Hierarchy Software Development Framework (h-dp-fwk) Project

A. Zaytsev[1, 2]

[1] Budker Institute of Nuclear Physics, Novosibirsk, Russia

[2] E-mail: Alexander.S.Zaytsev@gmail.com

Abstract. Hierarchy Software Development Framework provides a lightweight tool for building portable modular applications for performing automated data analysis tasks in a batch mode. The history of design and development activities devoted to the project has begun in March 2005 and from the very beginning it was targeting the case of building experimental data processing applications for the CMD-3 experiment which is being commissioned at Budker Institute of Nuclear Physics (BINP, Novosibirsk, Russia). Its design addresses the generic case of modular data processing application operating within the well defined distributed computing environment. The main features of the framework are modularity, built-in message and data exchange mechanisms, XInclude and XML schema enabled XML configuration management tools, dedicated log management tools, internal debugging tools, both dynamic and static module chains support, internal DSO version and consistency checking, well defined API for developing specialized frameworks. It is supported on Scientific Linux 4 and 5 and planned to be ported to other platforms as well. The project is provided with the comprehensive set of technical documentation and users' guides. The licensing schema for the source code, binaries and documentation implies that the product is free for non-commercial use. Although the development phase is not over and many features are to be implemented yet the project is considered ready for public use and creating applications in various fields including development of events reconstruction software for small and moderate scale HEP experiments.

1. Introduction

Hierarchy Software Development Framework ("h-dp-fwk") is a lightweight toolbox for building portable modular applications and performing automated data analysis tasks in a batch mode [1]. The history of design and development activities devoted to the project has begun in March 2005 and from the very beginning the design was targeting the case of building experimental data processing applications for the CMD-3 experiment [2, 3] at VEPP-2000 electron-positron collider [4, 5] which is being commissioned at Budker Institute of Nuclear Physics (BINP, Novosibirsk).

The prototype of the framework (Cmd3Fwk v1.x) with limited functionality is being used in CMD-3 collaboration as a main offline software integration tool since 2005. Nowadays the phase of intensive development of its fully functional version is to be finished in a few months to come so the release of production version 1.0 of h-dp-fwk product accompanied with the complete documentation pack is expected later in 2009. After the release of v1.0 the project will be considered ready for public use and creating user applications in various fields including the development of events reconstruction software for HEP experiments.

2. Framework Architecture

The design of the framework addresses the generic case of modular data processing application operating within the well defined distributed computing environment. The main of its features are:

- Modularity,
- Built-in message and data exchange mechanisms,
- XInclude and XML schema enabled XML configuration management tools,
- Dedicated log management tools,
- Internal debugging tools,
- Both dynamic and static module chains support,
- Internal DSO version and consistency checking,
- SCons [6] build and packaging system for handling both core components and modules,
- Well defined API for creating specialized frameworks.

It is currently supported on Scientific Linux 4 and 5 (both x86 and x86_64 architectures) and planned to be ported to other platforms as well (please refer to the design document for more details).

3. Implementation Status

3.1. Generic Architecture Layout

The framework itself consists of multiple modules organized into layers starting from the external library wrappers up to the layer holding the APIs which are intended to be exploited by the users. In addition to the libraries it provides the command line tools (CLI interfaces) and template applications.

The organization of project components is shown on Fig. 1. Most of the code of the framework components is written in C++, the service components like build system and internal code review are implemented in Bash and Python.

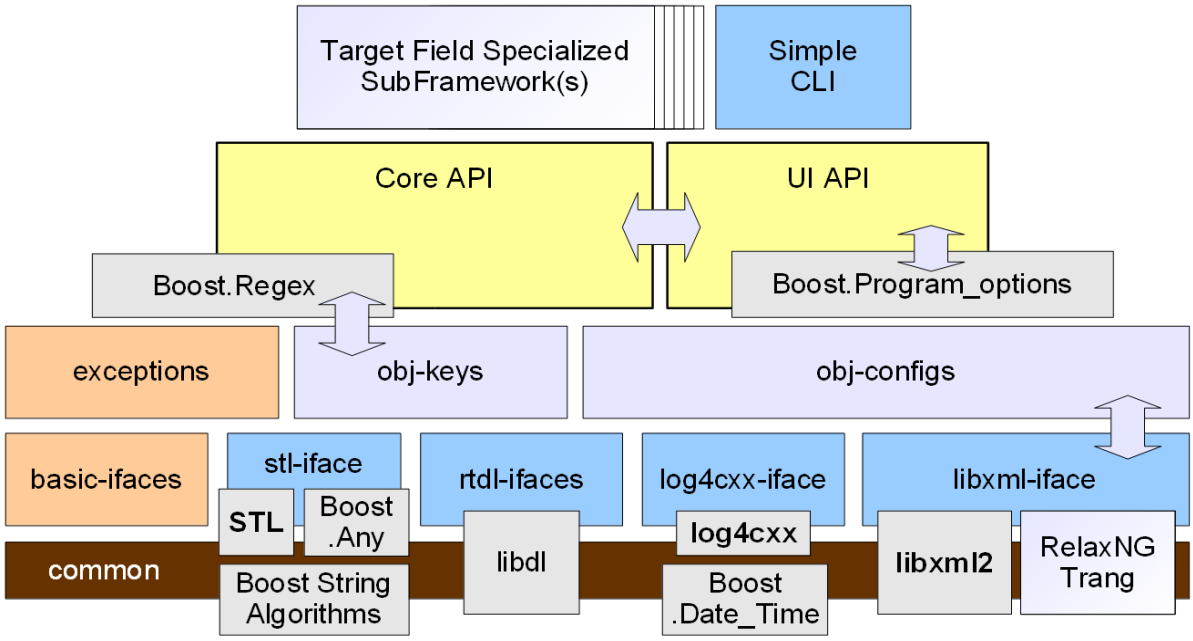

Figure 1. Project architecture layout (the dependent external software packages and the most important dependencies between the layers of hierarchy are also shown). “Target Filed Specialized SubFrameworks” on the top of the structure are to be implemented by the users.

3.2. Build-in Logger Tools

Logging and debugging tools are the most fundamental part of the framework since all the project components depend on it. Centralized logger of the framework is implemented on top of the log4cxx toolbox [7] which is capable of handling named logger hierarchies and provides a variety of message I/O and filtering tools. Use of high level macro wrappers allows one to remove all the logger related code or just a part of it via build system interface without changing and recompiling the code.

The same message handling schema is intended to be used not only by the base framework itself, but by user modules and derived frameworks as well. An overall layout of the built-in logger structure is shown on Fig. 2.

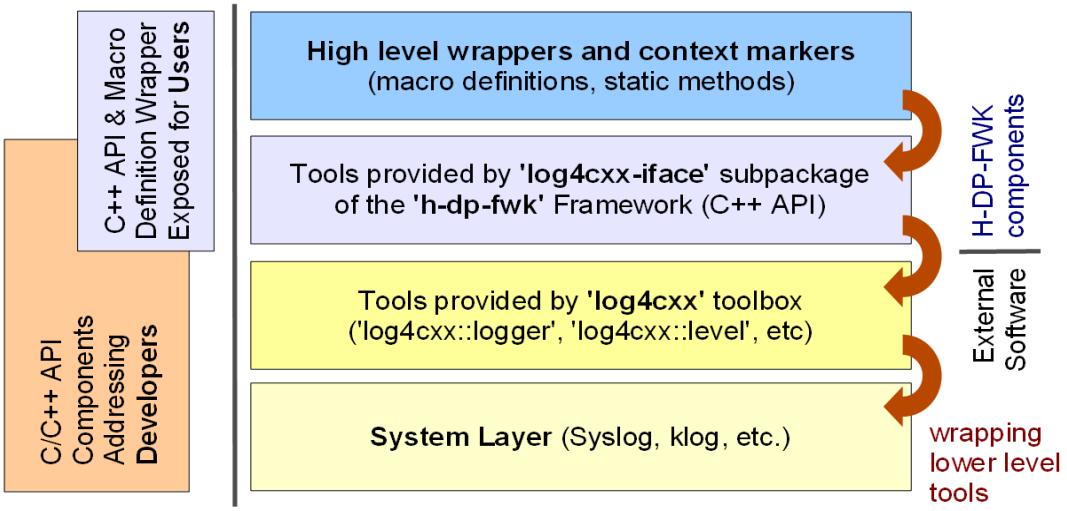

Figure 2. Internal logger layout and the summary of the external tools involved.

3.3. Build System

Build system exploited by the project is based on SCons product which an open source Python based multi-platform build system and high level wrappers placed on top of it. It is aware of the logger tuning options and capable of producing the binaries with requested level of verbosity built-in. In addition it supports the source code tree management tools (cleaning, spellchecking, Doxygen documentation production [8], etc.), batch RPM/tarball builds and provides a simple implementation of the build server functionality.

The framework is released as a set of source and binary RPMs published in public YUM repository plus a source tarball published on the web.

Build system layout it shown on Fig. 3.

3.4. Data Processing Session Organization

Data processing schema is defined by the selected approach to inter-modules interaction while wide spectra of solutions are supported:

- *Hard-coded sequence:* direct module chain resolution, sequence is defined by XML configuration,
- *Directed acyclic graph (DAG) based sequence resolution:* reverse module chain resolution, sequence can be defined by both module's code and XML configuration,
- *Supervisor modules control:* arbitrary module chain resolution defined mostly by the supervisor module's code,
- *Mixture of solutions listed above* exploited simultaneously.

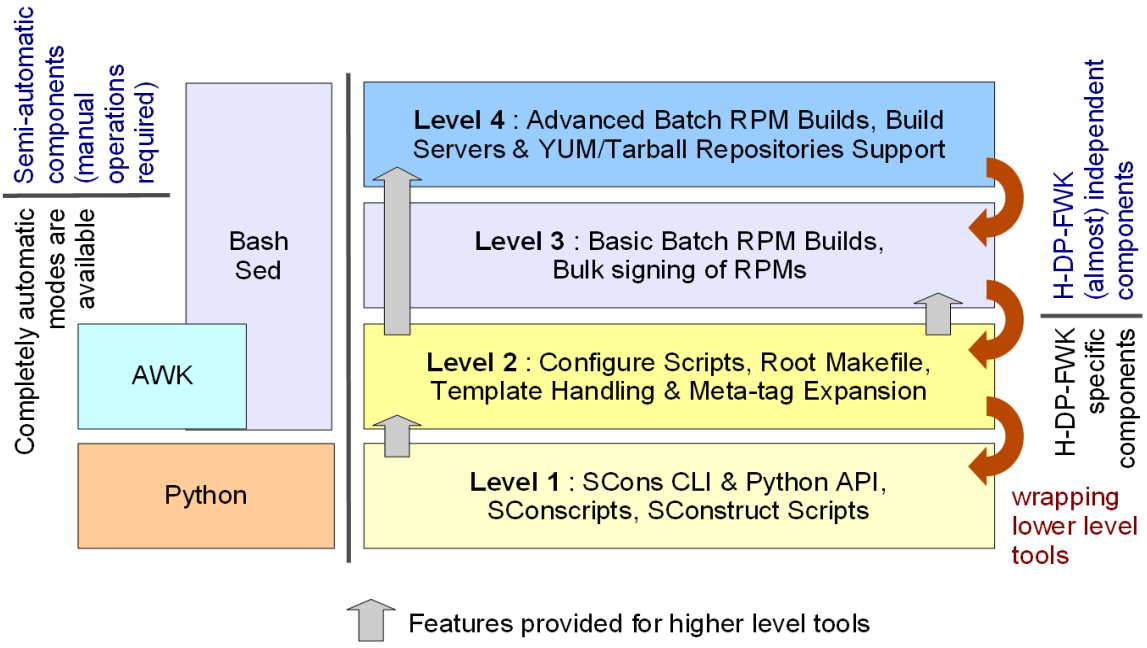

Figure 3. Internal build system layout and the summary of the external tools exploited on the different levels of hierarchy.

3.5. XML Configuration Handling Mechanisms

Individual module configurations and inter-module interaction schema are defined by the externally supplied XML configuration. On the low levels the XML configurations handling is done by libxml2 library [9]. High level wrappers provide XInclude/XML Schema and RelaxNG Schema [10] support and a user tunable XML configuration' sub-sections override mechanism. Combined these two features deliver a significant increase of usability when it comes to dealing with the large configuration, e.g. global reconstruction of experimental events in the particle detector which has multiple sub-detectors.

Generic XML handling is schema is shown on Fig. 4.

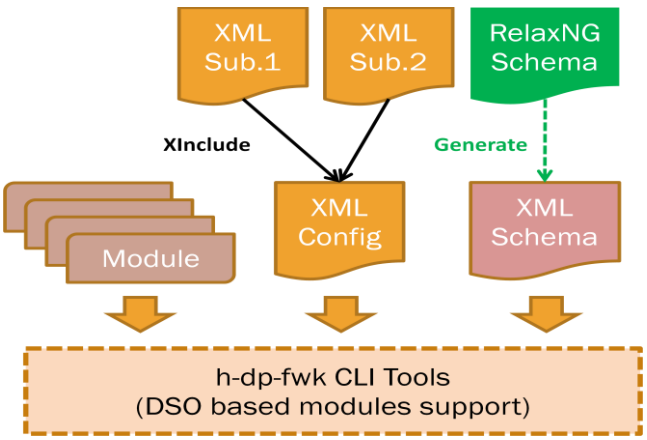

Figure 4. XML configuration handling schema.

3.6. Data Exchange Mechanisms

The framework provides a set of tools for data access management exposed to the end users and module developers:

- Multilayer cache for automatic optimization of the modules.
- Support of the call sequence in the reverse dependency lookup mode (in particular the cache can be tuned to take into account the periodic structures within the input data, e.g. experimental runs and datasets).
- So called "proxy dictionary" mechanism associating a particular data instance in memory with the structured human readable name.

3.7. Remaining Items in TODO List and Prospects for Future Development

The following items which are on the project TODO list were shifted to the next releases since they are not critical for testing the product in the development environment:

- Fully functional support of static builds with predefined set of modules and/or XML configuration for the long term use in production environment,
- GUI tool for editing XML configuration with RelaxNG/XML Schema and interactive dependency tree visualization,
- Standalone interactive tool for reviewing and filtering logger messages,
- Python API.

4. Documentation and Other Resources on the Web

The project is provided with the comprehensive set of technical documentation and users' guides including the dedicated seminar series consisting of 7 dedicated talks of which 3 are already available on the official website [1]. The licensing schema for the source code, binaries and documentation implies that the product is free for non-commercial use.

5. Conclusion

The h-dp-fwk project is now approaching the first publicly available production release v1.0 which will be provided with extensive technical documentation and a collection of application examples. The functionality implemented up to now is enough to build a software integration solution for the offline data processing of a small or moderate scale HEP detector experiment. In addition the possibility of developing the sub-frameworks for the target fields outside the scope of HEP data processing will be investigated later this year.

6. Acknowledgements

The author is grateful to all the members of CMD-3 detector collaboration for fruitful discussions and sustained interest in the project.

7. References

[1] Hierarchy Software Development Framework Official Web Site: http://hdpfwk.org
[2] D.N. Grigoriev, CMD-2 Detector Upgrade. *Preprint* hep-ex/0106009
[3] CMD-3 Collaboration Web Site: http://cmd.inp.nsk.su
[4] I.A. Koop, VEPP-2000 Project. *Preprint* physics/0106013
[5] VEPP-2000 Collaboration Web Site: http://vepp2k.inp.nsk.su
[6] SCons Build System Web Site: http://scons.org
[7] log4cxx Home Page: http://logging.apache.org/log4cxx
[8] Doxygen Web Site: http://www.doxygen.org
[9] libxml2 Library Web Site: http://www.xmlsoft.org
[10] RelaxNG XML Schema Web Site: http://www.relaxng.org