Anaconda Installer

Architecture

Presented by
Vratislav Podzimek

Today's Topics

1. What is Anaconda?
2. Why (UI) rewrite
3. The NewUI
4. Architecture
   a. Data representation
   b. Hub&Spoke model
5. Threads and communication
6. Initial Setup
7. Addon development

fedora

What is Anaconda?

- OS installer for Fedora, RHEL and derivatives doing everything else but installing
- Python package (pyanaconda) + main script, dracut lib and unit files
- supposed to:
  - support both automated (kickstart) and manual installations and also the combination of both
  - support graphical mode and text mode for old sequential-only terminals (s390x)
  - be simple but in the same time complex

Why (UI) rewrite

- decided at FUDCon Tempe 2011
- main reasons:
  - non-modern UI born more than 10 years ago
  - UI-controlling logic mixed with the installation logic
  - basically a single thread stepping the Gtk main loop manually
  - ksdata + installdata + UI elements attributes
  - gtk2 and pygtk based GUI mixing Glade files and widgets created in the code
  - ncurses based text mode with separate code base

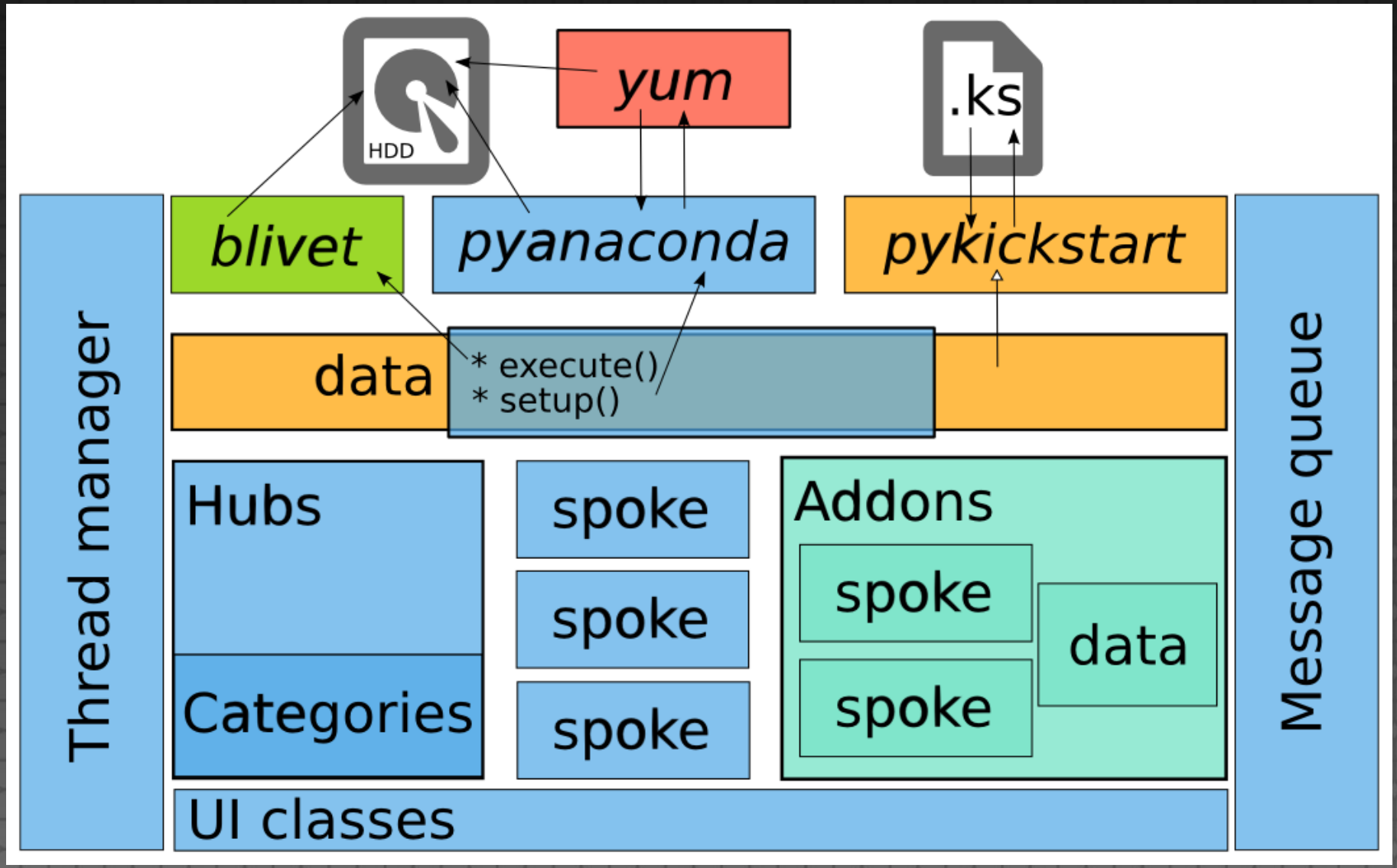
The NewUI

no more scary Anaconda

- modular, extensible, multi-thread
- Hub&Spoke as the basic model
- graphic designed by Máirín Duffy
- kickstart => self.data => kickstart
- customization screens during package installation
- code shared with the new purely textual text mode and Initial Setup
- more transactional
- pyanaconda.storage separated as *blivet*
- 47899 insertions(+), 64380 deletions(-)

fedora

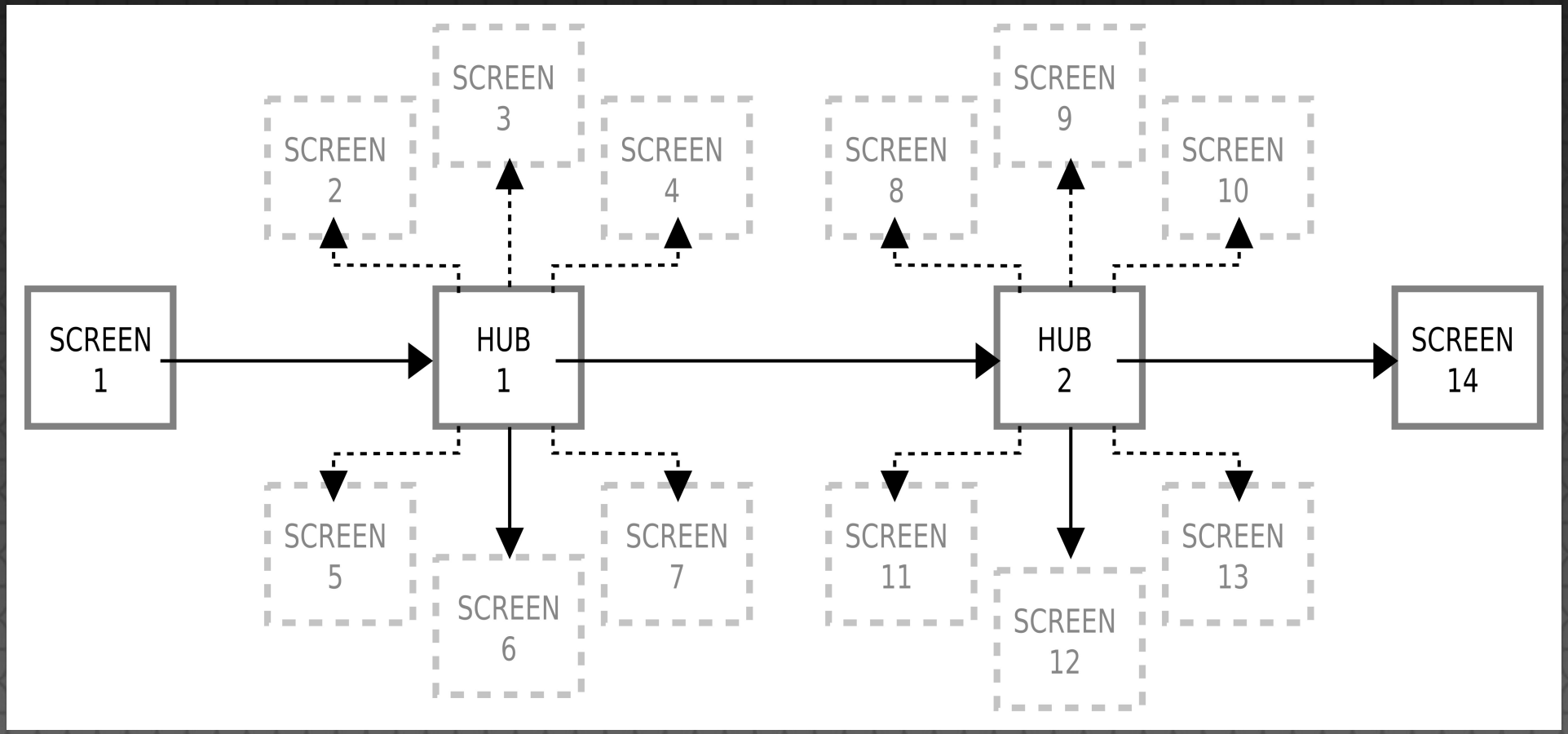
Architecture



Data

- all stored in the pykickstart.KSHandler instance
- life cycle:
  - loaded from the kickstart file (if any)
  - updated with user's choices made in the UI
  - used to drive the installation
  - written out as kickstart file
- tree structure
- read, updated and written out also by the Initial Setup
- *setup* and *execute* methods doing the installation logic

Hub&Spoke model



Hub&Spoke model

- easy and fast access to everything
- no need to visit every spoke
- overview of the settings (updated by background threads)
- layout with great support for extensions
- usable for both graphical and text mode

Hub, Spoke & Anaconda

fedora

Hubs

- standalone spokes and hubs are dynamically collected from the predefined places
- categories and spokes are dynamically collected for every hub
- the Summary hub and the Progress hub
- continue possible once all mandatory spokes are completed
- automated installations show summary and progress, but continue automatically (unless user changes anything manually)

Spokes

- StandaloneSpoke and NormalSpoke classes together with custom windows (Gtk widgets)
- marked for use in the Initial Setup or not
- supposed to contain only the UI-controlling logic, installation logic in blivet's, pyanaconda's and self.data's methods and functions
- UI defined in a .glade file (all that is possible)
- the *showable* property determines if the spoke should be shown or not

Normal Spoke

- basic building block of the NewUI
- API defined attributes:
  - *uiFile, mainWidgetName, category, icon, title,*
- API defined methods:
  - *initialize* and *refresh*
  - *apply* and *execute*
- API defined properties:
  - *ready, status*
  - *mandatory* and *completed*

fedora

Threads and Gtk

- Gtk main loop running in the main thread
- two Gtk main loops running in separate threads crash X server
- locks allowing controlling Gtk from multiple threads no longer supported (and never recommended)
- GLib.idle_add and related functions are the only supported way
- decorators and functions to facilitate usage
  - *@gtk_thread_wait*, *@gtk_thread_nowait*
  - *gtk_run_once*

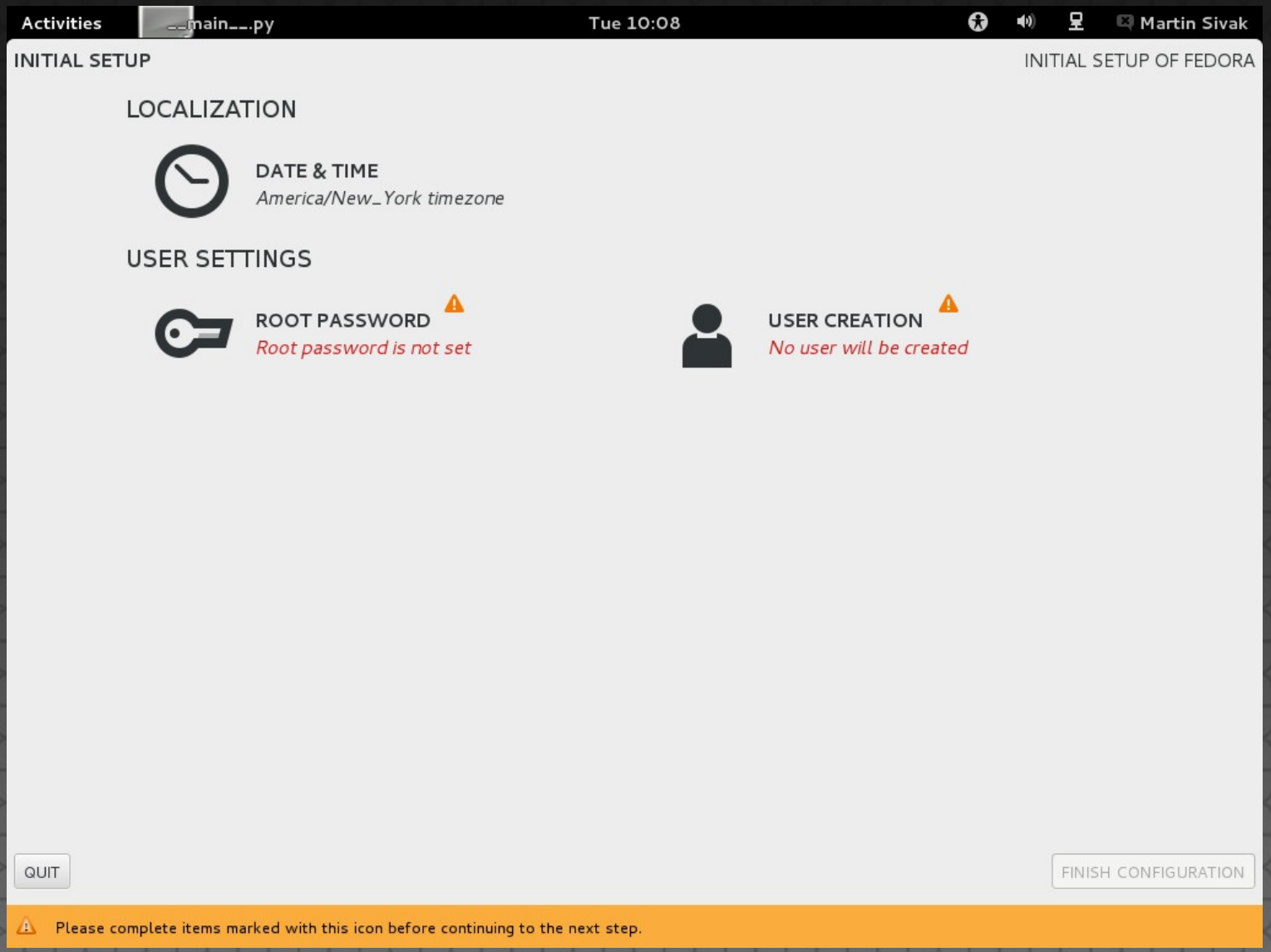
Threads and messages

- threads for all long lasting actions
- ThreadManager singleton and AnacondaThread class facilitating logging and threads usage (also exception handling)
- two message queues
  - hubQ for spoke to hub communication
  - progressQ for reporting and updating installation progress
- experimental implementation also for the text mode (GLib/Gtk main loop, almost always waiting for input)

Initial Setup

- Firstboot replacement, but the old one has to survive because of the legacy 3rd party plugins
- basically only 40 lines of code reusing the code and screens from Anaconda
- reads kickstart file produced by Anaconda and writes a new one at the end
- coordinates screens with Anaconda and Gnome Initial Experience
- targeting F19

Initial Setup



fedora

Addons

- many teams want to have something set in the installation process (or first boot), but we cannot develop and maintain all that stuff
- examples of possible addons:
  - AD/kerberos realm join with realmd
  - SCAP security profile
  - subscription management
  - Emacs :)

Addon development

- kickstart part (must be implemented):
  - class parsing lines from the special %addon section and storing data from them as its attributes
  - lives in the self.data.addons.* subtree
  - methods to modify runtime environment (*setup*) and configure installed system (*execute*)
- UI part (optional):
  - GUI and TUI spokes reading data from self.data and modifying them
  - can be marked also for the Initial Setup
- altogether like 100 lines of code

fedora

Addon structure

- a directory under /usr/share/anaconda/addons
- top-level directory named after the addon (e.g. org_fedora_hello_world)
- subdirs for particular parts -- ks, gui, tui
- placed to the installation tree by lorax or with product.img -- still being decided
- classes automatically collected and used if they are subclasses of classes defined by the API

Addon HOWTO

- well-commented Hello world addon [2]
- sources of realworking instances (coming soon)
- Anaconda Addon Development Guide [3]
- questions and answers on the anaconda-devel mailing list
- *anaconda*, *anaconda-widgets* and *anaconda-widgets-devel* packages installed
- *make runspoke* target in the Anaconda's Makefile

Addons FAQ

- Why such a bad name?
  - We don't have any better.
- What happens if the addon for some %addon section is missing?
  - Nothing. The %addon section is ignored and just pasted to the resulting kickstart file.
- Which languages are supported?
  - Python only.

Addons FAQ cont.

- Why this %addon section marking the functionality as being amended? Can't addon that needs only one line just register a new kickstart command?
  - It is possible, but the problem is with the *ksvalidator* tool that needs to distinguish between invalid command and a command of a missing addon.

Summary

- still work in progress
- multi-thread, Gtk3 based, better user experience
- better documentation, better maintainability
- modularity
- a lot of code shared between GUI, TUI and Initial Setup
- extensibility, easy to write addons
- altogether less scary for both users and developers.

Links

- [1] Anaconda/NewInstaller wiki
- [2] Hello world addon
- [3] Anaconda Addon Development Guide
- [4] Anaconda sources

Questions?

Contacts:

vpodzime@redhat.com

anaconda-devel-list@redhat.com

#anaconda @ Freenode