Transactional RPM (RPMT) Re-Engineering

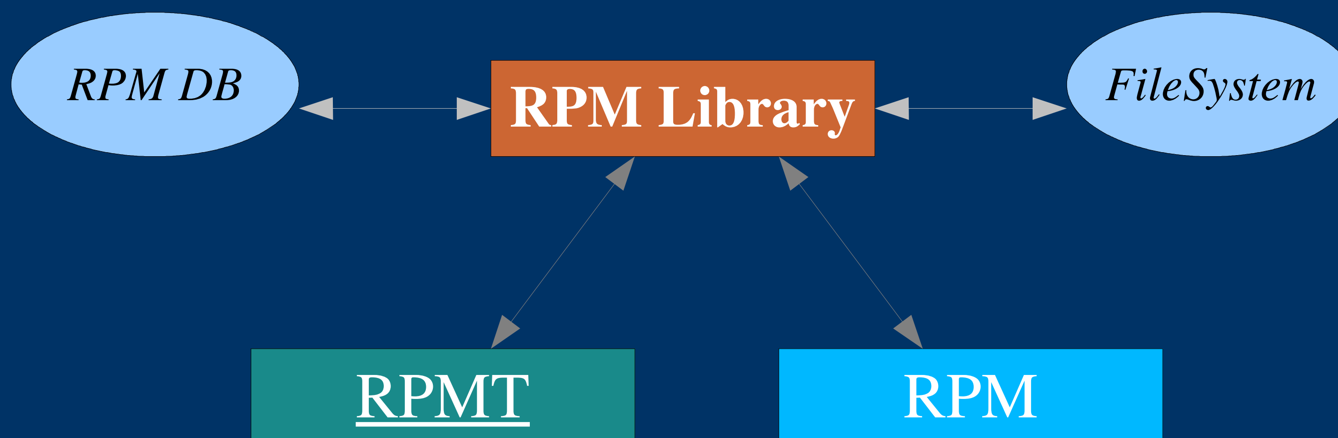
- Introduction
- Motivations
- Investigations & possible solutions
- Propositions
- Conclusion

Nicolas Pinto Supervisor : Germán Cancio Meliá

Introduction
Motivations
Solutions
Propositions
Conclusion

Actual Implementation of RPMT

- RPMT (RPM Transactions) is a small tool on top of the RPM libraries, which allows for multiple simultaneous package operations resolving dependencies (unlike RPM)
  - Example: ‘upgrade X, deinstall Y, downgrade Z, install T’ and verify/resolve appropriate dependencies...
- Does use basic RPM library calls (API), no added intelligence
- Written in ANSI/C



Introduction
Motivations
Solutions
Propositions
Conclusion

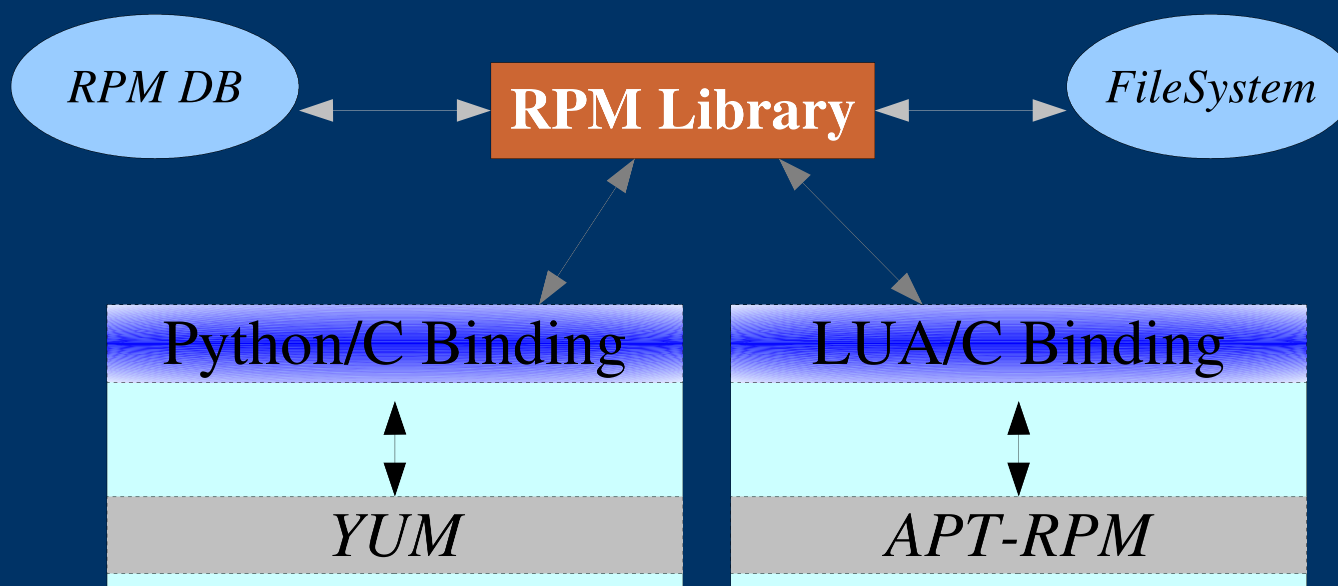
Why re-engineering this tool ?

- Because C RPM libraries are very unstable:
  - changing in time too frequently,
  - and poorly documented, etc.
- We would like to avoid having to recode RPMT in C every time there is a new RPM version available (every 3-4 months).
- Therefore, we need to understand if RPMT could be recoded using another language which provides a more stable interface to RPM API.
- The new implementation has to be :
  - well designed,
  - as flexible as possible,
  - ready to "fight" future versions of RPM libraries.

Introduction
Motivations
Solutions
Propositions
Conclusion

Investigations & existing solutions

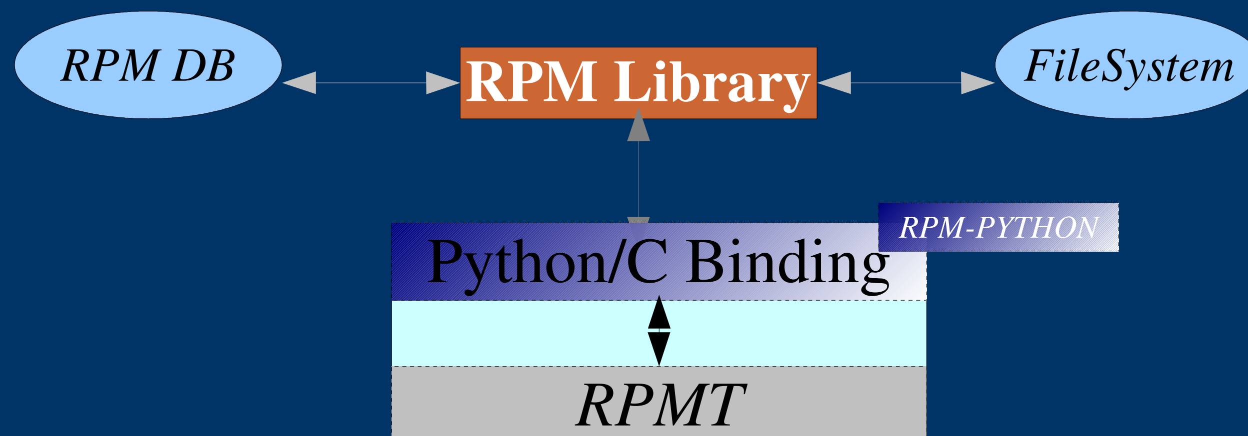
- Existing tools are using powerful scripting languages combined with binding libraries :
  - *Python* (YUM, up2date, anaconda),
  - *LUA* (apt-rpm),
  - *Ruby, etc.*
- We need to understand how stable the interfaces provided by them are and choose the best one according to our needs.



Introduction
Motivations
Solutions
Propositions
Conclusion

Proposition 1 : We should use Python

- As a matter of fact, on recent Red Hat systems, only *Python* is likely to succeed in the requirement of having a current stable RPM binding available, as RH uses it for their configuration and administration tools.
  Note : *Perl*, which is also commonly used, doesn't have a **good** binding library.
- Moreover, *Python* is an object-oriented programming language and supports procedural programming :
  - we can select the most suitable programming paradigm for each part...
- *Python* is simple, powerful and provides a vast standard library.
- You can easily use it with other high-level language such as C or Java.



Introduction
Motivations
Solutions
Propositions
Conclusion

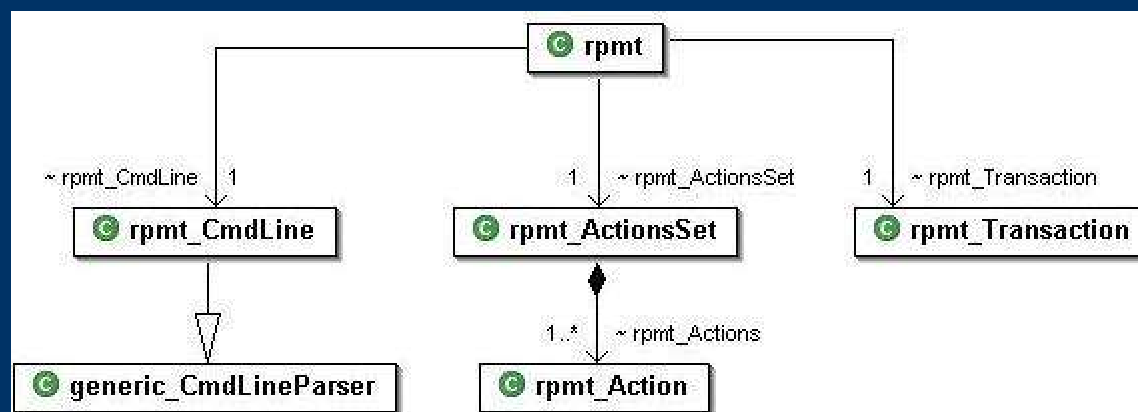
Proposition 2 : The “Attack” Plan

- First of all, create a hard-coded set of test in *Python* using the binding library (RPM-PYTHON) to see if each needed functionality is available.
- Second, correctly design the new RPMT application.
- Then, implement it using *Python* language and *RPM Libraries* version 4.2.3 .
- Then, see what is required to port it to *RPM Libraries* version 4.3.2 .
- Finally, adapt the design of the application according to the needed changes.

Introduction
Motivations
Solutions
Propositions
Conclusion

Proposition 3 : Simplified Design

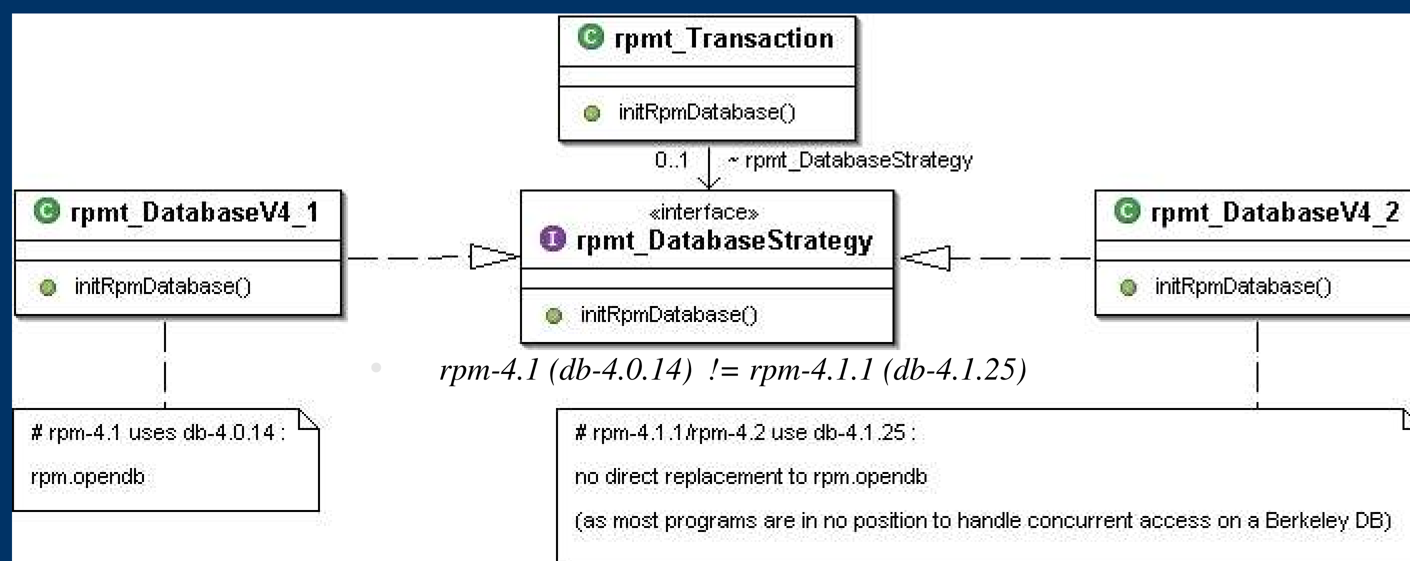
- RPMT performs 3 main jobs :
  - parses the command line and set the rpm flags according to it,
  - parses the given actions file (install, deinstall, upgrade packages),
  - updates the transaction set and runs it.
- Simplified design (class diagram) of the new RPMT, using OO paradigms :



Introduction
Motivations
Solutions
Propositions
Conclusion

Proposition 4 : Strategies for quick adaptations

- Using the *design pattern Strategy* could help us when a re-implementation is required :
  - first, we have to identify each set of RPM-operation we are using (ex: database access, transaction set operations, etc...).
  - then, we create a strategy for each set (if required).
  - finally, we implement a strategy per version of RPM Libraries (if required).
- Example : opening the RPM database (Berkeley DB)



Introduction
Motivations
Solutions
Propositions
Conclusion

Conclusion

- *Python* facilitates Rapid Application Development and Prototyping.
- Using a language like *Python* cuts down development time drastically - with the added advantage that you get highly robust and flexible code.

Learning and Coding in progress...

Introduction
Motivations
Solutions
Propositions
Conclusion

Thank you for your attention

- Questions ?
- Criticism ?
- Advices ?
- Comments ?