (conditioned upon HW or any system capability)
  - other non-standard installations - *the scope here has yet to be evaluated*
    - installation to custom dir (install prefix)
  - product installation and uninstallation
    - in this vision, product is a closely bound set of packages picked and integrated by any custom vendor
    - product may be handled either as a set of packages or as one unit
  - installation / management of multiple versions of a single library or application
- software license argeements
  - SWM can't cover the entire area but it can again provide a base for this functionality (in a

form of plugin interface)
- seamless updates of software
  - *special case:* update of application that is currently running without the need to restart it
  - categorize updates (security / core / library / others / ...)
    - set different policies for these categories? (ban/notify admin/offer during other oactions/apply automatically/ ...)
- listing of sw currently installed/available/outside of the scope (i.e. files that don't belong to any sw installed via the central management)
  - scan the computer for sw (most likely libraries) that got installed but is not necessary any more
- SW verification
- creation of custom SW
  - if I install any SW outside the scope of SW management system (swms), there should be a way how to make swms aware of this SW
  - this might be also cooperating with different parts of the system like linker (if I make swms aware of my locally installed library, I should have the option to configure linker so it can scan for it and swms to offer it as "provide")
- providing information about the software, both to users and other projects via APIs
  - 1st level information - the information given in packages
  - 2nd level information - the information outside of packages (like comps)

Software configuration

- packager-defined configuration profiles, pre-defined default
- common tasks:
  - automatic installation - pick sensible default
    - note: each package will have to contain default profile for non-interactive installation, otherwise it won't build
  - offer pre-defined profiles, user can pick the best suiting one
  - fine-tuning of profiles if possible
  - re-configuration at any point in time after package is installed
  - configuration migration
  - backup of current configuration / restoration of current configuration

Software runtime

- close cooperation with systemd and similar tools
- choose how sw should be run
  - default option (from maintainer), fallback mechanisms, user's choice
- prepare environment for such run
  - migration of these environments

What a SWM system does NOT handle:

- Distribution of software -> Delivery mechanism
  - SWM can and should still provide
    - tooling to create a full "distribution" or "repository" of applications
    - a way how to download applications from repositories
- entitlements management
  - we won't handle distribution of software - no sensible way to handle entitlements
  - SWM may however provide an interface for 3rd-party extentions that can handle entitlement management
- Custom configuration of software besides the profiles pre-set by maintainers
  - this task in general is up to System management