17-708 SOFTWARE PRODUCT LINES: CONCEPTS AND IMPLEMENTATION

IMPLEMENTATION 1: VERSION CONTROL SYSTEMS PARAMETERS

CHRISTIAN KAESTNER
CARNEGIE MELLON UNIVERSITY
INSTITUTE OF SOFTWARE RESEARCH

Chapter 4
LEARNING GOALS

Explain the difference between versions, revisions and variants

Understanding the appeal and the problems of clone-based development

Use version control systems to implement and track minor variations
GRAPH PL

GraphLibrary

Node
  - Color
  - Position

Edges
  - Directed
  - Weighted

Search
  - BFS
  - DFS

Algorithm
  - Cycle
  - ShortestPath
  - MST
  - Transpose
class Graph {
    Vector nv = new Vector(); Vector ev = new Vector();
    Edge add(Node n, Node m) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = new Weight();
        return e;
    }
    Edge add(Node n, Node m, Weight w)
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = w; return e;
    }
    void print() {
        for(int i = 0; i < ev.size(); i++) {
            ((Edge)ev.get(i)).print();
        }
    }
}

class Node {
    int id = 0;
    Color color = new Color();
    void print() {
        Color.setDisplayColor(color);
        System.out.print(id);
    }
}

class Edge {
    Node a, b;
    Color color = new Color();
    Weight weight = new Weight();
    Edge(Node _a, Node _b) { a = _a; b = _b; }
    void print() {
        Color.setDisplayColor(color);
        a.print(); b.print();
        weight.print();
    }
}

class Color {
    static void setDisplayColor(Color c) { ... }
}

class Weight { void print() { ... } }
VERSION

CONTROL

SYSTEMS
BRANCHING & Merging

Trunks

Branches

Merges

Tags

Discontinued development branch
SOFTW. CONFIGURATION MANAGEMENT

"tracking and controlling change in software dev."
revision control
build consistency and reproducability
process enforcement
reporting and auditing
defect tracking
release planning
<table>
<thead>
<tr>
<th>Variants</th>
<th>V 1.0</th>
<th>V 1.1</th>
<th>V 2.0</th>
<th>V 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor DB (Car)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Sensor DB (Habitat Monitoring)</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Sensor DB (Earthquake Monitoring)</td>
<td></td>
<td></td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>SmartCard DB</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Satnav DB</td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
</tbody>
</table>
PRODUCT LINES WITH VERSION CONTROL

[Staples&Hill, APSEC’04]
FEATURE-ORIENTATION?
UNIFORMITY

Java Code
Documentation
Models
Build scripts
License
Grammars
Compiled binaries
HTML, JavaScript, CSS
(binary files are difficult to merge)
APPEAL OF
CLONE-AND-OWN
PROBLEMS OF CLONE-AND-OWN
<table>
<thead>
<tr>
<th>Statement</th>
<th>Not at all</th>
<th></th>
<th>To a great extent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I’m aware of a product line strategy in my organization</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2. My team develops core assets that are later used by other teams</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>6. We regularly clone pieces of code</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>10. We measure how many times a certain core asset is used</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>17. People in my team know who should approve a change request</td>
<td></td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>19. We have work procedures that include cloning of artifacts</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>20. I feel our development process is well-defined</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>21. We clone to reuse artifacts between products of a product line</td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>22. All project teams follow in practice the defined process</td>
<td>1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>23. We have relatively many clones of code</td>
<td>6</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>27. We do not change main APIs without receiving an approval</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>31. We have relatively big clones of code</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>
Stanciulescu, Schulze, & Wasowski, Forked and Integrated Variants in an Open-Source Firmware Project, ICSME 2015
Observation 6. We have observed that Marlin developers preferred forking over integrated variability under the following reasons:

S1. The fork extension has little relevance to other users.
S2. The maintenance time span for the developed code is expected to be short.
S3. The external developer has no control over the upstream project.
S4. A developer wants to create experimental code.
S5. An active project provides a good skeleton for adding new functionality.
S6. A defunct project contains code that can be reused.
S7. The developer wants to change the programming language.

Observation 4. Decentralization of information in many forks is a challenge in fork-intensive development.

Observation 5. Propagation of bug-fixes is a problem for forking, just like for cloning in-the-small, even though git offers facilities for selective download of patches from upstream.

DISCUSSION

Modularity?
Traceability?
Effort?
Granularity?
Uniformity?
LOAD-TIME VARIABILITY
PARAMETER


[drive:][path][filename]
  Specifies drive, directory, and/or files to list.

  /A
  Displays files with specified attributes.

  attributes
  D Directories
  H Hidden files
  S System files
  L Reparse Points

  /B
  Uses bare format (no heading information or summary).

  /C
  Display the thousand separator in file sizes. This is the default. Use /-C to disable display of separator.

  /D
  Same as wide but files are list sorted by column.

  /L
  Uses lowercase.

  /N
  New long list format where filenames are on the far right.

  /O
  List by files in sorted order.

  sortorder
  N By name (alphabetic)
  E By extension (alphabetic)
  G Group directories first

  /P
  Pauses after each screenful of information.
PARAMETER –I IN GREP

```c
int match_icase;

int main (int argc, char **argv)
{
    [...] 
    while ((opt = get_nondigit_option (argc, argv, &default_c, 
        switch (opt)
            {
                [...] 
                case 'i':
                    match_icase = 1;
                    break;
                
            }
    }
}

static const char *
print_line_middle (const char *beg, const char *lim,
    const char *line_color, const char *match_color)
{
    [...] 
    if (match_icase)
    {
        ibeg = buf = (char *) xmalloc(i);
        while (--i >= 0)
            buf[i] = tolower(beg[i]);
    }
```
class Conf {
    public static boolean Logging = false;
    public static boolean Windows = false;
    public static boolean Linux = true;
}

class Main {
    public void foo() {
        if (Conf.Logging)
            log(“running foo()”);
        if (Conf.Windows)
            callWindowsMethod();
        else if (Conf.Linux)
            callLinuxMethod();
        else
            throw RuntimeException();
    }
}
class Graph {
    Vector nv = new Vector(); Vector ev = new Vector();
    Edge add(Node n, Node m) {
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        if (Conf.WEIGHTED) e.weight = new Weight();
        return e;
    }
    Edge add(Node n, Node m, Weight w) {
        if (!Conf.WEIGHTED) throw RuntimeException();
        Edge e = new Edge(n, m);
        nv.add(n); nv.add(m); ev.add(e);
        e.weight = w; return e;
    }
    void print() {
        for (int i = 0; i < ev.size(); i++) {
            ((Edge)ev.get(i)).print();
        }
    }
}

class Node {
    int id = 0;
    Color color = new Color();
    void print() {
        if (Conf.COLORED) Color.setDisplayColor(color);
        System.out.print(id);
    }
}

class Color {
    static void setDisplayColor(Color c) { ... }
}

class Weight {
    void print() { ... }
}

class Edge {
    Node a, b;
    Color color = new Color();
    void print() {
        if (Conf.COLORED) Color.setDisplayColor(color);
        System.out.print(id);
    }
}

class Conf {
    public static boolean COLORED = true;
    public static boolean WEIGHTED = false;
}
PROPAGATING PARAMETERS

replacing global variables by parameters

```java
public Sequence openSequence(Transaction txn,
                               DatabaseEntry key,
                               SequenceConfig config)
    throws DatabaseException {

checkEnv();
DatabaseUtil.checkForNulDb(key, "key", true);
checkRequiredDbState(OPEN, "Can't call Database.openSequence:");
checkWritable("openSequence");
trace(Level.FINEST, "Database.openSequence", txn, key, null, null);

    return new Sequence(this, txn, key, config);
}

/**
 * Javadoc for this public method is generated via
 * the doc templates in the doc_src directory.
 */
public void removeSequence(Transaction txn, DatabaseEntry key)
    throws DatabaseException {

    delete(txn, key);
}
```
CONFIGURATION

httpd.conf -- win32 Apache
Building a Web Server, for Windows

Listen 80
ServerRoot "/www/Apache2"
DocumentRoot "\www/webroot"

ServerName localhost:80
ServerAdmin admin@localhost

ServerSignature On
ServerTokens Full

DefaultType text/plain
AddDefaultCharset ISO-8859-1

UseCanonicalName Off
HostnameLookups Off

ErrorLog logs/error.log
LogLevel error

PidFile logs/httpd.pid
Timeout 300

KeepAlive On
MaxKeepAliveRequests 100

Command line parameter
Config files
Dialogs
Source code

Preferences...
DISCUSSION
TRADEOFFS

Modularity?
Traceability?
Effort?
Granularity?
Uniformity?
FURTHER READING

