Understanding the Requirements for Developing and Designing Open Source Software

Walt Scacchi
Institute for Software Research
University of California, Irvine
Irvine, CA 92697-3425
Wscacchi@ics.uci.edu

UCI ISR-NSF Workshop on Continuous Design of Open Source Software
23 September 2003

Overview

• Research methodology
• Open source processes for Requirements
• Software development informalisms
• Implications
• Conclusions
Research methodology

• Prior empirical (case) studies of Open Source Software Development (OSSD) Projects
  – Reis and Fortes, 2002, Mozilla Web browser
  – Schach et al., 2002; Holt et al., 2000, Linux Kernel
  – Koch and Schneider 2001; German 2002, GNOME User Interface
  – Jorgensen, 2001, FreeBSD operating system
  – Garg et al., 2002, OSSD (“progressive open source”) within HP
Research methodology

• Individual case studies: significant details, but limited (and premature) generalization, little/no comparative analysis

• No studies that examine multiple OSSD projects in multiple domains
  – Such studies would offer higher degree of comparative analyses and generalization of results
Research methodology

• Comparative case studies
  – Multiple open software development projects
    • Across multiple communities
• Qualitative ("grounded theory") techniques
• Analyzing and modeling
  – development processes
  – work practices and roles
  – development artifacts and tools
  – community structures and process dynamics
OSS processes for Requirements

• *Post-hoc* assertion of requirements+design after implementation
• Reading, sense-making, accountability
• Continually emerging webs of discourse
• Condensing and hardening discourse
• Global access to this discourse
OSS processes for Requirements/Design

• OSS Requirements/Designs are
  – not explicit
  – not formal

• OSS Requirements/Designs are embedded within “informalisms”

• Example OSS informalisms follow
Benefits of Qt3?
by Matt Perry on Friday July 27, @09:22AM

What are the benefits of moving to Qt3?

[Reply To This | View ]

* Re: Benefits of Qt3?
  by Justin on Friday July 27, @09:41AM
  - Support for Arbor and Hewtow
  - RichText classes
  - Database support
  - Component model
  - No more cutpaste problems (but only between Qt3 apps)

One of the most complained about aspects of X is the data clipboard, so getting KDE based on Qt3 will solve a lot of headaches. But this is from a user perspective.

From a developer perspective, KDE-DB is going to utilize Qt3's database support, and this can't happen until they make the switch. KWord currently uses a backported toolkit for use with Qt2. So you can see that there is a drive/need in KDE to use the new Qt3 features.

[Reply To This | View ]

* Re: Benefits of Qt3?
  by Meike on Friday July 27, @12:04PM

What is the purpose of database support in a "widget toolkit"? Isn't this just like placing TUI/IE support in /etc/passwd or another similarly unrelated place?

[Reply To This | View ]

* Re: Benefits of Qt3?
  by Aaron J. Seigo on Friday July 27, @12:36PM

There is often a need to access data from a database and display it in a GUI or vice versa. In those cases having a db API that abstracts the details of the actual data access away (connecting, sending queries, retrieving results, details specific to a given db implementation, etc) that works nicely with your widgets (even so far as to make the widgets aware of the database) is very very nice.

Making such things simple and convenient opens the door to making more applications database aware (e.g. financial packages, email apps, contact information systems).
Ideas

Here are a few ideas of tools, classes and projects that you could start. More are forthcoming.

Runtime

We need a verifier that can run on an executable (assembly) and tells whether the metadata for the executable is correct or not. It should report any anomalies.

For a list of anomalies in assemblies, check the various assertions that are described on the ECMA documentation.

This will help test our generated executables and can also be used as an external verifier.

Classes

TODO-jxta: The JXTA Peer to Peer foundation
  • Implement a JXTA protocol implementation: http://www.jxta.org

TODO-mail: Mail API
  • Implement a Mail API, similar to Camel or JavaMail (Camel has significant architecture features that are required on a real mailer).

You can check the current Cam implementation.

Such an implementation could be used both with Microsoft .NET and Mono.

TODO-multimedia
  • Interfacing to Multimedia systems. You might want to look into the Quicktime API. I know Vladimir has researched the problem before.

TODO-gtk: GTK+ wrappers for Mono and .NET
  • Wrap the GTK+ API. This is simple and can be done on Windows as GTK+ 2.0 works on Windows.

This work can also be done on Windows and will enable developers on Windows to use some of GTK+’s advanced features.

The idea is to wrap the GTK+ API and allow us to build GUI applications using GTK+ and in the future other GTK+ based interfaces from Mono (GTK, Gnome).

There is extensive knowledge on wrapping GTK+ in other languages (GTK has been done this for Perl, Python, Java, Scheme, Haskell and other languages in the past).

Projects

• Implement an xmlStorageSystem for the CLI: http://www.soapware.org/xmlStorageSystem

TODO-juava: Java compiler for .NET
• You could take one of the existing Java compilers (Kava comes to mind as it is so nice) and modify it to generate .NET code rather than JVM byte codes.

This should be a pretty straightforward task. Kava has the advantage of being written in C++ and it could be compiled with the Microsoft managed C++ compiler and produce a .NET executable with it.
The greatest thing about the Unreal Engine Technology which Unreal Tournament uses, is that you can actually create your own levels, mods, skins, models, and more; making for an endless amount of new things to play with, and new things to see. Heck, people even get hired for doing these things, so give it a shot, you just might like it.

Below are links to various sections which give some detail on where to learn about editing, as well as a few tutorials and downloads to help get you started. Good luck!

- Editing Resources
- Editing Utilities
- Console Commands
- UnrealED Quick Reference
The Chandra Automatic Data Processing Infrastructure

Davidflummer and Sreelatha Subramanian
Harvard-Smithsonian Center for Astrophysics, 60 Garden St. MS-81, Cambridge, MA 02138

Abstract:
The requirements for processing Chandra telemetry are very involved and complex. To maximize efficiency, the infrastructure for processing telemetry has been automated such that all stages of processing will be initiated without operator intervention once a telemetry file is sent to the processing input directory. To maximize flexibility, the processing infrastructure is configured via an ASCII registry. This paper discusses the major components of the Automatic Processing infrastructure including our use of the STScI OPUS system. It describes how the registry is used to control and coordinate the automatic processing.

1. Introduction

Chandra data are processed, archived, and distributed by the Chandra X-ray Center (CXC). Standard Data Processing is accomplished by dozens of “pipelines” designed to process specific instrument data and/or generate a particular data product. Pipelines are organized into levels and generally require as input the output products from earlier levels. Some pipelines process data by observation while others process according to a set time interval or other criteria. Thus, the processing requirements and pipeline data dependencies are very complex. This complexity is captured in an ASCII processing registry which contains information about every data product and pipeline. The Automatic Processing system (AP) polls its input directories for raw telemetry and ephemeral data, pre-processes the telemetry, kicks off the processing pipelines at the appropriate times, provides the required input, and archives the output data products.

2. CXC Pipelines

A CXC pipeline is defined by an ASCII profile template that contains a list of tools to run and the associated run-time parameters (e.g., input/output directory, root-names, etc.). When a pipeline is ready to run, a pipeline run-time profile is generated by the profile builder tool, pbuilder. The run-time profile is executed by the Pipeline Controller, pctr. The pipeline profiles and pctr support conditional execution of tools, branching and converging of threads, and log file output containing the profile, list of run-time tools, arguments, exit status, parameter files, and run-time output. This process is summarized in Figure 1.

Figure 1: The CXC Pipeline Processing Mechanism
4. Standard Pipeline Processing Threads

Figure 2 represents the series of pipelines that are run to process the Chandra data. Each circle represents a different pipeline (or related set of pipelines). Level 0 processing (de-commutation) will produce several data products that correspond to the different spacecraft components. Data from the various components of the spacecraft will follow different threads through the system. The arrows represent the flow of data as the output products of one pipeline are used as inputs to a pipe (or pipes) at the next level. Some pipelines are run on arbitrary time boundaries (as data are available) and others must be run on time boundaries based on observation intervals start and stop times (which are determined in the level 0 pipe, OBE_DET).

Figure 2: Standard Processing Threads.

5. Pipeline Processing Registry

The complete pipeline processing requirements for Chandra are very complex with many inter-dependencies (as can be seen in Figure 2). In order to run the pipelines efficiently in a flexible and automated fashion we configure the Automatic Processing system with a pipeline processing registry. We first register all the Chandra input and output data products. We can then capture the processing requirements and inter-dependencies by registering all the pipelines. Data products are registered with a File_ID, file name convention, file name conversion (using regular expressions), method for extracting start/stop times, and archive ingest keywords (detector, level, etc.). Pipelines are registered with a Pipe_ID, pipeline profile name, shoulder arguments, kickoff criteria (detector, focal plane, gratings info, etc.), input and output data products (by File_ID), and method for generating the “root” part of output file names.
Traditional vs. OSS processes for Requirements

- Elicitation
- Analysis
- Specification and modeling
- Validation
- Communicating and managing

- *Post-hoc* assertion
- Reading, sense-making, accountability
- Continually emerging webs of discourse
- Condensing and hardening discourse
- Global access to discourse
Software Informalisms

• Community communications
  – Threaded discussion forums
  – Email (list servers)
  – Newsgroups
  – IRChat/Instant messages
  – Community digests ("Kernel Cousins")
Software Informalisms

• *Scenarios of Usage* as linked Web pages
Software Informalisms

• How-To guides, To-Do lists, FAQs
• Traditional software user documentation
  – Unix/Linux man pages
• External publications
  – trade articles
  – scholarly research papers
  – books (cf. O’Reilly Books)
Software Informalisms

• Open Software Web Sites
  – Community Web sites
  – Community Software Web sites
  – Project Web sites
  – Source code Webs/Directories
PHP-Nuke 7.0 ALPHA 1 Released

The very first version 7.0 has been released, for Club Members only, which includes a new points system, modules access and users groups by points. Also includes some security fixes and the latest phpBB forums port version 2.0.6. The new points system and users groups will be improved in coming releases. Enjoy!

Enter the club and get it!

Addons: RGB to HEX Converter module

aronco writes “This is a simple module that will convert RGB color codes to hex and hex codes to RGB. A cool tool for use in designing Web pages.

It can be reviewed here, and downloaded here.”

Posted by aronco on Sunday, September 21 @ 15:15:31 EDT (77 reads)
(Read More… | Addon | Score: 0)

The first book on PHP-Nuke - hot of the press - please add your language now

Anonymous writes “Brandnew - The first book on PHP-Nuke soon gets hot of the press..., the first book on phpnuke is so hot that the publishers covered it red - see here for full details and please see some additional info on phpnuke-book.com.”

Posted by Anonymous on Sunday, September 21 @ 15:11:03 EDT (156 reads)
(Read More… | 2722 bytes more | Score: 0)

PNUkeNews 1.8 Released

Anonymous writes “PNUkeNews version 1.8 has been released. PNUkeNews is a stand-alone Windows client that allows PHP-Nuke site operators design their own news client for their users and distribute it royalty free after registration.

It includes realtime notification on new news, forum postings or downloads. Installation is configured by the user to either a selected web file, flash icon in the tray, or both. It also includes autopaste features and skin themes. And last, but not least, print support.”

Posted by Anonymous on Sunday, September 21 @ 17:01:03 EDT (150 reads)
(Read More… | 5762 bytes more | Score: 0)
eGroupWare roadmap for the 1.0 release

Roadmap last time changed: Fri Sep 19

API

- LDAP-support bugfixes

Calendar

- day-view: show user-timed and untimed (todo's) events
- enabling contacts to be included in meetings (with editable email-notification)

Email

- some bugfixes
- support for email admin
- include preferences

Email admin

- finish programming of email admin
- support for postfix-ldap
- add support to manage folder acl on the imap server

Felanimail

- fix-parsing of message/rfc822-mime-type
- some bugfixes
- some-changes at the user-interface

Forum
Software Informalisms

• Software bug reports
  – Ad hoc report Web
  – Bugzilla (database tracking)

• Issue tracking
  – Issuezilla
Bug: dmextract

- Bug: Crashes when extracting a PHA, while using a stack of files for background extraction [6155]

- Bug (fixed): A memory corruption causes the tool to crash when creating a long HISTORY string [6205]

- Caveat: BACKSCALE areas for complicated regions may have small (of order 1-2 percent) errors, as they are calculated using an approximate algorithm. CIAO 2.2 is improved relative to CIAO 2.1, but still not perfect.

Last modified: 14 November 2001
Software Informalisms

- Software extension mechanisms
  - Inter-application scripting
    - Csh, Perl, Python, Tcl, scripting
    - Pipelines (cf. CXCDS)
  - Intra-application scripting (e.g., *UnrealScript*)
  - Plug-in architectures
    - Apache server architecture
Software Informalisms

• Free/OSS licenses – institutionalizing F/OSS culture (values, norms, and beliefs)
  – GNU Public License (GPL)
  – and 35 more (http://opensource.org)
  – “Creative Commons” Project at Stanford Law School developing public license framework
Other Licenses: Frequently Asked Questions
Version 1.0 - February 9, 2001

Q: Are there other licenses that meet the definition of an Open Game?

A: Yes, there are several.

Q: How about the GNU licenses?

A: The General Public License (GPL), the Lesser General Public License (LGPL), and the GNU Free Documentation License (GFDL) all provide terms that could be used to publish an Open Game.

Q: There are lots of Open Source software licenses. Can those be used to create Open Games?

A: In general, if a license meets the Open Source Definition, it will almost certainly provide the tools to distribute an Open Game as well.

Q: Why not use those licenses then?

A: The biggest impediment to using the Open Source licenses is that most of them do not provide for a separation between game rules and trademarks, setting content, fiction, illustrations, and maps. The Open Gaming License does this through the use of the Product Identity clause, and by not requiring that everything in a given work be Open Game Content.

Q: How about the Dominion Rules License?

A: The DRL provides terms that can be used to publish an Open Game.

Q: What about the October Open Game License?

A: The October Open Game License provides terms that can be used to publish an Open Game.

Q: Why not use one of those licenses then?

A: The DRL is designed to support the development of the Dominion Rules game system. While it is fully capable of being used for a non-affiliated game system, the terms of the license will leave bits and pieces of the Dominion Rules copyright notices and licensing requirements behind. It is simply not designed to be used as a generic Open Game license.
Implications

- Software informalisms are the *media* of software requirements/design
- Software informalisms are the *subject* of software requirements/design
- OSS requirements/design tasks are *implied activities or capabilities*
- *(Re)reading, reviewing, and reinterpreting informalisms is a prerequisite to writing OSS.*
Implications

• Developing open software requirements is a *community building process*
  – not just a technical development process
  – OSS peer review creates a *community of peers*

• OSSD processes often iterate *daily* versus infrequent singular (milestone) SLC events
  – frequent, rapid cycle time (easier to improve) *vs.* infrequent, slow cycle time (hard to improve)
Implications

• Determining the quality of OSS requirements/designs:
  – *not* targeted to consistency, completeness, correctness
  – *instead* focusing attention to community building, freedom of expression, ease of informalism navigation (traceability), implicit vs. explicit informalism structuring
Conclusions

• Developing OSS requirements is *different* than requirements engineering
  – not better, not worse, but different and new
  – more social, more accessible, more convivial

• OSS systems don’t need and probably won’t benefit from classic software requirements engineering.
Acknowledgements

• **Project collaborators:**
  – Mark Ackerman, UMichigan, Ann Arbor
  – Les Gasser, UIUC
  – Margaret Ellliot, Chris Jensen, UCI-ISR
  – Julia Watson, The Ohio State University

• **Funding support:**
  – National Science Foundation, IIS#-0083075, ITR#-0205679, ITR#-0205724, and IIS#-0350754.
References


