

DALHOUSIE UNIVERSITY

TOPICS IN PROGRAM COMPREHENSION

CSCI 6306

---

# Architecture Extraction

---

*Group : Something*

Bhupendra RAJAWAT

Bryan Thomas D'SILVA

Saurabh SINGH

June 12, 2017



DALHOUSIE  
UNIVERSITY

Note: Our fourth group member, Anto Anthony Arulappan had an emergency in the family and had to leave to India for the same. He has dropped the course and hence will no longer be a part of our group.

# Contents

<b>1</b>	<b>Finding a Software System</b>	<b>3</b>
<b>2</b>	<b>About GIMP</b>	<b>3</b>
<b>3</b>	<b>Lines of Code</b>	<b>4</b>
3.1	Count Lines of Code . . . . .	4
3.2	LocMetrics . . . . .	5
<b>4</b>	<b>Architecture Extraction Tool</b>	<b>5</b>
<b>5</b>	<b>Directory Structure</b>	<b>5</b>
<b>6</b>	<b>Entry Point</b>	<b>6</b>
<b>7</b>	<b>Build Process</b>	<b>6</b>
<b>8</b>	<b>Control Flow</b>	<b>9</b>
<b>9</b>	<b>Dependencies</b>	<b>9</b>
<b>10</b>	<b>Algorithms</b>	<b>9</b>
<b>11</b>	<b>Conclusion</b>	<b>10</b>

## Abstract

Your abstract.

# 1 Finding a Software System

We looked at the following software systems and games.

1. **Kodi**: Kodi is an open source software media player and entertainment hub. This is available on atleast seven platforms. This was one of the first softwares we looked at and kept it as a contender
2. **Quake III**: The source code for Quake is challenging and interesting. However, there is almost no documentation for the same and hence we decided not to go ahead with this for now.
3. **GIMP**: GIMP is a cross platform image editor with an excellent community backing it up. We decided to extract the architecture for GIMP. We talk more about GIMP in the further sections.

We also looked at Doom3, SuperTuxKart, Wordpress, TuxRunner and Linux but found GIMP to be the most interesting one to use for this assignment

# 2 About GIMP

GIMP was designed initially as a General image manipulation program. This was later changed to GNU Image Manipulation Program. It is free and openly available to retouch and edit images, convert image formats and more specialized tasks targeted towards image manipulation. GIMP provides an interface to handle simple graphics needs without learning advanced methods for image manipulation. GIMP is more user friendly as compared to other photoshop techniques and can be modified to fit your needs. And the best part about it is that it is free.

```

5102 unique files.
1859 files ignored.
[
github.com/AlDanial/cloc v 1.72 T=42.12 s (81.5 files/s, 87718.6 lines/s)

```

Language	files	blank	comment	code
P0 File	446	567383	750322	1378612
C	1409	142802	85749	608029
C/C++ Header	1229	17629	25137	61853
make	153	1698	512	12006
HTML	7	8	0	11229
Racket	55	1077	1443	5946
Python	27	974	832	3739
Perl	12	596	216	3680
m4	8	580	36	3108
Windows Module Definition	10	14	6	2970
Qt	5	0	13	1911
XML	37	75	15	1424
yacc	3	114	73	588
Bourne Shell	7	84	59	422
C++	1	70	25	387
JSON	1	0	0	317
XSLT	4	53	12	207
lex	3	91	63	188
Bourne Again Shell	4	37	1	115
Windows Resource File	1	33	45	111
DTD	4	14	4	57
Markdown	1	16	0	39
DOS Batch	1	4	0	32
INI	1	0	0	29
Lua	2	18	45	24
vim script	1	4	12	12
diff	2	3	11	11
CMake	1	1	0	4
SUM:	3435	733378	864631	2097050

Figure 1: CLOC output for GIMP

## 3 Lines of Code

### 3.1 Count Lines of Code

Count Lines of Code[2], abbreviated as CLOC is a software used to count comment lines, blank lines and physical lines of code. We used this software initially when we looked at what software system we need to pick. We can look at figure 1 for the output for GIMP. To sum it we have 3,695,059 LOC. However, we would assume that the chunk of our work would be for C which sums to about 608,029 LOC.

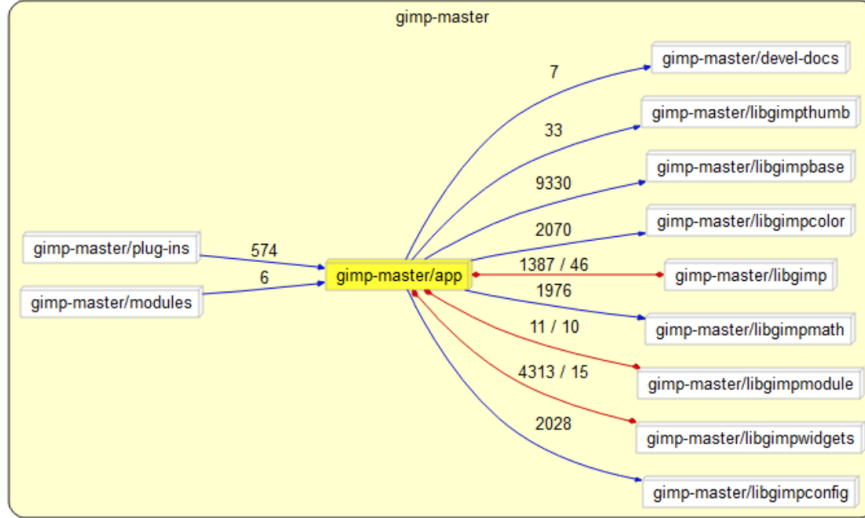


Figure 2: Directory Structure

### 3.2 LocMetrics

LocMetrics[1] is very similar to what CLOC does. We did find this software interesting because of how it visualized the output.

## 4 Architecture Extraction Tool

## 5 Directory Structure

We generated a butterfly diagram in figure 2 for the directory structure. Using this and the documentation we did want to identify the main folder we would want to have a look at. We concluded that the app directory is the main directory that we are concerned about. It contains the source code for GIMP without any external tools or plugins. We would ideally ignore the modules, plugins and libraries for now and focus on the app directory. So we did the same for the app folder in figure 3. It was easy to comprehend since GIMP is very well organized with a good community support and documentation. The app/core directory is the guts of the GIMP software. Looking at the

documentation provided for GIMP, major changes that need to be made start here. With the goal of refactoring in mind, this is where we would start, keeping in mind that changing files here will have an avalanche effect, so we need to be prepared for big changes in other files depending on these.

## 6 Entry Point

While trying to build the application, we looked at MakeFile.am. The following line in MakeFile.am which gave us an idea about the entry point into the program.

**`gimp_@GIMP_APP_VERSION@_SOURCES = $(libapp_sources) main.c`**

Understand C++ has an option to generate a callby graph. Running this on main.c, instantly we can guess that main is not called by anything which further reinforces the idea that main is the entry point into the program. The call by graph for main.c is shown in figure 4

So looking into the main.c to further reinforce our idea.

The main.c has a main function which points to functions that talk about the installation directory. We did gain a lot of information from the Understand C++ software.

Failures: We tried to use Doxygen to generate call graphs for comparison and better understanding. However, the tool crashed on the first go. And on the second go generated quite a bunch of call graphs when run directly on the app directory so will leave this tool aside for now.

## 7 Build Process

Analysis of any application can not be completed without building it. Like every application provide documentation related to how one can build the application on any platform eg Linux, Mac etc. Building any application not only help to compile any application to create executable but end up to helping to figure out what are the entry point with in the code of application. Like in case of Gimp they have "MakeFile.am" which is a autogenerator file with in "master/app" directory to create "Makefile.in" . If you analyze Makefile.am you will get that "main.c" is the file from were compilation starts.

We tried to use Doxygen to generate call graphs for comparison and

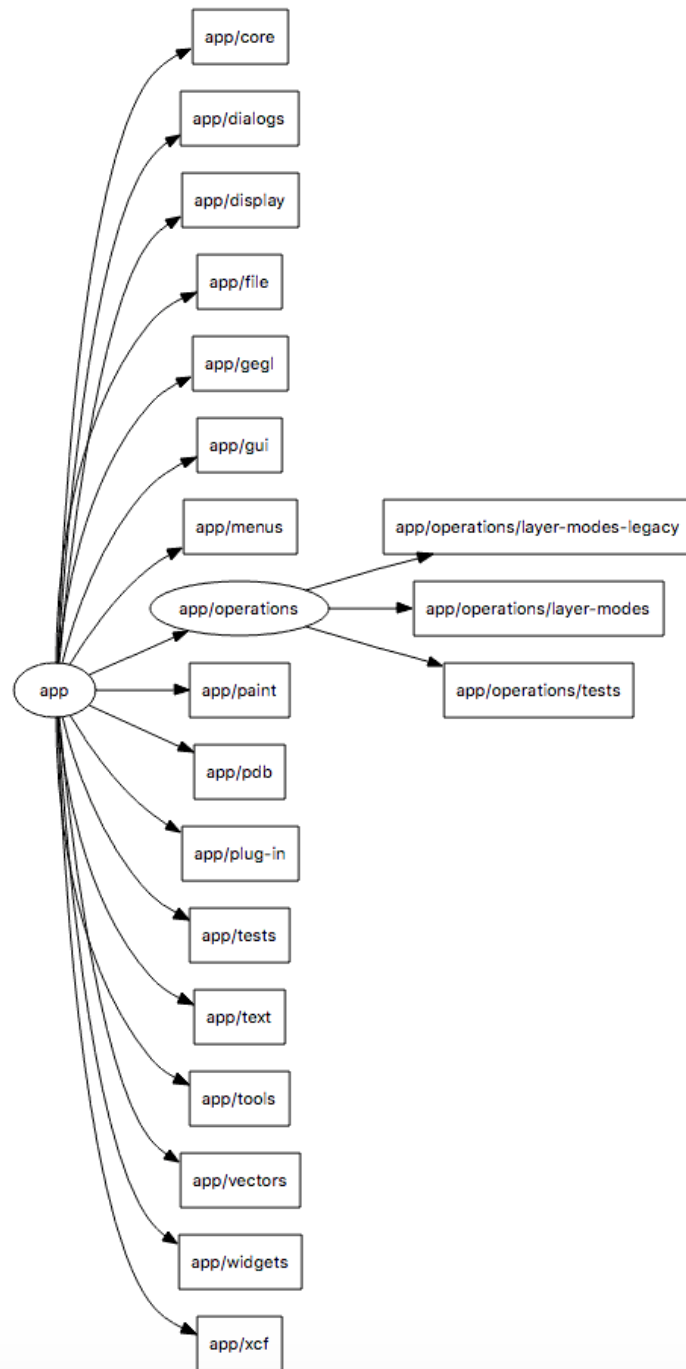


Figure 3: Directory Structure for app folder



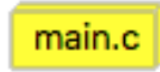


Figure 4: Callby graph for main.c

better understanding. However, the tool crashed on the first go. And on the second go generated quite a bunch of call graphs when run directly on the master/app directory so will leave this tool aside for now.

Talking about Dependencies required for the Gimp to get build you can check with in GIMP master folder you will find the INSTAL.in file which will tell how to build the application on particular platform with dependencies required. We are using Linux environment to build the application.

To build this process we have two different type of build system provided one through a Tarball and other building from latest source (Git). We used the latter one to build the application.

Being a product of GNOME ,gimp use many of their pre existing libraries which they utilize for their other products as well like GLib , GObject ,GTK+, Cairo, Pango.

The two major libraries which gimp uses are GEGL and babl which one need to install before building this application. And both can be installed either using tarball or git on system before building. Links for both the libraries are present on wiki page of the application.[3].

Once this is done we are require to just run autogen.sh which is a auto maker for configuration file. Once configuration file is generated we can run it to auto generate makefile. Also this is where you get to know if you have installed all libraries or not. This is the place where we got stuck as I already had a previous gimp installation and did not change the path for the new installation and end up having conflict between both of the installation which got resolved by just changing the default path. Once make file is generated you can run the “make” command and then “make install” to complete the build process.

So if we see the major flow of the build after installing all libraries is as follows:

**autogen.sh—config—make—make install**

Operation name	PDB operation name	GUI menu item	Implementation file	Description
gimp:flood	gimp-selection-flood	Select -> Remove holes	<a href="#">gimpoperationflood.c</a>	<a href="#">Flood algorithm</a>

Figure 5: Documentation on operation algorithms

## 8 Control Flow

## 9 Dependencies

Gimp has a dependency on many libraries some important ones are as follows:

1. **GLib:**

A common library which is basically used in most of the gnome applications, it contains the various utilities and data structures needed by the program in c.

2. **GEGL:**

It is a graph based image processing library which is used by GIMP in order to support high bit depth images.

3. **babl:**

It is a support library for GEGL. It is helpful in color-space conversions.

4. **PDBTypes.h:**

It is helpful in defining enums for various fields contained in PDB.

5. **GDK:**

It helps by acting as a wrapper around low level functions.

6. **GObject:**

It is a library which is used to implement C objects.

## 10 Algorithms

To find the underlying algorithms behind the operations performed on GIMP, the wiki for the same had a link to the algorithms used. We expected to find all the underlying algorithms here but encountered a problem. There was only one operation described here shown in figure 5.

Since this is implemented in the `gimpoperationflood.c`, the best way to find the algorithms for the other operations performed would be to have a look at where this file is located.

The `app/operations` folder is where this file located. Looking at the other files in the folder, we got an idea of where to look incase we had to change or add any further operations. Since these operations are triggered using the GUI, changing something here would cause us to look at the UI elements to be updated as well.

## 11 Conclusion

## References

- [1] <http://www.locmetrics.com/>
- [2] <https://github.com/AlDanial/cloc>
- [3] <https://wiki.gimp.org/>