



**Free/Libre and Open
Source Software Metrics**



Sponsored through Framework Programme Sixth (Call 5) by

Document Information

Version: 1.0
Date : Apr 15, 2007
 revision: 0

Owning Partner: AUTH

Author(s):
 Ionannis Antoniadis
 Jesus M. González-Barahona
 Santiago Dueñas
 Gregorio Robles
 Stefan Koch
 Ksenia Fraczek
 Anis Hadzisalihovic

Reviewer(s):
 Jesus M. González-Barahona

To:
 CONSORTIUM

Purpose of distribution:
Initial Version

The FLOSSMetrics Consortium consists of: Universidad Rey Juan Carlos, University of Maastrich, Wirtschaftsuniversitaet Wien, Aristotle University of Thessaloniki, Conecta s.r.l., Zea Partners and Philips Medical Systems PMS Nederland B.V.

**Printed
on at**

Status:

- Draft
- To be reviewed
- Proposal
- Final/Released

Confidentiality:

- Public - Intended for public use
- Restricted - Intended for FLOSSMETRICS consortium only
- Confidential - Intended for individual partner only


Deliverable ID: D2.1

Title:


Design of retrieval system

License for distribution:

This work is licensed under a [Creative Commons Attribution-Share Alike 2.5 License](http://creativecommons.org/licenses/by-sa/2.5/).
 (The license can be found in <http://creativecommons.org/licenses/by-sa/2.5/>)

	Design of retrieval system Deliverable ID: D2.1	Page : 2 of 90
		Version: 1.0 Date: Apr 15, 2007
		Status : Final Confid : Public

<p>Deliverable: D2.1</p> <p>Title: Design of retrieval system</p> <p>Executive Summary:</p> <p>One of the main activities to be performed in the FLOSSMetrics project is the retrieval of information from thousands of libre (free, open source) software repositories. One key component needed for this activity is the system that will automate it: the retrieval system. This report describes its main requirements, and its design (including how external tools will be integrated in it).</p> <p>This report will later be used as a base for the implementation of the system itself. In addition, the report includes an exhaustive list of tools that are useful for the retrieval of data from repositories, and which are candidates for integration in the system. This list is in part a result of other project, QUALOSS.</p>
--

	<p style="text-align: center;">Design of retrieval system</p> <p style="text-align: center;">Deliverable ID: D2.1</p>	Page : 3 of 90
		Version: 1.0 Date: Apr 15, 2007
		Status : Final Confid : Public

CHANGE LOG

Ver.	Date	Author	Description
0.1	05/12/2006	Jesús M. González-Barahona	Initial proposal for structure
0.2	21/03/2007	Ionannis Antoniadis Jesús M. González-Barahona Santiago Dueñas Stefan Koch Ksenia Fraczek Anis Hadzisalihovic	First comprehensive version, some details are still missing
1.0	15/04/2007	Ionannis Antoniadis Jesús M. González-Barahona Santiago Dueñas Gregorio Robles Stefan Koch Ksenia Fraczek Anis Hadzisalihovic	Final version

APPLICABLE DOCUMENT LIST

Ref.	Title, author, source, date, status	Deliverable Identification

Contents

1	Introduction	6
2	Design of the retrieval system	7
2.1	Requirements	7
2.1.1	Functional Requirements	7
2.1.2	Non-functional Requirements	8
2.2	High-level design	9
2.3	Description of System Components	9
2.3.1	Database of projects	9
2.3.2	Ontology of tools	10
2.3.3	Source Data Manager (SDM)	10
2.3.4	Project Update Decision Module	11
2.3.5	Tool Input Adapter	12
2.3.6	Tool Output Processor	13
2.4	Run scenario	14
3	Open issues and future work	15
3.1	External modules	15
3.1.1	Web user interface	15
3.1.2	System Database Manager	16
A	Available Tools	17
A.1	CVSAnalY	17
A.2	MailingListStats	19
A.3	SEAL	21
A.4	pyTernity	22
A.5	Wholine	23
A.6	Carnarvon	25
A.7	SLOCCount	26
A.8	CODD	27
A.9	Audit-Perl	28
A.10	Bloof	29
A.11	C Code Analyzer	31
A.12	calltree	33
A.13	Cinclude2dot	34
A.14	codeanalyzer	35
A.15	cstor	36
A.16	CVS Statistics Generator	38

A.17 CvsGraph	39
A.18 DEPS	40
A.19 pylint	41
A.20 Eclipse Metrics Plugin	42
A.21 Perl Metrics	43
A.22 MetricsAnalyzer	44
A.23 PyMetrics	45
A.24 The Metrics (for Python)	47
A.25 Pymmetry - Python Trust Metrics	49
A.26 CodeMetrics	50
A.27 RSM - Resource-Standard-Metrics	51
A.28 aopmetrics	53
A.29 Delphi Code Analyser	55
A.30 Pythius	57
A.31 Perfometer	58
A.32 CodeAnalyzer - Multi-Platform Java Code Analyzer	59
A.33 CAP - Code Analysis Plugin	60
A.34 cvsstats	61
A.35 cccc	62
A.36 LOCC	63
A.37 OSSMole	65
A.38 Koch 2004 SourceForge/CVS + SourceForge	66
A.38.1 Usage	66
A.38.2 Input	67
A.38.3 Process	68
A.38.4 Output	74
A.39 Koch 1999 Gnome/CVS+Email	75
A.39.1 Usage	75
A.39.2 Input	76
A.39.3 Process	76
A.39.4 Output	79
A.40 Iskenderoglu 2006 Email	80
A.40.1 Usage	80
A.40.2 Input	80
A.40.3 Process	80
A.40.4 Output	85
A.41 Aksu 2006 Bugzilla	86
A.41.1 Usage	86
A.41.2 Input	86
A.41.3 Process	86
A.41.4 Output	88

Chapter 1

Introduction

This deliverable is concerned with the FLOSSMetrics retrieval system. The main purpose of this system is to download publicly available information from FLOSS projects that is available on the Internet and to extract facts and data from these sources by means of external, if possible already implemented tools. The main goal of this deliverable is to design a system that serves for the purpose of analyzing publicly available information from FLOSS projects on the Internet. The retrieval system should be a software that acts as a conceptual black box. The input is given by a set of URLs where the information can be retrieved while the output delivered is a set of databases obtained from the analysis performed by external tools. The output of the system is given by the sum of the outputs provided by all these external tools in a relational database-suited format. The design, proper functioning and optimization of the database that holds the resulting data is not a topic in this deliverable as it will be discussed, designed and implemented in later deliverables in the FLOSSMetrics project. Each tool will hence provide an independent database at this level.

The scope of this deliverable is to present the design of the retrieval system at a high conceptional level. In this sense, requirements and the description of the components of the system will be presented and discussed. The actual implementation of the system, including a lower-level description of classes and functions will be covered in a future FLOSSMetrics deliverable.

For the purpose of the deliverable, we have used some work from the QUALOSS EC-funded project, mainly the list of tools that analyze FLOSS projects. This list can be found in the appendix and provides insight into the possibilities that the FLOSSMetrics retrieval system will offer as output.

The SQO-OSS project has been contacted regarding this retrieval system as this project has been identified as a possible user.

The conception of the retrieval system has been done in parallel by two partners of the project. Both have independently discussed the goals of the system and have proposed an architecture for it at the module and high functional level. These documents have been exchanged and reviewed by the other partner and a joint proposal has been achieved after several iterations. This document describes the output of this process.

Chapter 2

Design of the retrieval system

This chapter describes the main lines of the retrieval system. It presents the requirements of the system, illustrates the high-level design, describes its components and finally gives a running scenario.

2.1 Requirements

The retrieval system has been designed considering following requirements:

2.1.1 Functional Requirements

- **Access to Source Repositories:** The system should be capable to access to the distinct FLOSS projects repositories . As was described in the deliverable 1.1 "Study of Available Tools" of FLOSSMetrics project, there are different types of repositories with different properties and behaviours. For these version, only will be supported the repositories described with deep: system control version, mailing lists and bug tracking systems. To be able to access to the repositories, previously the parameters of access should have been obtained and will be input of the retrieval system. The access could be by remote or local. This last case is supported due to the sources could be download previously to carry out with the non functional requirement of non-intrusive system.
- **Repository Finder Interaction:** The system should interact with the repository finder to obtain the access parameters of FLOSS repositories. The repository finder is a tool developed in the D1.3 deliverable of FLOSSMetrics. The repository finder tasks are: find FLOSS repositories, analyse its characteristics and obtain its access parameters in an automatic way. These facilitates to the system the access to FLOSS repositories due to, in other way, the researcher should provide thousands of parameters by hand. These does not implies that the way of provide parameters will done only with the repository finder.

- **Data extraction:** The system should be capable of extract data from the source repositories. This extraction will be performed by the re-utilization of external tools developed for these tasks. A set of these tools can be found at the of of this document in Appendix A.
- **Storage:** After the extraction, the data need to be storage for further analysis. During the FLOSSMetrics project, a database will be designed and developed to storage this data. The system should be interact with the database.

2.1.2 Non-functional Requirements

- **Modular:** The system should be prepared to include external tools for the extraction of data and facts and the measurement of public repositories. The rationale for this requirement is that there already exists a large amount of tools that can be reused for this purpose. As many of them are released under a libre software license, embedding them in the system should be a characteristic to strive for. External tools should therefore be considered as plug-ins, that should be embedded into the system. This assumes a clear plug-in interface, so that new external tools can be easily included. The system is therefore designed to accommodate user-provided plug-ins later on during its life-cycle.
- **Non-intrusive:** The data-gathering activity may be a process that overloads the equipment used by software projects for a purpose that differs from the original intentions of the developers. This may lead to malfunctioning of the project infrastructure and the adoption of measures by the project that may make future research studies impossible (for instance, to ban access). Therefore, the retrieval system should include a way to minimize the effect of the analysis performed on foreign infrastructure, mainly by having a local copy (or cache) of the repositories under study. While this may require further hardware and network infrastructure at the research center, it will speed up the analysis.
- **Easy configuration and running procedure:** Users should only need to provide a list of URLs where the publicly project information (versioning system, mailing lists, etc.) can be retrieved from, with the rest of the process handled in a transparent manner by the retrieval system.
- **Overall control:** The retrieval system should have an overall controlling module that supervises the whole process. This module should report and, if possible, solve problems that appear due to malfunctioning of the network, of the software repositories under study or of the software tools in use. Logs and errors should are to be stored for human inspection if wanted. The prior requirement of modularity implies that the system will have to manage a range of different behaviors for the plug-ins. So, for instance, external tools may behave differently when errors occur or may provide information of the process with different levels of verbosity if at all that make the reproduction of the retrieval process or the identification of errors more difficult.

2.2 High-level design

The high-level design of the retrieval system is shown in Figure 2.1. The system has been designed, as specified by the requirements, as a black box where the input will be information extracted from the Internet and the output a set of databases with the results of the analysis performed by the various tools integrated into the system.

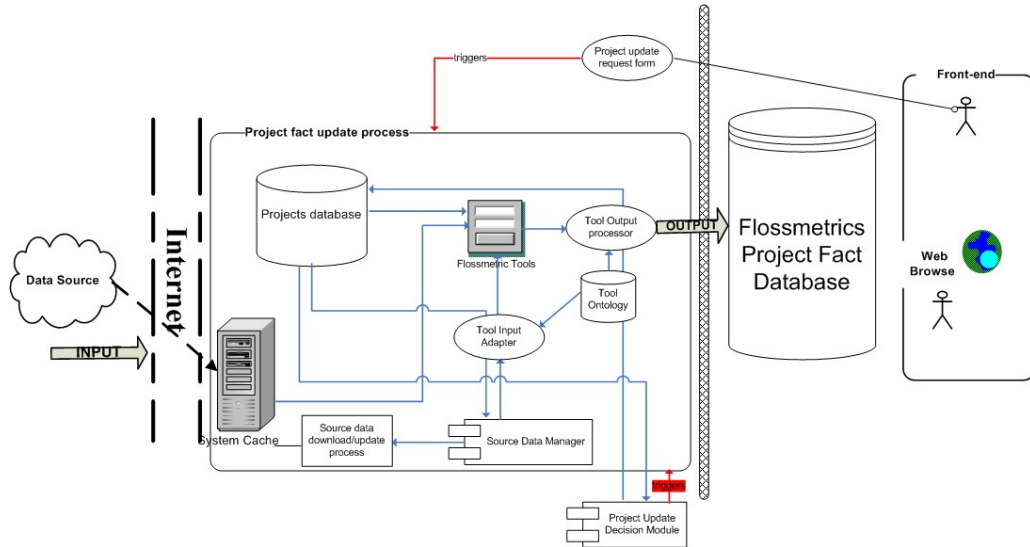


Figure 2.1: High-level design of the retrieval system.

Figure 2.1 shows which components we consider part of the retrieval system and which not. In this sense, the data sources from libre software projects are not considered part of it, but the possibility of having a local copy (a cache) of these sources is taken into account for the retrieval system. On the output side, the database with facts about the projects does not belong to the retrieval system and will be handled as the entity that will receive the results of the system. In this sense, a front-end to results (possibly a web interface) that displays the results in a user-friendly way has not been designed as part of the retrieval system.

2.3 Description of System Components

The inner parts of the box have been divided into various modules to allow the integration of external software as well as to handle multiple tasks and activities that are intended to be accomplished by the retrieval system. These modules are described in the following sections.

2.3.1 Database of projects

The database of projects contains information that is needed to retrieve the data from the Internet about the projects. This information should be given

by the user as a textual file (possibly in XML format), via a web form or with information from another database. The values will be used to identify the information sources (versioning systems, mailing lists, etc.), the type of tools to be launched and other database and debugging options. The values can be:

- URLs: Universal Resource Locator of the source. A source may be a versioning system, a mailing list archive, a bug tracking-system, etc.
- Repositories: A URL for a repository may be given. If this is the case, a spider that searches for information sources (versioning systems, mailing list archives, etc.) will be launched and a set of URLs as in the previous point will be generated.
- Type of tool to be launched: The analysis can be global (i.e. affecting all the available analysis tools in the system) or just partial (i.e. only some tools will analyze the current information source).
- Debugging options: The user may specify various debugging options that will allow him to monitor the retrieval and analysis process at various levels of detail.
- Database options: Specific database configurations may be required for special purposes.

Additional information should be included to allow incremental analysis, i.e. to perform task on a new set of data without having to execute the tools on data already analyzed.

2.3.2 Ontology of tools

The ontology of tools should provide information about whether a given tool has the capability or not to do incremental analysis. Together with the information stored in the database of projects, the command line arguments or the calls to functions for the various tools should be generated. Possible dependencies or complementarities between different tools can also be stored here.

2.3.3 Source Data Manager (SDM)

The purpose of the Source Data Manager (SDM) is to manage the source data (source code files, repositories, mailing list archives, etc.) so that it is conveniently stored locally. The task of this module is to handle retrieval of the artifacts and have everything ready to analyze them in a future step. The SDM should therefore take into account what artefact's are already stored locally, for which of the project data sources there is a local copy (a cache), etc. When a given source has already been downloaded, it will also consider if it is convenient to update it or not.

The input required by the SDM is a list of one or more project IDs from the FLOSS project table. For each project ID, following information should also be available:

1. an array of Boolean values (1/0) indicating which input types (eg. C++ source files tarball, CVS repository, mailing list mbox file, etc) will be checked for downloads/updates and which not,

2. an array of reals indicating the time (in days) that should pass before the particular project cached data should be updated. If $T < 0$, this indicates that the corresponding data should be deleted from cache.

The output will consist in an array of integers indicating the result of the download/update procedure at possibly different levels of granularity (project-level or even file-level). For example, a 1 may denote a successful update or download, while a 0 will be indicative for an error (for instance, no such data in the local cache). These values (and the corresponding error description) will be stored to a log file.

The process that the SDM will follow is described next:

- For the data download/update option:
 - The SDM checks for each project indicated in the ID list and for each input type indicated by a value of 1 in the corresponding Boolean array whether data exists in local cache or not.
 - If the data does not exist in local cache or data exists locally but files are older than the specified time parameter T , then downloading/update the data over the Internet by looking up the necessary access information in Project Database. If $T < 0$, corresponding data that exists in local cache is deleted.
 - If the download is successful, date information in Project Database for the corresponding file types is updated. If the download was not successful, output proper error message in system log and also record an error code at a corresponding field in project database, so that projects with problems in downloads/updates can be easily identified.
 - For each project that data is downloaded for the first time, it will be stored locally in separate path (eg. `<dir_project_name(ID)>/Source_data/SVN` or `<dir_project_name(ID)>/Source_data/developer_list` according to some proper naming convention. At the same time, the corresponding paths are stored in proper fields of project database under the particular project id.
- For the data download/update option:
 - Deletes for each project indicated in the ID list and for each input type indicated by a value of 1 in corresponding Boolean array and if the corresponding time parameter.

2.3.4 Project Update Decision Module

The aim of the Project Update Decision Module (PUDM) is to decide whether to download/update source data (repository, mailing list etc.) for a project and what project data to delete, if system cache is full. It runs as a scheduled task and provides the required input to Source Data Manager.

This input will be a list of project Ids, which will have to be checked. By default, the Ids are those of all projects in project database.

The process that this module will follow is described next:

1. The system should first check in project database whether paths (URL) for corresponding files exist for a particular project. If not, then the corresponding Boolean field value in project database is set to *false*.
2. For each project, and for each corresponding path, the system will calculate the time T that should pass before the particular project cached data should be updated. The algorithm for the calculation should be based upon:
 - (a) The average request frequency in a past time window for viewing corresponding project fact data. This rate must be recorded in project fact database.
 - (b) A measure of the average difference between two consecutive versions of the corresponding files. For example, if two consecutive updates of a particular project repository show only slight changes in source code on the average for the past few updates, then the time for the next update should be increased.
 - (c) If the above information is not available, there will be a default time set initially by the project administrator in corresponding fields in the project database.
3. Negative values for T, i.e. indication that a file should be deleted will be given only if allocated disk space for system cache is nearly used up. In this case, project source data will be deleted starting with the projects with lowest user request frequency, slowest development rate (see item 2.2 above), and those which take up the least amount of space in system cache (because, if needed, they can be downloaded again the fastest from the Internet).
4. For each project id, the system sets the values to the Boolean fields in the project database indicating which input types (eg. C++ source files tarball, CVS repository, mailing list mbox file, etc) will be checked for downloads/updates and which not.
5. For each project id, the system sets the values to the corresponding database fields indicating the time T that should pass before the particular project cached data should be updated (T<0, indicates that corresponding data should be *deleted*).
6. A trigger to call the SDM is issued.

2.3.5 Tool Input Adapter

The objective of the Tool Input Adapter is to automatically adapt the format of input data as they exist in project database according to the execute requirements of a particular metric tool.

The input of this module should be the id of the tool to be used for the analysis and the id of the project to be analyzed, while the output will give the argument sequence as required by the tool for the particular purpose (as configured) and for the particular project.

The complete process will run as follows:

1. The system checks the tool ontology (described by proper XML schema) for the description of tool input. Example: Suppose a particular tool requires a path to a repository plus file selection indicated by wild-cards (eg. Debian/* or debian/kernel/ab*.*). Another tool may require just the path to the repository with no file selection. In the first case the adapter will add a /* to the path.
2. For a full specification of how this module will work, a complete ontology of the existing tools must be created.

2.3.6 Tool Output Processor

The goal of the Tool Output Processor is to process the output of each analyzing tool and to provide the results in the proper way for storing it in a database.

The input of this module will be the id of the project under study, the id of the tool that is used and other configuration options of the analyzing tool such as the numerical output and the output files.

The output could be a text file in database (SQL) format, although data could be stored directly into the database avoiding the step of having an intermediate file. In addition and in order to monitor the process, messages will be displayed in the system log.

The procedure by which this module works is described next:

1. The system checks the tool ontology (described by proper XML schema) for the description of tool output. For each tool, the output processor will have a specific procedure of what to do with the output data. For example, to which fields in the project fact database to add scalar output data, in which local path to save output files such as image files, etc.
2. For a full specification of how this module will work, a complete ontology of the existing tools must be created.

2.4 Run scenario

The run scenario for full project data update describes the process that the retrieval system goes through to obtain the output information. The retrieval system handles data that is constantly evolving, so it will work as an endless machine that constantly retrieves information from the repositories stored in a list. New repositories may be included in the list and thus added to the loop.

The retrieval system will have two lists associated which will mark the status of the system. The first one is a list with the projects to be studied, while the second one will contain the list of analysis tools (plug-ins) that have been embedded into the system.

Hence, the process will run as follows. For each project in the list of projects, the following tasks will be fulfilled:

1. Check project database whether retrieval allowed by administrator. If yes, continue, else move to next project.
2. For each tool in the System
 - (a) Check tool table whether tool is allowed to run by system administrator. If yes continue, else move to next tool.
 - (b) Check tool database for type of input required by the tool (eg. CVS/SVN repository, source code files, mailing list mbox files etc) and pass input to Source Data Manager (SDM) module to check the availability/up-to-dateness of data either in local cache or remote source.
 - (c) If SDM returns a success indication for availability of data, then feed local path to corresponding tool, after being processed by the Tool Input Adapter, which makes necessary adjustments to the raw file paths to fit the particular tool execute requirements. If SDM returns no success, then the process aborts and the proper error message is returned to system log.
 - (d) After tool executes, output files are processed by the Tool Output Processor which moves each file to the proper directory, sets the local names of files according to predefined naming convention, and updates the Project Facts Database with corresponding data.

Chapter 3

Open issues and future work

The retrieval system has been designed to accomplish the tasks of downloading the data from the data sources on the Internet and of performing an analyses of these data by means of mainly external tools that can be embedded into the system.

There are two main open issues that are not handled in the system:

- The first one is that the retrieval system does not consider any further data massaging and treatment of the results offered by the different analysis tools (integration and consolidation). This will be discussed in a later deliverable that will be centered on the design of the database structure and other actions to preserve its integrity and confidence.
- The second issue is the interaction with the user: at first, the retrieval system will work as a set of sequential scripts, probably integrated into a unique command line call. But future work should include the creation of a user-friendly interface that allows the user not only to configure and run the tool easily, but also to monitor the process and inspect the error logs.

3.1 External modules

Although the database and the user interface are not part of the retrieval system and hence have to be considered as external modules, in the following subsections we describe some guidelines that these modules could follow.

3.1.1 Web user interface

The aim of the web user interface is to allow users to run and control the retrieval system easily. It should also have the required functionality to request new or update data for a given project or just to run a specific tool on the projects. System administrators may use the same web-based track to perform manually issue requests.

The interface should consist of a series of web forms that hide the internal values used by the retrieval system (project id, tool id) from the user.

The output will provide feedback to the user about the process, pointing out if it finished correctly and the errors that occurred. A link to the generated logs should also be provided.

Basically, the functioning should be as follows:

1. The user sees a list of all available projects in project database which the system administrator has marked as active in the database (including the ones for which no project data yet exist) with a proper indication of when the data were last updated. He/she may select only one project. Alternatively, the user can provide respective information for a new project (URL of repositories etc.).
2. The user sees a list of available tools with a short description of what each tool does. The user may select one or more tools.
3. The project id and tool id sequence is posted to server by standard HTTP message.
4. On the server side the user request updates the user request counters corresponding to the particular project and to each particular data type required by the tool chosen by the user.
5. The request is queued and executed.

3.1.2 System Database Manager

The purpose of the system database manager is to handle the project fact database (the output of the various analysis tools). Technologically this can be an existing database management tool such as MySQL or Postgres.

A future deliverable will be specifically devoted to discuss the database requirements as well as the database architecture.

Appendix A

Available Tools

This appendix includes an exhaustive list of tools useful in the retrieval of information from libre software projects repositories. This list is included here for convenience, since any of the tools could be integrated in the retrieval system, and several of them will in fact be integrated.

This appendix is based on a similar list prepared for the QUALOSS project (deliverable D1.1 - **Evaluation Report on Existing Tools and Existing FLOSS Repositories**). Here it is completed and (slightly) edited to suit better the needs of the project, and is included with the permission of the QUALOSS project.

The order for presentation of the tools has no special meaning.

A.1 CVSanaly

- **URL:** <http://cvsanaly.tigris.org/>
- **Author:** Alvaro Navarro <anavarro at gsync.escet.urjc.es>, Gregorio Robles <greg at gsync.escet.urjc.es>.
- **Description:** CVSanaly is a tool that extracts statistical information out of CVS (and recently Subversion) repository logs and transforms it in database SQL formats. This tool has a web interface, called CVSanaly-web, where the results can be retrieved and analyzed in an easy way.
- **Technologies:** Python
- **Execution mode:**
 - `python cvsanaly.py`
- **Input data:** CVS or SVN repositories to study.
- **Output data:** Analysis of CVS and SVN repositories.
- **Technology limitation:**
 - Some features included for CVS analysis are not available yet for SVN.

- **Maturity:** stable.
- **Dependencies:** cvs, mysql-server, python, python-mysql, python-mysqldb, python-imaging, gnuplot, ploticus.
- **Documentation:**
 - <http://cvsanaly.tigris.org/servlets/ProjectDocumentList>
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Number of developers making changes in the project.
 - Number of non-active developers.
 - Number of changes made in the project.
 - Number of modules present in the project.
 - Number of files present in the project.
 - Number of files changed in the project.
 - Metainformation associated to every commit can be used to reconstruct the modification requests.
 - Username of the commiter.
 - File (or files in the system supports modification requests) where the changes occurred.
 - Date of the change.
 - Number of lines added, removed or changed (counted as both added and removed).
 - Inspecting the contents of the file or looking at its extension, filetype can be obtained.
 - Information about the code ownership.
 - Size and complexity measurements (as in version control systems source data).
 - Number of commits.

A.2 MailingListStats

- **URL:** <https://svn.libresoft.es/svn/projects/trunk/maillingListStat>
- **Author:** Israel Herraiz <herraiz at gsync.escet.urjc.es>.
- **Description:** MailingListStats is a tool for mapping mbox files of any mailing list to a database.
- **Technologies:** Python, MySQL.
- **Execution mode:**
 - A little 'How To' about mailingListStat execution can be found in <https://svn.libresoft.es/svn/projects/trunk/maillingListStat/doc/MLS.Howto.txt>
- **Input data:** URLs of each mailing list or a directory with the mboxes of each mailing list separated by directories.
- **Output data:** Database with information about headers of each parsed mbox and its mailing list.
- **Technology limitation:**
 - Each mailing list database cannot be updated without reparsed all mailing list again.
 - Duplicated information is stored in the database when MLS is running using a mbox of a mailing list studied previously.
 - Not fault tolerant. MLS cannot resume its work when the system down unannounced. Is needed to run MLS from the beginning for all mailing lists.
- **Maturity:** alpha.
- **Dependencies:** python, mysql.
- **Documentation:**
 - <https://svn.libresoft.es/svn/projects/trunk/maillingListStat/doc/MLS.Howto.txt>.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Name of the poster.
 - Number of non-active developers.
 - Name of the poster.
 - E-mail address of the poster.
 - Date when the message was sent by the poster, and received in the mailing list server.
 - Subject of the message.
 - Mailing list address where the message was sent to.

- Name and e-mail addresses of other recipients different of the list (for example, other people include in the CC field of the message).
- Unique identification tag for the message in the mailing list.
- Identification tag of the original message if the message is a reply to another.
- Content of the message, including usually attachments.
- Name of the program used to write the message.
- Activity (number of messages).
- Participation (number of people participating).
- Number of messages over time.
- Number of people writing in the list over time.
- SNA methods to study the flow of information within the community. Evolution over time.
- Mean length of the threads over time. Standard deviation.
- Statistics of usage of the different programs in the mailing list.
- List of keywords of the topics discussed in the lists.
- The list of keywords could be obtained in a monthly basis.
- With a list of keyword for each month, we can find out which topics were the most discussed and if the topics have evolved.
- Can be crossed correlated topics with people involved in the discussion of those topics.

A.3 SEAL

- **URL:** <https://svn.libresoft.es/svn/projects/branches/improvedSeal/>
- **Author:** Gregorio Robles <grex at gsync.escet.urjc.es>, Roberto Andradas Izquierdo <randradas at gsync.escet.urjc.es>
- **Description:** SEAL is a tool for identify persons in a mailing list conserving their privacy.
- **Technologies:** Python, MySQL.
- **Execution mode:**
 - Export data about a mailing list from MailingListStat database:
 - * i.e : `python xx2xml.py > data.xml` (`xx2xml.py` is a python script which must be configured in order to connect to the MLS database).
 - Run SEAL:
 - * i.e: `python seal-feed.py data.xml`
- **Input data:** XML file with names and email addresses associated with these names. Is generated by `xx2xml.py` script.
- **Output data:** A database with the information of the identities of each person.
- **Technology limitation:**
 - SEAL has to recalculate all matches (related identities) each time new identities are introduced from a XML file (although this task is not slow).
 - Not fault tolerant. SEAL cannot resume its work when the system down unannounced. Is needed to run SEAL from the beginning for all mailing lists.
 - Unreadable code.
- **Maturity:** alpha.
- **Dependencies:** python, mysql.
- **Documentation:**
 - <https://svn.libresoft.es/svn/projects/branches/improvedSeal/README>
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Information of the identities of each person who participate in a mailing list.

A.4 pyTernity

- **URL:** <https://svn.libresoft.es/svn/projects/trunk/pyternity/>
- **Author:** Diego Barceló <dibarman at gsync.escet.urjc.es>, Gregorio Robles <grex at gsync.escet.urjc.es>
- **Description:** pyTernity is a statistical analysis tool for building and analyzing distributions of ownership/contribution data from software source packages primarily designed to study the patterns in contributions from developers working on libre software projects.
- **Technologies:** Python, MySQL.
- **Execution mode:**
 - **Edit:** `config.py` to configure pyTernity
 - **Run:**
 - * `python pyTernity.py`
 - * `python pyTernity_cleaning.py`
 - * `python pyTernity_copyright_distribution.py`
- **Input data:** Source files of projects packed in tarballs or uncompressed into directories.
- **Output data:** A database with the information of the distinct analyzed projects.
- **Technology limitation:**
 - Could be duplicated information if the program is rerunning and the database is not deleted.
 - Some authors could not be neither detected nor classified by the application.
 - Dangerous application (could delete directories if the configuration is not set properly)
 - Unreadable code (with some bugs)
- **Maturity:** unstable.
- **Dependencies:** python, mysql, nilsimsa, sloccount.
- **Documentation:**
 - <https://svn.libresoft.es/svn/projects/trunk/pyternity/trunk/README>
- **License:** GNU General Public License (GPL)
- **Metrics:**
 - Distributions of ownership/contribution data from software source packages.
 - Patterns in contributions from developers.

A.5 Wholine

- **URL:** <https://svn.libresoft.es/svn/projects/trunk/wholine2>
- **Author:** Jorge Gascón <jgascon at gsync.escet.urjc.es>
- **Description:** Wholine is a tool which analyze a CVS/SVN repository, obtains all revisions of one repository and does an intensive analysis of the modifications of each line of code.
- **Technologies:** Python
- **Execution mode:**
 - `python wholine <project_name> <protocol> <protocol.project_url> [OPTIONS]`
 - * i.e : `python wholine wholine svn https://svn.libresoft.es/svn/projects/trunk/wholine2`
- **Input data:** URL of one CVS/SVN repository
- **Output data:** Statistics and graphs with analysis of the repository.
- **Maturity:** stable.
- **Dependencies:** python, cvs, subversion.
- **Documentation:**
 - <https://svn.libresoft.es/svn/projects/trunk/wholine2/Doc/>
- **License:** GNU General Public License (GPL)
- **Metrics:**
 - Init date for control version system using.
 - Committers.
 - Number of committers.
 - Number of files.
 - Number of revisions and their age.
 - Number of code lines and their modifications.
 - Most changed files.
 - Kind of contributions (added lines, modified lines, deleted lines).
 - The most active committers and percent of lines they have changed.
 - Lines more changed in all project.
 - Kind of modifications made in most changed code lines:
 - * refactor constants.
 - * space assignation function.
 - * different types.
 - * spaces functions.

- * refactor variable.
- * separator variable.
- * refactor assignation function.
- * refactor function.
- * refactor variable.
- * spaces variable.
- Kind of modifications made in all code lines:
 - * characteristics before plus next.
 - * spaces unknown.
 - * separator unknown.
 - * spaces constants.
 - * comments.
- Most changed lines in most changed files.
- Most changed lines by different developers (average) (For line X 3.3 developers have changed it).

A.6 Carnarvon

- **URL:** <http://carnarvon.tigris.org>
- **Author:** Alvaro Navarro <anavarro@gsyc.escet.urjc.es>, Carlos González <carcgonzalezsanch@uoc.edu>
- **Description:** Carnarvon analyses how old the software system is on a per-line basis and extracts figures and indexes that make it possible to identify how ‘old’ the software is, how much it has been maintained and how much effort it may suppose to maintain it in the future.
- **Technologies:** Python.
- **Execution mode:**
 - **Install:** `python setup.py install`
 - **Configure:** `carnarvon -w my.conf`
 - **Run:** `carnarvon my.conf`
 - **Analysis:** After carnarvon finishes the analysis of the given project, run the next tools in order to get graphs and a nice website: `carnarvon2web my.conf`
- **Input data:** A downloaded (check out) project from its CVS/SVN repository.
- **Output data:** A nice website with graphs about the analysis.
- **Technology limitation:** May be interesting that Carnarvon can do the repository check out by itself. But, authors says that this is a feature.
- **Maturity:** stable.
- **Dependencies:** python, cvs, subversion, python-2.x-MySQLdb, gnuplot.
- **Documentation:**
 - <http://carnarvon.tigris.org/documentation/quick-guide.html>.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - How ‘old’ the software is.
 - How much it has been maintained.
 - * Changes made in a line.
 - * Lines added, removed or changed (counted as both added and removed).
 - How much effort it may suppose to maintain it in the future.

A.7 SLOCCount

- **URL:** <http://www.dwheeler.com/sloccount>
- **Author:** David Wheeler <[dwheeler at dwheeler.com](mailto:dwheeler@dwheeler.com)>
- **Description:** SLOCCount is a suite of programs for counting physical source lines of code (SLOC) in possibly large software systems. The tool can count physical SLOC for a wide number of languages; can take a large set of files and automatically categorize their types using a number of different heuristics; and also, comes with analysis tools.
- **Technologies:** C, Perl, sh
- **Execution mode:**
 - `sloccount <dir>`
- **Input data:** Directory containing source code to analyze.
- **Output data:** Analysis with data grouped by language and development effort estimated.
- **Technology limitation:**
 - Not capable of analyze CVS/SVN modules.
- **Maturity:** stable.
- **Dependencies:** No dependencies found.
- **Documentation:**
 - <http://www.dwheeler.com/sloccount/sloccount.html>
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - SLOCCount can automatically identify and measure a great variety of languages such as Ada, Assembly, awk, C/C++, C#, Fortran, Haskell, Java, Pascal, Perl, PHP or Python.
 - Handles language constructs that are often mishandled by other tools, such as Python's constant strings when used as comments and Perl's "perlpod" documentation.
 - Estimate the effort, money and time during the development.
 - Use the basic COCOMO model, which makes these estimates solely from the count of lines of code.

A.8 CODD

- **URL:** <http://libresoft.urjc.es/Tools/CODD>
- **Author:** Vipul Ved Prakash <mail at vipul.net>, Rishab Ghosh <rishab at dxm.org>
- **Description:** Codd is a statistical analysis tool for building and analyzing distributions of ownership/contribution data for software source packages. Codd is primarily designed to study the patterns in contributions from developers working on Open Source Software projects.
- **Technologies:** Perl
- **Execution mode:**
 - **Configure:** create the directories `codds` and `db`
 - **Run:**
 - * `codd-create-package -dir=codds/ -indir=<full path to packages dir> -workdir=<path to tmp dir>` (for packages stored in one directory)
 - * `codd-create-new -dir=codds/ -workdir=<full path to src dir> -codd=<output file>.codd` (for directory trees of source code)
 - * `codd-dep -build` (builds database with interfaces and shared files)
 - * `codd-dep -shared -resolve` (resolves interface and shared files dependencies)
 - * `codd-dep -ownergrep` (greps for authorship in the `-non-shared-files`)
 - * `codd-dep -sdb` (searches authors in shared files)
 - * `codd-render codds/package.codd` (obtain the authors of a package)
 - * `codd-2sql codds/package.codd` (transform from the `codds` format to SQL)
 - * `codds2sql` (transform all `codds` in a directory)
- **Input data:** Software source packages.
- **Output data:** Analysis of ownership.
- **Maturity:** stable.
- **Dependencies:** No dependencies found.
- **Documentation:** No documentation found.
- **License:** GNU General Public License (GPL) - after version 2.0.17 - , Artistic License - before version 2.0.17 -.
- **Metrics:**
 - Development patterns used by developers.

A.9 Audit-Perl

- **URL:** <http://hinterhof.net/max/audit-perl>
- **Author:** Max Vozeler <max at hinterhof.net>
- **Description:** `Audit::Source` is a group of modules to make processing of source code vulnerability scanners easier. Each scanner is implemented by a separate module which parses the results and stores them into a common data format. Results from different scanners can be matched against each other.
- **Technologies:** Perl, C.
- **Execution mode:**
 - Using Audit-Perl in the code

```
use Audit::Source;
$audit = new Audit::Source;
$audit->run_all($file);
$audit->textdump();
```
- **Input data:** Source code files.
- **Output data:** Information about vulnerable lines of code.
- **Technology limitation:** Include perl modules in the source code.
- **Maturity:** stable.
- **Dependencies:** perl.
- **Documentation:**
 - README file within tarball.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Processing of source code vulnerability scanners.

A.10 Bloof

- **URL:** <http://bloof.sourceforge.net/>
- **Author:** Lukasz Pekacki <pekacki at users.sourceforge.net>, Dirk Draheim <draheim at users.sourceforge.net>
- **Description:** Bloof is an infrastructure for analytical processing of version control data. Its main application, the Bloof Browser provides a comfortable user interface for analysing and navigating the history of software projects.
- **Technologies:** Java.
- **Execution mode:**
 - Using Bloof in your own code:

```
public class BloofExample {

    public static void main(String[] args) {
        try {
            RepositoryLocation repo =
                new RepositoryLocation(":pserver:anonymous@cvs.sourceforge.net" +
                                      ":/cvsroot/bloof:bloof:anoncvs");

            LoginDetails loginDetails = new LoginDetails("anonymous", "");
            ScmAccess cvs = new CvsAccess(repo, loginDetails, "bloof", "example cvs module");

            DbAccess database =
                new DefaultDbAccess(
                    "Postgres Databae on local machine",
                    "bloof",
                    "bloof",
                    "jdbc:postgresql:fasel");

            Bloof.importProject(cvs, database);
            MetricParameter metricParam = new MetricParameter(null,null,null);
            Metric metric = MetricRegistry.createMetric("Lines of code");
            metric.setupMetric(metricParam);
            Bloof.runMetric(metric);
            metric.runMetric();
            Bloof.exportResult("Lines of code",null);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

- **Input data:** CVS repository and a database.
- **Output data:** Bloof generates graphs and a database with information about the analysis.
- **Technology limitation:** Include Bloof functions in the source code.
- **Maturity:** stable.
- **Dependencies:** Java runtime environment version 1.4 or greater.
- **Documentation:**
 - <http://bloof.sourceforge.net/index.html>

- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Evolution of software projects related to next attributes:
 - * file age.
 - * most frequently changed files.
 - * top contributors.
 - * biggest changes.
 - * file count.
 - * lines of code.
 - * list of developers.
 - * yime line of the rate of entropy in the development process.
 - * daily colaboration.
 - * change impact.

A.11 C Code Analyzer

- **URL:** <http://www.drugphish.ch/~jonny/cca.html>
- **Author:** Jonathan Heusser <jonny at drugphish.ch>
- **Description:** The C Code Analyzer (CCA) is a static analysis tool for detecting potential security problems in C source code. This analyzer was built with the following principles in mind:
 - CCA tries to spot only the errors that can actually cause problems. Not every strepy is a security problem.
 - No code annotations or tweaking is required - it's fully automatic. It's not realistic that an auditor has to crawl through thousands of LOC telling the analyzer "watch this, watch that". It's possible to extend the set of dangerous functions, malloc wrappers etc, though.
 - Seamless integration in existing development platforms. The Eclipse platform has been chosen as completion to the command line tool.
- **Technologies:** C, Perl, ML, sh.
- **Execution mode:**
 - i.e : `cilly --mydangerous=mydang --docodechecker asf_mmst_streaming.c 2>/dev/null`
- **Input data:** Source code file.
- **Output data:** Text with vulnerabilities found.
- **Technology limitation:**
 - CIL operates after preprocessing. If you need to see comments, for example, you cannot use CIL. But you can use attributes and pragmas instead. And there is some support to help you patch the include files before they are seen by the preprocessor. For example, this is how we turn some `#defines` that we don't like into function calls
 - CIL does transform the code in a non-trivial way. This is done in order to make most analyses easier. But if you want to see the code e1, e2++ exactly as it appears in the code, then you should not use CIL
 - CIL removes all local scopes and moves all variables to function scope. It also separates a declaration with an initializer into a declaration plus an assignment. The unfortunate effect of this transformation is that local variables cannot have the `const` qualifier
- **Maturity:** stable.
- **Dependencies:** ocaml, perl
- **Documentation:**
 - doc directory of the C Code Analyzer tarball.

- **License:** FreeBSD license.
- **Metrics:**
 - Potential security problems detection.
 - Memory leak detection.
 - Multiple/dangling free detection.
 - Array out of bound accesses.
 - Potential bufferoverflow detection.

A.12 calltree

- **URL:** <http://cdrecord.berlios.de/old/private/index.html>
- **Author:** Jörg Schilling <schilling at fokus.fhg.de>
- **Description:** The calltree command parses a collection of input files (assuming C syntax) and builds a graph that represents the static call structure of these files. Calltree is similar to cflow but unlike cflow, calltree is not based on lint. Calltree implements some more functions than cflow, but does not list the return types of the functions. This is because calltree includes an own C parser and thus may be used even on systems that don't have lint. The disadvantage is that the C parser that is used by calltree is not completely correct and may not find all calls of a function. This is mainly true for calls that are done via function pointers. Calltree is able to detect recursive function calls (e.g.functions that call themselves). Recursive function calls are marked with an ellipsis in the output.
- **Technologies:** C.
- **Execution mode:**
 - `calltree [calltree_options] [cpp_options] file1..fileN`
- **Input data:** Source code files.
- **Output data:** Graph representing the static call structure of the files.
- **Maturity:** stable.
- **Dependencies:** No dependencies found.
- **Documentation:**
 - README file in the tarball.
- **License:** GNU General Public License (GPL).

A.13 Cinclude2dot

- **URL:** <http://www.flourish.org/cinclude2dot/>
- **Author:** Darxus <schilling@fokus.fhg.de>
- **Description:** Generates a graph of the `#include` relationships within a C source tree using graphviz.
- **Technologies:** Perl.
- **Execution mode:**
 - `cinclude2dot.pl > source.dot`
 - `neato -Tps source.dot > source.ps`
- **Input data:** Source code files.
- **Output data:** `.dot` file for input into graphviz.
- **Maturity:** stable.
- **Dependencies:** Graphviz : <http://www.graphviz.org>
- **Documentation:**
 - <http://www.flourish.org/cinclude2dot/cinclude2dot.1> (man page)
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Number of includes per C file
 - Graph containing all dependencies because of `#include` director.

A.14 codeanalyzer

- **URL:** <http://www.codeanalyzer.teel.ws>
- **Author:** Mark Teel <mark at teel.ws>
- **Description:** Code Analyzer is a software source file metrics application. Metrics calculated include total files (for multiple file metrics), total lines, code lines, comment lines, whitespace lines, average line length, code lines/file (for multiple file metrics), comment lines/file (for multiple file metrics), code/comments ratio, code/whitespace ratio, and code/(comments + whitespace) ratio. It includes built-in comment and file extension definitions for: C, C++, Java, HTML, and assembly.
- **Technologies:** Java (NetBeans 4.1 and JSE 1.5).
- **Execution mode:**
 - `java -jar CodeAnalyzer-0.7.0.jar`
- **Input data:** A set of code written in C, C++, Java, HTML or assembly.
- **Output data:** Text, CSV or HTML reports.
- **Maturity:** stable.
- **Dependencies:** JRE 1.5.
- **Documentation:**
 - User's guide:
 - * <http://www.codeanalyzer.teel.ws/docs/help.html>
 - CodeAnalyzer Java Doc:
 - * <http://www.codeanalyzer.teel.ws/javadoc/index.html>.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Total files (for multiple file metrics).
 - Total lines.
 - Code lines.
 - Comment lines.
 - Whitespace lines.
 - Average line length.
 - Code lines/file (for multiple file metrics).
 - Comment lines/file (for multiple file metrics).
 - Code/comments ratio.
 - Code/whitespace ratio.
 - Code/(comments + whitespace) ratio.

A.15 cstor

- **URL:** <http://www.visualco.de>
- **Author:** Andreas Spindler <info at visualco.de>
- **Description:** cstor is a cross-reference compiler and reverse engineering tool. It compiles one text file (database) from multiple source modules (currently reads C, C++, and Perl). The tool combines reverse engineering capabilities, code validation, and an HTML documentation generator. The database can then be reused from own scripts and programs, e.g. to implement cross-compilers, validate coding conventions, build statistics etc. The built-in documentation generators are actually an application of the database.
- **Technologies:** C/C++, Perl, sh.
- **Execution mode:**
 - `cstor [OPTIONS]`
- **Input data:** Source code files.
- **Output data:** Output data:
 - lib create library file (enabled by default).
 - HTML run the built-in HTML documentation generator.
 - Text reserved.
 - LaTeX reserved.
 - man reserved.
 - texinfo reserved.
 - DocBook reserved .
- **Maturity:** stable.
- **Dependencies:** No dependencies found.
- **Documentation:**
 - Reference Manual: <http://www.visualco.de/cstor/cstor.html>.
 - Users Manual: <http://www.visualco.de/cstor/cstor-manual.html>.
- **License:** Open Software License.
- **Metrics:**
 - Class Hierarchy index.
 - Class maps.
 - Global Declarations index.
 - Include dependencies.
 - Member declarations.

- Total declarations.
- Number methods.
- Number attributes.
- Number classes.
- Number classes.
- Number unions.
- Number enums.
- Number typedefs.
- Number functions.
- Number namespaces.
- Number objects.
- Preprocessor messages per file.

A.16 CVS Statistics Generator

- **URL:** <http://freshmeat.net/projects/cvstat>
- **Author:** Juli Mallett <jmallett at freebsd.org>
- **Description:** CVStat is a Perl script which reads information from the 'cvs log' command output and generates some statistics and prints them in HTML, for display on the Web.
- **Technologies:** Perl.
- **Execution mode:**
 - **Edit:** `update.sh` with de directory where cvsstats can find the project im interested.
 - **Execute:** `update.sh`
- **Input data:** CVS check out of a project.
- **Output data:** HTML file with statistics.
- **Maturity:** alpha.
- **Dependencies:** No dependencies found.
- **Documentation:**
 - README file within tarball.
- **License:** Public Domain.

A.17 CvsGraph

- **URL:** <http://www.akhphd.au.dk/~bertho/cvsgraph>
- **Author:** B. Stultiens <cvsgraph at akhphd.au.dk>
- **Version:** 1.6.1
- **Description:** CvsGraph is a utility to make a graphical representation of all revisions and branches of a file in a CVS/RCS repository. It has been inspired by the 'graph' option in WinCVS, but I could not find a stand-alone version of this graph code.
- **Technologies:** C.
- **Execution mode:**
 - cvsgraph [options] <file>
 - * i.e: `cvsgraph -r /home/to/repository -m module -o mygraph.png myfile.c`
- **Input data:** CVS repository available in the filesystem.
- **Output data:** Image with the generated graph.
- **Maturity:** stable, active.
- **Dependencies:** yacc, libgd2.
- **Documentation:**
 - <http://www.akhphd.au.dk/~bertho/cvsgraph>.
 - <http://www.akhphd.au.dk/~bertho/cvsgraph/README> file.
- **License:** GNU General Public License (GPL)

A.18 DEPS

- **URL:** <http://deps.alioth.debian.org>
- **Author:** Yann Dirson <ydirson at altern.org>
- **Version:** 0.13
- **Description:** DEPS is a toolkit to extract internal dependencies from a "project", manipulate the dependency graph in arbitrary ways, and produce graphs to help in managing your project, including locating abusive `#includes`.
The current features are:
 - Supported languages: C/C++ (complete), Perl5 (partial).
 - Supported transformations: transitive reduction, regexp-based grouping, consolidation of multiple group levels.
 - Supported styling operations: group-based coloring, edge annotation.
 - Supported graph renderers: GraphViz (usable), Tulip (preliminar).
- **Technologies:** Perl.
- **Execution mode:**
 - `graph-includes [options] <src_files> [ch]`
* i.e: `graph-includes examples/graphincluds/project/wesnoth.pm`
- **Input data:** C/C++ or Perl source files.
- **Output data:** `.dot` graph by default (also includes support for jpg images).
- **Maturity:** alpha.
- **Dependencies:** Perl, libset-object-perl (`Set::Object`)
- **Documentation:**
 - <http://deps.alioth.debian.org/manual/index.html>
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Includes between files.
 - Detecting abusive `#includes`.

A.19 pylint

- **URL:** <http://www.logilab.org/projects/pylint>
- **Author:** Alexandre Fayolle <alexandre.fayolle at logilab.fr>
- **Version:** 0.3.3
- **Description:** Pylint is a tool that checks for errors in python code, and tries to check that a given coding standard is respected by the coders. This is similar but nevertheless different from what pychecker provides, especially since pychecker explicitly does not bother with coding style. Pylint will display a number of errors and warnings as it analyzes the code, as well as some statistics about the number of warnings and errors found in different files. If you run pylint twice, it will display the statistics from the previous run together with the ones from the current run, so that you can see if the code has improved or not.
- **Technologies:** Python.
- **Execution mode:**
 - `pylint [options] <module/package>`
 - * i.e: `pylint pythonmodule`
- **Input data:** Python module.
- **Output data:** HTML or raw text.
- **Maturity:** beta, inactive.
- **Dependencies:** Python, logilab, Optik.
- **Documentation:**
 - doc directory of the package.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Stylistic verification.
 - Coding standard enforcements.
 - Number of the encountered problems.
 - Severity of the encountered problems.
 - Improvements between versions can be checked.

A.20 Eclipse Metrics Plugin

- **URL:** <http://sourceforge.net/projects/metrics>
- **Author:** Frank Sauer <sauerf at users.sourceforge.net>, Uwe Kubosch <donv70 at users.sourceforge.net>, G.B. <gboissier at users.sourceforge.net>, tuBrody <tubrody at users.sourceforge.net>, William Keehn <williamkeehn at users.sourceforge.net>
- **Version:** 1.3.6
- **Description:** Provide metrics calculation and dependency analyzer plugin for the Eclipse platform. Measure various metrics with average and standard deviation and detect cycles in package and type dependencies and graph them.
- **Technologies:** Java.
- **Execution mode:**
 - Run Eclipse and install the plugin from <http://metrics.sourceforge.net/update>.
 - To start using the Metrics View use **Windows -> Show View -> Other** and navigate to the **Metrics View**.
 - Collect metrics for a project, right click on the project and from the popup menu select **"Metrics->Enable"**.
- **Input data:** Java Eclipse project.
- **Output data:** Metrics view or XML file.
- **Technology limitation:**
 - Only works with projects written in Java.
- **Maturity:** stable, inactive.
- **Dependencies:** Java, Eclipse 3.1.
- **Documentation:**
 - <http://metrics.sourceforge.net>.
- **License:** Eclipse Public License.
- **Metrics:**
 - Depth of Inheritance Tree (DIT).
 - Number of Overridden Methods (NORM).
 - McCabe Cyclomatic Complexity.
 - Weighted Methods per Class (WMC).
 - Lack of Cohesion of Methods (LCOM).
 - Afferent Coupling (Ca).
 - Efferent Coupling (Ce).
 - Instability (I).
 - Abstractness (A).
 - Normalized Distance from Main Sequence (Dn).

A.21 Perl Metrics

- **URL:** <http://www.dunhackin.org/eisen/perl-metrics>
- **Author:** Hal Eisen <[eisen at dunhackin.org](mailto:eisen@ dunhackin.org)>
- **Version:** 0.04
- **Description:** This program is intended to help perl programmers write better code by becoming more aware of their coding style. In particular, one would like to know the code-to-comment ratio, the average number of lines per subroutine, the longest subroutine, and things like that.
- **Technologies:** Perl.
- **Execution mode:**
 - `perl-metrics.pl [-s] file [file1..fileN]`
 - `* perl-metrics.pl -s perl-metrics.pl`
- **Input data:** Files written in Perl.
- **Output data:** Text statistics.
- **Technology limitation:**
 - Only implemented basic metrics.
 - Only implemented metrics for Perl code.
- **Maturity:** alpha, inactive.
- **Dependencies:** Perl.
- **Documentation:** No documentation found.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Code with comments.
 - Subroutine lines.
 - Blank lines.
 - Subroutines.
 - Comment lines.
 - Pure code.
 - Total lines.
 - Non-subroutine lines.
 - Code-to-comment ratio.
 - Avg. lines per subroutine.
 - Longest subroutine.

A.22 MetricsAnalyzer

- **URL:** <http://metricsanalyzer.sourceforge.net>
- **Author:** Vesa Karvonen <vesa_karvonen@hotmail.com>
- **Version:** 02.12.03-1
- **Description:** MetricsAnalyzer is a simple tool designed to help and encourage collecting and analyzing software metrics. MetricsAnalyzer stores metrics into a SQL database and allows the user to write SQL queries to analyze the stored metrics.
- **Technologies:** Java.
- **Execution mode:**
 - `java -cp "metricsanalyzer.jar:minijslisp.jar:javautils.jar:your JDBC driver" metricsanalyzer.gui.Main`
- **Input data:** XML files obtained using JavaNCSS.
- **Output data:** SQL.
- **Technology limitation:**
 - Only implemented basic metrics (empty lines, source lines, comments, etc).
 - Only implemented metrics for Java code.
 - The code that collects metrics from the Importers and stores them to the database is probably rather slow.
 - Query visualization and export is missing.
- **Maturity:** alpha, inactive.
- **Dependencies:** Java 1.4, JavaNCSS, JDepend.
- **Documentation:**
 - <http://metricsanalyzer.sourceforge.net> (documentation section)
- **License:** GNU General Public License (GPL)
- **Metrics:**
 - Basic metrics for Java. Empty lines, source lines, comments...

A.23 PyMetrics

- **URL:** <http://sourceforge.net/projects/pymetrics>
- **Author:** Reginald B. Charney <rcharney at users.sourceforge.net>
- **Version:** 0.7.6
- **Description:** PyMetrics produces metrics for Python programs. Metrics include McCabe's Cyclomatic Complexity metric, LoC, Comments percent, and so on. Users can also define their own metrics using data from PyMetrics. PyMetrics outputs SQL command files and CSV output.
- **Technologies:** Python.
- **Execution mode:**
 - PyMetrics [options] file1 ...
 - * PyMetrics examples/sample.py
- **Input data:** Source files written in Python.
- **Output data:** SQL, CSV or text with the metrics obtained from the files.
- **Technology limitation:**
 - The application does not allow to specify the level of granularity of the measurements.
- **Maturity:** stable, inactive.
- **Dependencies:** python.
- **Documentation:** No documentation found.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - blockCount.
 - maxBlockDepth.
 - numCharacters.
 - numComments.
 - numCommentsInline.
 - numDocStrings.
 - numFcnDocStrings.
 - numFunctions.
 - numKeywords.
 - numLines.
 - numModuleDocStrings.
 - numMultipleExitFcns.

- tokenCount.
- percent of Comments.
- percent of CommentsInLine.
- percent of FunctionsHavingDocStrings.
- functions DocString present or missing.
- McCabe complexity metric.
- COCOMO 2's SLOC metric.

A.24 The Metrics (for Python)

- **URL:** <http://sourceforge.net/projects/pythonmetric>
- **Author:** annelih <annelih at users.sourceforge.net>, Anders Storsveen <bushwakko at users.sourceforge.net>, Morten Svendsen <mlsvendsen at users.sourceforge.net>, Rune E. J. <runeerle at users.sourceforge.net>, Thomas Oesterlie <zortoaster at users.sourceforge.net>
- **Version:** 1.0.
- **Description:** This program will calculate and output metrics on code written in Python.
 - Metrics with different levels of granularity : class, function, module
 - Reports can be generated in text or XML-files.
 - A plug-in system lets new metrics be added to the program.
 - Includes metrics like cyclomatic complexity, Chidamber-Kemerer, Henderson-Sellers, ...
- **Technologies:** Python.
- **Execution mode:**
 - `python Main.py [options] file/dir`
 - * i.e: `python Main.py TestingFiles`
- **Input data:** Source files written in Python.
- **Output data:** XML or text files with the metrics calculated.
- **Maturity:** stable, inactive.
- **Dependencies:** python.
- **Documentation:** No documentation found.
- **License:** GNU General Public License (GPL)
- **Metrics:**
 - Average CC of the functions.
 - Cyclomatic complexity (not in sub classes/functions).
 - Function with the highest CC.
 - CC of the function with lowest CC.
 - The cyclomatic complexity for entire sub tree.
 - Full class and function path of class/function.
 - Returns the classname of this class.
 - Returns the filename of the module (file).
 - Returns the path of the module (file).
 - An test plugin only, and not an metric.

- Returns the functionname of this function.
- Lack of Cohesion in Methods, Chidamber-Kemerer.
- Lack of Cohesion in Methods, Henderson-Sellers.
- Returns the modulename.
- Average NLOC in a classes.
- Average NLOC in functions.
- Maximum NLOC in classes.
- Maximum NLOC in functions.
- Minimum NLOC in classes.
- Minimum NLOC in functions.
- Total NLOC inside the given node.
- The number of classes.
- Number of functions in a module, class or function.
- Number of imports for a module.
- An internal test plugin.
- Makes an list of all found AST node types..
- Total number of classes.
- Total number of functions.

A.25 Pymmetry - Python Trust Metrics

- **URL:** <http://sourceforge.net/projects/pymmetry>
- **Author:** Luke Kenneth Casson Leighton <lkcl at users.sourceforge.net>
- **Version:** 1.0.
- **Description:** Trust Metrics with the implementation of Ford-Fulkerson Maximum Flow algorithm and mod_virgule's Trust Metrics code.
- **Technologies:** Python.
- **Maturity:** stable, inactive.
- **Dependencies:** Python.
- **Documentation:**
 - README files within the package.
 - Trust Metrics Evaluation Project
 - * <http://moloko.itc.it/trustmetricswiki/moin.cgi/TrustMetricsEvaluationProject>
- **License:** GNU General Public License

A.26 CodeMetrics

- **URL:** <http://sourceforge.net/projects/codemetrics>
- **Author:** Eric M. Klein <klein.eric at gmail.com>
- **Version:** 0.1.1.
- **Description:** CodeMetrics is a flexible and extensible Perl package to count lines stats from code. This application counts the number of lines and classified them by types: comments, code and blank. Currently the VB.Net, Perl and C++ languages are supported.
- **Technologies:** Perl.
- **Execution mode:**
 - `codeMetrics.pl [-h] [-l language] [-s folder] [-o output] [-f file]`
 - * i.e: `codeMetrics.pl -l perl -s CodeMetrics`
- **Input data:** Source files written in Perl, Visual Basic .NET or C++.
- **Output data:** Text column, CSV or HTML file with the stats.
- **Maturity:** beta, inactive.
- **Dependencies:** perl.
- **Documentation:** No documentation found.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Total lines.
 - Comments.
 - Lines of code.
 - Blank lines.

A.27 RSM - Resource-Standard-Metrics

- **URL:** <http://msquaredtechnologies.com>
- **Author:** Resource-Standard-Metrics
- **Version:** 6.94.
- **Description:** RSM is a source code metrics and quality analysis program for the languages of C, C++, Java and C#. Some of its features are described below:
 - Windows/Unix application.
 - Metrics with different levels of granularity : class, function, method, interface, module, project.
 - Calculates LOC/SLOC, eLOC, lLOC, cyclomatic complexity, inheritance tree, inheritance children and depth.
 - Integrates with Visual Studio, .NET, JBuilder, Eclipse and other popular IDEs.
- **Technologies:** Unknown
- **Execution mode:**
 - `rsm.lnx [options] files/dirs`
 - * i.e: `./rsm.lnx -c -fa -r"h,cpp,java project/"`
- **Input data:** Projects or source files written in C/C++, Java or C#.
- **Output data:** HTML, CSV or text repor files.
- **Technology limitation:**
 - Proprietary License.
 - The shareware version only generates stats of 10 files.
- **Maturity:** stable, active.
- **Dependencies:** No dependencies found.
- **Documentation:**
 - <http://msquaredtechnologies.com/m2rsm/docs/index.htm>
- **License:** M Squared Technologies LLC Software/Source Code License
- **Metrics:**
 - Project:
 - * rolup statistics on file, classes and methods.
 - * LOC/SLOC, eLOC, lLOC.
 - * comments and physical lines.
 - Package:

- * LOC/SLOC, eLOC, lLOC.
- * comments and physical lines.
- File:
 - * number of classes.
 - * methods statistics.
 - * LOC/SLOC, eLOC, lLOC.
 - * comments and physical lines.
- Interface:
 - * LOC/SLOC, eLOC, lLOC.
 - * comments and physical lines.
- Class:
 - * public, protected and private methods.
 - * public, protected and private data.
 - * inheritance tree, inheritance children and depth.
 - * LOC/SLOC, eLOC, lLOC.
 - * comments and physical lines.
- Method:
 - * base class inheritance.
 - * template identification.
 - * namespace.
 - * class scope.
 - * cyclomatic complexity
 - * overall complexity.
 - * LOC/SLOC, eLOC, lLOC.
 - * comments and physical lines.

A.28 aopmetrics

- **URL:** <http://aopmetrics.tigris.org>
- **Author:** Michal Stochmialek <misto at e-informatyka.pl>
- **Version:** 0.3.
- **Description:** The goal of the aopmetrics project is to provide a common metrics tool for the object-oriented and the aspect-oriented programming. The project aims to provide following features:
 - Chidamber and Kemerer metrics suite.
 - Robert Martin’s metrics suite.
 - Support for Java (also 1.5) and AspectJ (also 1.5) languages.
- **Technologies:** Java.
- **Execution mode:**
 - Write a config file with the files to analyze. Each file per line.
 - Execute the command 'aopmetrics.sh'
 - * i.e: `aopmetrics.sh -configfile config.lst -workdir build/work -resultsfile results.xml`
- **Input data:** Source files written in Java and AspectJ.
- **Output data:** XML and XLS files.
- **Maturity:** unstable, inactive.
- **Dependencies:** Java 1.4.
- **Documentation:**
 - Metrics: <http://aopmetrics.tigris.org/metrics.html>
 - Execution: <http://aopmetrics.tigris.org/cmdline.html>
 - javadoc api in the package.
- **License:** Common Public License
- **Metrics:**
 - Simple metrics:
 - * lines of Class Code (LOCC).
 - CK metrics suite for AOP:
 - * weighted Operations in Module (WOM).
 - * depth of Inheritance Tree (DIT).
 - * number Of Children (NOC).
 - * crosscutting Degree of an Aspect (CDA).
 - * coupling on Intercepted Modules (CIM).
 - * coupling on Advice Execution (CAE).

- * coupling on Method Call (CMC).
 - * coupling on Field Access (CFA).
 - * coupling between Modules (CBM).
 - * response For a Module (RFM).
 - * lack of Cohesion in Operations (LCO).
- Package dependencies:
- * number of Types (NOT).
 - * abstractness (A).
 - * afferent Couplings (Ca).
 - * efferent Couplings (Ce).
 - * modified Efferent Couplings (Ce).
 - * instability (I).

A.29 Delphi Code Analyser

- **URL:** <http://dca.sourceforge.net/phpwiki>
- **Author:** Joerg Wuethrich <chewiebug@users.sourceforge.net>
- **Version:** 0.1.31.
- **Description:** Delphi Code Analyser calculates some metrics for Delphi source code and verifies the implementation against a logical architectural description of the project.
The project also has the architectural audit option that checks the implementation for correct dependencies compared to a logical description of the software architecture.
- **Technologies:** Pascal/Delphi.
- **Execution mode:**
 - `dca.exe [options] files`
 - * i.e: `dca.exe -p commandline/dca.pr`
 - * i.e: `dca.exe -a architecture/dca.architecture.xml`
- **Input data:** Source file with all the includes and for architectural audit a XML file with the architectural specification.
- **Output data:** XLS files.
- **Maturity:** alpha.
- **Dependencies:** Windows, Delphi 7.1 Professional, DUnit 9.0.5, want 0.3.2, pasdoc 0.8.8.
- **Documentation:**
 - Summary:
 - * <http://dca.sourceforge.net/phpwiki/index.php/DelphiCodeAnalyser>
 - Metrics:
 - * <http://dca.sourceforge.net/phpwiki/index.php/MetricsImplemented>
 - ArchitecturalAudit:
 - * <http://dca.sourceforge.net/phpwiki/index.php/ArchitecturalAudit>
- **License:** Mozilla Public License 1.1 (MPL 1.1).
- **Metrics:**
 - Simple metrics:
 - * the metrics in this category have a mere statistic value.
 - * number of operations (NOO) : counts the number of operations (including constructors, destructor) of a class.
 - * number of attributes (NOA) : counts the number of attributes of a class.

- Cohesion:
 - * cohesion metrics are concerned with the interdependency inside a class
 - * lack of cohesion between methods (LCOM) *calculates the percentage of methods that don't use an attribute, averaged over all attributes
- Coupling: coupling metrics measure the interdependence between classes
- Coupling between objects (CBO) : CBO is defined as the number of non-inherited classes who are associated with target class
- JDepend
 - * efferent coupling (JDCe): The number of other packages that the classes in the package depend upon is an indicator of the package's independence.
 - * afferent coupling (JDCa): The number of other packages that depend upon classes within the package is an indicator of the package's responsibility.
 - * abstractness (JDA): The ratio of the number of abstract classes (and interfaces) in the analyzed package to the total number of classes in the analyzed package. The range for this metric is 0 to 1, with A=0 indicating a completely concrete package and A=1 indicating a completely abstract package.
 - * instability (JDI):The ratio of efferent coupling (Ce) to total coupling (Ce + Ca) such that $I = Ce / (Ce + Ca)$. This metric is an indicator of the package's resilience to change. The range for this metric is 0 to 1, with I=0 indicating a completely stable package and I=1 indicating a completely instable package.
 - * distance to the main sequence (JDD) : The perpendicular distance of a package from the idealized line $A + I = 1$. This metric is an indicator of the package's balance between abstractness and stability. A package squarely on the main sequence is optimally balanced with respect to its abstractness and stability. Ideal packages are either completely abstract and stable (x=0, y=1) or completely concrete and instable (x=1, y=0). The range for this metric is 0 to 1, with D=0 indicating a package that is coincident with the main sequence and D=1 indicating a package that is as far from the main sequence as possible.

A.30 Pythius

- **URL:** `http://pythius.sourceforge.net`
- **Author:** Jürgen Hermann <jh at web.de>, Frank J. Tobin <ftobin at neverending.org>
- **Version:** 1.4.
- **Description:** Pythius is a set of tools to assess the quality of Python code. This is commonly done by applying different code metrics. Simple code metrics are the ratio between comments and code lines, module and function size, etc.
- **Technologies:** Python.
- **Execution mode:**
 - `python pystats [options] files/dirs`
 - * i.e: `python pystats.py TestingFiles`
- **Input data:** Source files written in Python.
- **Output data:** XML or text files with the metrics calculated.
- **Technology limitation:**
 - Only provides information about number of lines, blank lines and comments per module and functions.
- **Maturity:** beta, inactive.
- **Dependencies:** python.
- **Documentation:** No documentation found.
- **License:** GNU General Public License (GPL)
- **Metrics:**
 - Number of lines.
 - Blank lines.
 - Comments per module.
 - Functions.

A.31 Perfometer

- **URL:** <http://alexvn.freesevers.com/s1/perfometer.html>
- **Author:** Alex Vinokur <alex.vinokur at gmail.com>
- **Version:** 2.9
- **Description:** The program enables to get performance of C/C++ program and separated pieces of code for any metrics (for instance : time, memory, metrics defined by user etc.). The measurement results are represented in detailed/summary reports.
The user may set various parameters in order to control the measurement/comparison process :
 - measurement report and detailed measurement report flags.
 - total tests.
 - total iterations.
 - measurement scale.
 - confidence threshold.
- **Technologies:** C/C++.
- **Execution mode:**
 - Add includes "time.h" and "sys/time.h" into the source code.
 - Compile the program with make.
 - Execute `perfo.exe -x`.
- **Input data:** No input data defined.
- **Output data:** log with results
- **Maturity:** stable, inactive
- **Dependencies:** gcc
- **Documentation:**
 - <http://alexvn.freesevers.com/s1/perfometer.html>
- **License:** GNU General Public License (GPL)
- **Metrics:**
 - Provides global metrics (number of functions, etc).
 - For each function it makes an analysis (average run cost, description, name, ...).

A.32 CodeAnalyzer - Multi-Platform Java Code Analyzer

- **URL:** <http://www.codeanalyzer.teel.ws>
- **Author:** Mark Teel <mteel at users.sourceforge.net>
- **Version:** 0.7.0.
- **Description:** Code Analyzer is a java application for C, C++, Java, Assembly, HTML and user-defined software source metrics. It calculates metrics across multiple source trees as one project. It has a nice tree view of the project with flexible report capabilities.
- **Technologies:** Java.
- **Execution mode:**
 - `java -jar CodeAnalyzer-0.7.0.jar`
- **Input data:** Set of source files written in C/C++, Java, Assembly and HTML.
- **Output data:** HTML, CSV or text file.
- **Maturity:** stable, inactive.
- **Dependencies:** Java 1.4 or Java 1.5.
- **Documentation:**
 - User's Guide: <http://www.codeanalyzer.teel.ws/docs/help.html>
 - JavaDoc: <http://www.codeanalyzer.teel.ws/javadoc/index.html>
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Code lines.
 - Blank lines.
 - Comment lines.
 - Average Line Length.
 - Code Lines/File (For multiple file metrics).
 - Comment Lines/File (For multiple file metrics).
 - Code/Comments Ratio.
 - Code/Whitespace Ratio.
 - Code/(Comments + Whitespace) Ratio.

A.33 CAP - Code Analysis Plugin

- **URL:** <http://cap.xore.de>
- **Author:** Johannes Schneider <info at johannes-schneider.info>
- **Version:** 1.2.0.
- **Description:** CAP is a plugin for the eclipse platform and analysis the dependencies of your Java project. It opens a own perspective and displays the results in an clear way using different diagrams. Showing the weaknesses in your architektur, will help to improve the encapsulation. As a result your application will have a maximized reusability and a much better maintainability.
- **Technologies:** Java.
- **Execution mode:**
 - Use the plugin installed in Eclipse.
- **Input data:** Java Project.
- **Maturity:** stable.
- **Dependencies:** Eclipse 3.0, JDK 1.4, JFreeGraph, GEF.
- **Documentation:**
 - <http://cap.xore.de/cap.php?show=cap.doc.04> (in german).
- **License:** GNU General Public License (GPL).

A.34 cvsstats

- **URL:** <http://cvsstats.sourceforge.net>
- **Author:** Dan Weeks <danimal@users.sourceforge.net>
- **Version:** 0.2.
- **Description:** The CVS Log Stats Parser (cvsstats) is a suite of tools to gather information on a checked out CVS source tree. Stats are gathered globally and on a per-user basis with the generation of text and graphic reports.
- **Technologies:** Perl.
- **Execution mode:**
 - `cvsstats -cvsdir <dir> -outdir <dir> [-label <string>]`
 - * i.e: `cvsstats.pl -cvsdir cvs/checkout -outdir output -label "My CVS Stats"`
- **Input data:** CVS checkout.
- **Output data:** HTML file.
- **Technology limitation:**
 - Old and limited application. Use CVSanaly instead.
 - Poor statistics.
- **Maturity:** stable.
- **Dependencies:** Perl, Perl Date::Calc
- **Documentation:** No documentation found.
- **License:** FreeBSD License

A.35 cccc

- **URL:** <http://cccc.sourceforge.net>
- **Author:** Tim Littlefair <tim_littlefair at hotmail.com>
- **Version:** 3.1.4.
- **Description:** CCCC is a tool which analyzes C/C++ and Java files and generates a report on various metrics of the code.
- **Technologies:** C/C++, Java.
- **Execution mode:**
 - `cccc [options] file1 ...fileN`
 - * i.e.: `cccc test/prn11.cc`
- **Input data:** Source files written in C/C++ or Java.
- **Output data:** HTML, XML or database file with the metrics obtained from the files.
- **Technology limitation:**
 - Older versions included support for Ada83 and Ada95 languages.
- **Maturity:** stable, inactive.
- **Dependencies:** gcc
- **Documentation:**
 - `readme.txt` and `changes.txt` included in the package.
- **License:** GNU General Public License (GPL).
- **Metrics:**
 - Procedural metrics include lines of code, lines of comment and McCabe's cyclomatic complexity.
 - Metrics of OO design (proposed by Chidamber and Kemerer) include depth of inheritance tree, number of children, coupling between objects, and weighted methods per class.
 - Structural metrics (based on the work of Henry and Kafura), include fan-in, fan-out and information flow.

A.36 LOCC

- **URL:** <http://csdl.ics.hawaii.edu/Tools/LOCC>
- **Author:** Mike Pauldin <mpauldin at hawaii.edu>, Joe Dane <jdane at hawaii.edu>, Philip Johnson <johnson at hawaii.edu>
- **Version:** 5.1
- **Description:** LOCC is an extensible system for producing hierarchical, incremental measurements of work product size that are useful for estimation, planning, and other software engineering activities. Supports size measurement of grammar-based languages through integrated support for JavaCC. Currently, LOCC comes with a prebuilt parser for Java, plus an experimental parser for C++. LOCC can also measure the size of non-grammar-based languages (such as simple ASCII text). Supports incremental size measurements. In other words, given two files containing successive versions of Java source code, it can perform a hierarchical "diff" of the two files, producing the number of new and changed packages, classes, methods, and lines of code within each method.
- **Technologies:** Java.
- **Execution mode:**
 - LOCC Total: `java -cp locc-all.jar csdl.locc.sys.LOCTotal [arguments]`
 - * i.g: `java -cp build/locc-all.jar csdl.locc.sys.LOCTotal -sizetype javaline -srcdir src .java`
 - LOCC Diff: `java -cp locc-all.jar csdl.locc.sys.LOCDiff [arguments]`
 - GUI LOCC: `java -cp build/locc-all.jar csdl.locc.sys.LOCC`
- **Input data:** Source files written in Java or C++.
- **Output data:** CSV, XML or plain text with data about size measurements.
- **Technology limitation:**
 - To analyze java source written in JDK 1.5 is needed to use LOCC 3.3 due to JDK 1.5 includes the new reserved word 'enum'.
- **Maturity:** stable.
- **Dependencies:** Java JDK 1.4 or later.
- **Documentation:**
 - <http://csdl.ics.hawaii.edu/Tools/LOCC/dist/doc/userguide/userguide.html>.
 - <http://csdl.ics.hawaii.edu/Tools/LOCC/dist/README.html> (links to other documentation files) .
- **License:** GPL, Java2HTML License, Apache Software License, IBM Public License, Sun Binary Code License Agreement with Supplemental Terms.

- **Metrics:**

- Number of packages.
- Number of classes in each package.
- Number of methods in each class.
- Number of lines of code in each method.
- Differences between two versions of code (previously analyzed).

A.37 OSSMole

- **URL:** <http://ossmole.sourceforge.net>
- **Author:** Megan Conklin, James Howison, Kang-ming Liu and Kevin Crowston: <[ossmole-discuss at lists.sourceforge.net](mailto:ossmole-discuss@lists.sourceforge.net)>
- **Description:** OSSmole is a SourceForge.net mining project which aims to provide data and reports about open source projects and teams, provide scripts for analyzing the OSSmole raw data and provide a Java API so that you can gather your own data from web projects.
In addition OSSmole provides a web front-end to make queries the database project.
- **Technologies:** Java, SQL.
- **Execution mode:**
 - LOCC Total: Web front-end: sql queries.
- **Input data:** Data from open source projects and teams mainly stored in forges: SourceForge.net, Freshmeat, RubyForge. etc.
- **Output data:** Classified data stored into a database and raw data from projects.
- **Maturity:** stable.
- **Dependencies:** Java.
- **Documentation:**
 - Web site : <http://ossmole.sourceforge.net>.
 - Javadoc API : stored in ossmole-tools package.
- **License:** GNU General Public License (GPL)

A.38 Koch 2004 SourceForge/CVS + SourceForge

A.38.1 Usage

For any analysis of open source software development and its enactment in virtual communities, information on a large number of projects is necessary. Therefore, SourceForge.net, the well-known and largest software development and hosting site, is a logical source of data. The mission of SourceForge.net is 'to enrich the open source community by providing a centralized place for open source developers to control and manage open source software development'. To fulfil this mission goal, a variety of services is offered to hosted projects, including tools for managing support, mailing lists and discussion forums, web server space, shell services and compile farm, and source code control. While SourceForge.net publishes several statistics, e.g. on activity in their hosted projects, this information is not detailed enough for some analyses.

One important property of open software development is the efficiency of this model. Software engineering has dealt with the problem of estimating the effort for a software project for decades and has produced a multitude of methods to this end [Koc05a]. For effort estimation, the COCOMO ¹ is an algorithmic model, which is used in software development for the costing and expenditure estimation. COCOMO is model using [Sof07a]: the underlying cost estimation equations, every assumption made in model, is based on a precise definition of the product design phase of a project. There are also other possibilities to determine effort estimation. One is software equation by Putnam. It is based on the Norden/Rayleigh function and generally known as a macro estimation model for large projects [Sof07b].

Productivity, creativity and efficiency are key in the software development. These functionalities can be measured, using either LOC or Function Points due to different programming language. We propose the method of Data Envelopment Analysis DEA. DEA is a technology for efficiency and productivity analysis from economic- and development-research. Efficiency and productivity in development reproduce output measure using LOC. In that economical research most as an analysis appears. The famous analysis is function points (FP) method. FP is evaluating a system capabilities from a user point view. Analysis of the FP completely depends on the user in 5 phases [Q/P07]:

- Data functions
 1. Internal Logical Files
 2. External Interface Files
- Transactional functions
 1. External Inputs
 2. External Outputs
 3. External Inquiries

¹COConstructive COSt MOdel

For the excellent analysis of the FP and efficiency for output measure commercial software is not well suitable e.g. problematic. On the other hand the problem from free- & open-software projects are unknown the effort invested. Therefore at the free software project is to be estimated which is not the case in the commercial project. The necessary factors into this projects are in case of each also estimated which speak the dispatching between team are widened e.g. inestimably. The large advantages in the comparison with that commercial software is that it is free and usably selectable which can test it at any time in differently member team, committed, feedback gives, bug of report to provide, further for programming suitably and upgrading.

Production of software projects can be defined as follows:

- production function which define possible output for a given amount of input [Var05].

The main concept of DEA is object analysis with observance Decision Making Unit (DMU). DMU includes relatively flexibly each unit which is responsible for the transformation of inputs into outputs [Kocnt]. The basic idea of DEA is a global generalization of efficiency evaluation. The measure between DEA and DMU is relationship either from input to output or from output to input. DEA measures production behavior directly and uses this data for the evaluation of all DMU.

Each DMU is used with weighted positive linear combination from all inputs and outputs. The weights are defined such as to maximize the production relationship of the examined unit, in order to let these become as efficient as possible [Kocnt]. For each DMU the DEA supplies a variable array of weighting factor and an DEA efficiency score. If the result is one, then the DMU is DEA efficient. For inefficient DMUs weighting factors were found in the context of the selected model variant which resulted in a higher efficiency value in the case of at least another DMU [Kocnt].

Overall, this software and resulting data have been used in [Koc04, Kocnt, Koc05a, Koc05b], and the methodology itself has been published in [HK05].

A.38.2 Input

The input is the maximum id number of SourceForge.net the program should proceed to. This is used for downloading the relevant web pages.

Algorithm A.38.1: INPUT(*Input*)

```

$url = "http://sourceforge.net/cvs/?group_id=" . $m;
my $req = new HTTP::RequestGET => $url;
my $res = $ua => request($req);

if ($res => is_success) {
    open (DATA, "> /home/src/sourceforge/projects/$m.html");
}

```

A.38.3 Process

We present the retrieval of data from SourceForge with a large number of different open source projects, used information stored by the available and open to the public software development. Especially the source code control system offered, in the form of Concurrent Version System CVS, a free system which is being used extensively in the free system which is being used extensively in the free software community [K.F99], was used as the main source of information [HK05]. The CVS is the free source code control system used in free open source collective. The CVS with information on the web page is hosted on the SourceForge.net, what cause more facilities for users. See also the following figure, for a schematic of the download of data.

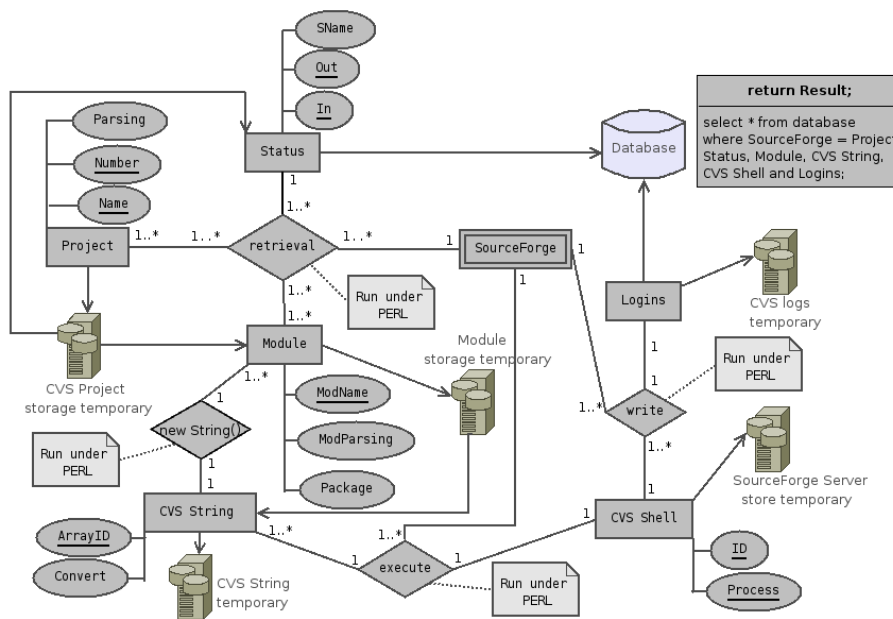


Figure A.1: SourceForge and CVS

This model with retrieval process divides at several parts on. These parts are accomplished with each other as:

1. Project
2. Module
3. Status
4. CVS String
5. CVS Shell
6. Logins

The SourceForge as web server has differently projects and ordered with large number of currently projects therefore designated we as Weak Enty (WE).

SourceForge with WE accepts different primaries attributes and relationships as identification projects. As the first SourceForge with Relationship *retrieval* tried to get a list of project which the CVS services enabled. This can be done for all project, if the CVS server hosted at SourceForge. CVS Provider necessarily account data of each Project therefore the data to correct positions to be supplied. It recognized each project with its primary- and identified attributes of project. These attributes are: **Name** and **Number**. This comes in one Web Page retrieved per Project. That web page parsed with CVS services as long as both tasks are settled. The development process runs under programming language Perl. We present how the representation of arbitrary project can be identified.

Algorithm A.38.2: GETPROJECT(*username, res*)

```

comment: username (ua) and password (pwd) are known
$ua = new ua: UserAgent;
$ua → agent("OpenSourceSoftware" . $ua → agent);

for ($m = 21300; $m ≤ 35000; $m++) {
    $url = "http://sourceforge.net/cvs/?groupid=" . $m
    my $req = new HTTP :: Request GET ⇒ $url;
    my $res = $ua → request($req);

    if ($res → issuccess) {
        open (DATA,
            " > /home/source/sourceforge/projects/$m.html");
        print DATA $res → content;
        close DATA;
    }
    else {
        open (ERROR,
            " >> /home/source/sourceforge/projects/errorlog");
        print ERROR "Couldn't get list $url";
        close ERROR;
    }

    print $url;
}

```

We show the parsing project an next step:

Algorithm A.38.3: PARSING(*project*)

```
$datadir = " home/source/sourceforge/projects"  
my@dlist = grep(!/:.?$/, readdir(IDIR))  
  
foreach $file (@dlist) {  
    $currentfile = $file;  
    print "Parsing $file";  
  
    if (!open (LIST, "< $datadir/$file")) {  
        logerror ("Couldn't open file $file");  
        next;  
    }  
}
```

As the CVS interface can only handle statements concerning the modules of which a project is composed [HK05]. It is to defined a CVS interface necessarily for each modules certain name. This name SourceForge determines with the assistance relationship *retrieval* selected primary attributes. This attribute is ModName. The web page can call the different pages-browsing with the CVS repository retrived for each project in the module storage is temporary.

Algorithm A.38.4: GETMODULES(*ua*)

```
open (PROJECTS,  
      " < /home/source/sourceforge/projects/projectlist.work);  
$ua = newLWP :: UserAgent;  
  
while (< PROJECTS >) {  
  
    if (/.*cvsroot(.*)$/) {  
        $url =  
            "http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/" . $1 . "/";  
        my $req = new HTTP :: Request GET => $url;  
        my $res = $ua -> request($req);  
        if ($res -> issuccess) {  
            open (DATA,  
                  " > /home/source/sourceforge/modules/$1.html");  
            print DATA $res -> content;  
            close DATA;  
        }  
        else{  
            open (ERROR,  
                  " >> /home/source/sourceforge/modules/errorlog");  
            print ERROR "Couldn't get list $url";  
            close ERROR;  
        }  
  
        print $url;  
    }  
}
```

SourceForge has a status indicator assigned to each project. This source received with Perl script for downloading the web pages. The result status is saved in the database.

Algorithm A.38.5: GETMODULES(*ModName*)

```
while (< HTML >) {
  if(/.* DevelopmentStatus :
    .* () - [<]+ < A >< BR >< LI > .* /)
  {
    $status = $1;
    $ok = 1;
    $insertstring = "INSERT INTO project
                    VALUES('$project', '$status')";
  }

  if ($ok == 0) {
    $insertstring = "INSERT INTO project
                    VALUES('$project', '0')";
  }

  print $insertstring;
  $dbh → do ($insertstring) || logerror("Databaseerror".
    $dbh → errstr."at".scalar (localtime (time)));
  close HTML;
}

open (MISSPROJ, " < $missingprojects");
$dbh = DBI → connect ("dbi : mysql : database = opensource",
  "Database", "") ||
  inform($DBI :: errstr);

while (< MISSPROJ >) {
  if (/.* listhttp : .* /([/]+)$/) {
    $insertstring = "INSERT INTO project
                    VALUES ('$1', '0')";
    print $insertstring;
    $dbh → do ($insertstring) ||
    logerror("Database error".
      $dbh → errstr."at".scalar(localtime(time)));
  }
}

close MISSPROJ;
$dbh → disconnect;
```

Using CVS data from temporary module storage with relationship *new String()* compose with Perl program for querying the CVS String.

Algorithm A.38.6: CVSSTRING(*ArrayID*)

comment: Number 1

```
echo Starting Project enlightenment
mkdir enlightenment
cd enlightenment
echo Retrieving and Logging Project enlightenment / Module e16
cvs -d :pserver :anonymous@cvs.enlightenment.sourceforge.net
/cvsroot/enlightenment co e16
cvs -d :pserver :anonymous@cvs.enlightenment.sourceforge.net
/cvsroot/enlightenment log e16 >> ../enlightenment.log
```

Executing this resulting large script results both in the downloaded source code and an output log file for each project [HK05]. The CVS log command recommended the history of all file in the module. CVS String with relationship *execute* gives the shell scrip command, like for example a message:

Algorithm A.38.7: CVSSHELL(*ID, Process*)

```
: pserver : anonymous@cvs.enlightenment.sourceforge.net :
/cvsroot/enlightenmentA
```

The log files from CVS produces the whole history from all files of CVS String temporary. A file, as identified by a filename and a directory path (which is necessary as some filenames are duplicates, e.g. a file named *makefile* exists in several directories) can be checked into CVS by a programmer [HK05]. The information from log file has stored in the database. CVS log files find CVS logs temporary.

The result of whole operational sequence of the CVS process develop as a SQL query. This query read as:

Algorithm A.38.8: DATABASE(*resQuery*)

```
$statement = "SELECT filename, projectname,
COUNT(*), SUM(locplus), SUM(locminus),
COUNT(distinctprogrammername),
MIN(date), MAX(date)
FROM checkin GROUP BY filename";
```

A.38.4 Output

The output is a database which can be accessed using simple SQL queries. An SQL query reads for example:

```
"select file_name, project_name, count(*),  
sum(loc_plus),sum(loc_minus),count(distinct programmer_name),  
min(date),max(date) from checkin group by filename;
```

The result would then for example give:

1. accessserver;thaig;35427;194
2. beepcore-tcl;jkint;188
3. caudium;redax;3090;43
4. dbdom;iter;57938;98
5. embermud;zak0;265242;893

A.39 Koch 1999 Gnome/CVS+Email

A.39.1 Usage

GNOME Project includes GNOME, GNU Network Object Model Environment and open source project building a desktop environment. This project includes a set of standard desktop like GNU Image Manipulation Program (GIMP), Abi-Word, GnomeMeeting, GNOME Bug Report, GNOME-Apt, GNOME Log Viewer, GNOME Network Tool, GNOME System monitor, GNOME Terminal and Gnome Control Center. GNOME is part of the GNU Project and is Free Software.

The main data source was the source code control system, in the case of GNOME project the most widely used one in the open source community CVS [K.F99]. The database of CVS controls change to the code with commit by a reconstruction of prior states. Especially the modification of source code and file identification are stored with large security by all commits. Files and directories can be checked in to CVS-System by a development. The CVS-System checked the changes in LOC and to transfer the information. The information is taken from CVS-repository.

In order to access CVS-archives in a more convenient way the Mozilla project developed Bonsai which allows connections using a web-based interface [Fie99]. The web interface of the CVS-repository as offered by Bonsai was used to retrieve the necessary data concerning checkins [Koc05a]. This done by using a Perl script. Perl script generates queries in a web browser input. Each query is stored in history file of CVS archive starting with earliest entries of the project. The result of each individual query was a HTML page which was subsequently parsed extracting the necessary attributes conforming to the data model [Koc05a]. In the GNOME project, 301 programmers were identified, who differ significantly in their effort for this project [Koc05a]. GNOME project uses data on project participation, not on the output results. It based on metric for inputs to the software development process. The number of people usefully employed at any given time is assumed to be approximately proportional to the number of problems ready for solution at that time(τ), represented by the difference between total effort(K) and accumulated effort to this date($C(\tau)$) in person-years [Koc05a]. We can represent this result as follows:

$$\frac{dC(t)}{dt} = p(t) \cdot [K - C(t)] \quad (\text{A.1})$$

The learning rate of the team is modeled as a linear function of time in years since project start ($p(\tau)$)

$$p(t) = 2 \cdot at$$

which governs the application of effort [Koc05a]. Manpower-function($m(\tau)$) give time a Rayleigh-type ² curve governed by a parameter(a) which plays an important role in the determination of the peak manpower [Koc05a]. $m(t)$ after differentiation $C(t)$ in in the certain time t follows:

$$m(t) = 2 \cdot at \cdot [K - \exp^{-1} \cdot t^2] \quad (\text{A.2})$$

²Rayleigh-type is the Rayleigh-Norden model assumes full-time employees, this number needs to be adapter, or else the results would also be scaled to open source developer-years [Koc05a].

$$m(t) = 2 \cdot K \cdot at \cdot \exp(-at^2)$$

By deriving the $m(t)$ relative to the time and finding the zero value, the relationship between time of peak manning and this parameter can be found [Koc05a]. The value of the peak connoted can be derivation by substitution this term in the $m(t)$. Manpower distribution for the GNOME project will demonstrated on the following figure.

This figure illustrates the number of active programmers in each month, can be

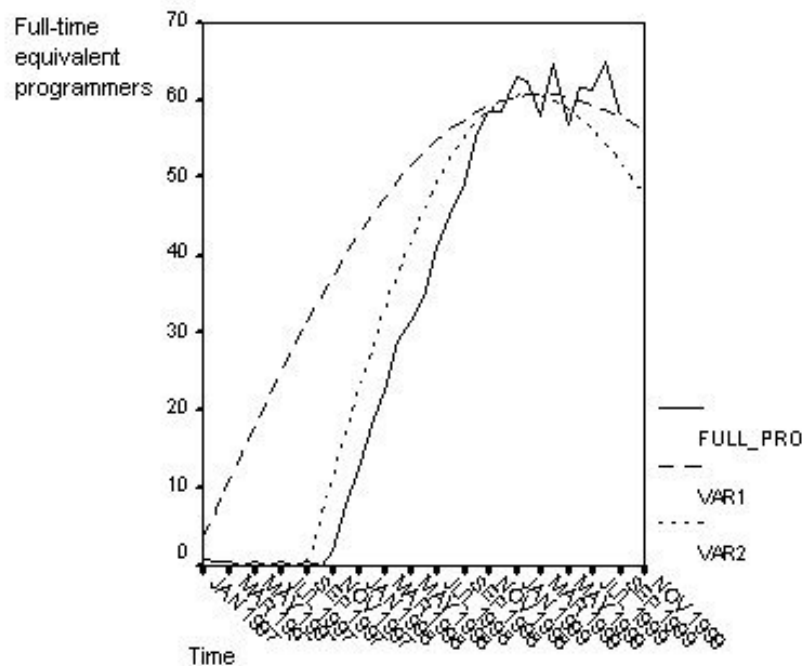


Figure A.2: Manpower Function from dataFULL-PRO and projectedVAR1 & VAR2 [Koc05a]

retrieved from the data(FULL-PRO) and seems to follow a Rayleigh-type curve at least until a certain point, this information can be used for estimating the effort [Koc05a].

A.39.2 Input

There is actually no input to this program, it was developed solely for the GNOME project and therefore has the few parameters (like mailing list or CVS repository addresses) hardcoded.

A.39.3 Process

After downloading the CVS logs using Bonsai and its web interface using Perl, this process describes parsing files in the CVS area.

Algorithm A.39.1: CVSFILTER(*CVSfilter*)

```
my@history;
$errorlogfile = "/usr/logins/src/parseerrorlog";
$datadir = "/usr/logins/src/cvslog";
opendir (IDIR,$datadir) or die "Couldn't open masterdirectory";

my @dlist = grep (readdir (IDIR));
close (IDIR);

foreach $file (@dlist) {
    $current_file = $file;
    print "Parsing $file";

    if (!open (CVS," < $datadir/$file")){
        log_error ("Couldn't open file$file");
        next;
    }

    my $content;
    while (< CVS >) {
        $content. = $_;
    }

    close CVS;
    undef (@history);

    my $insertstring =
        "INSERT INTO checkin
        VALUES ('$file_name','$path',
                 $short_pname','$checkin_date',
                 '$revision_number','$loc[0]',
                 '$loc[1]','$description')";

    print $insertstring;
    $dbh → do($insertstring) ||
    log_error ("Database error". $dbh → errstr." at"
              .scalar (localtime (time)).);
}
```

Database recreates data with the "INSERT" statement. Some statement can look as:

Algorithm A.39.2: STATEMENT(*Statement*)

```

my $insertstring =
    "INSERT INTO chekin
    VALUES ('$file_name','$path',
             $short_pname','$checkin_date',
             $revision_number','$loc[0]',
             '$loc[1]','$description')";

```

Give the defined data from the URL and print data as a result.

Algorithm A.39.3: GETPOSTING(*GetPosting*)

```

@listname = ("gnome - list", "gnome - devel - list",
             "gnome - announce - list", "gnome - components - list",
             "gnome - gui - list", "gnome - kde - list",
             "gnome - themes - list",
             "gnumeric - list", "guppi - list", "balsa - list",
             "gnome - pilot - list", "gnome - doc - list", "calendar - list",
             "gnome - i18n", "gnome - mailer - list", "wm - spec - list",
             "gnome - packaging - list", "gnome - db - list", "gtk - list",
             "gtk - app - devel - list", "gtk - devel - list",
             "gtk - doc - list", "gtk - i18n - list");

@year = (1999, 1998, 1997);

@month = ("January", "February", "March", "April", "May",
          "June", "July", "August", "September",
          "October", "November", "December");

$list_url = "http://www.gnome.org/mailling - lists/archives/" .
            $listname[$m]." /" . $year[$i]." - " . $month[$j]." /";

my $req = new HTTP :: RequestGET => $list_url;
my $res = $ua -> request($req);
if ($res -> is_success) {
    open (DATA,
          "> /usr/logins/open - source/postings/${listname[$m]}
          -${year[$i]}-${month[$j]}.html");
    print DATA $res -> content;
    close DATA
}

```

Inserting the mailinglist in the database.

Algorithm A.39.4: INSERTDATA(*Insertdata*)

comment: INSERT DATA INTO DATABASE

```
$current_file = /(.*)-(.*)-(.*)/;
my $mailinglist = $1;
my $filename =
    "http://www.gnome.org/mailling - lists/archives/" .
    $mailinglist . "/" . $2 . "/" . $3;
my $insertstring =
    "INSERT INTO full_posting
    VALUES ('$filename', '$content')";
print $insertstring;
```

A.39.4 Output

As output of the program can as data base SQL regard. SQL statement reads the following:

Algorithm A.39.5: INPUT(*Input*)

```
$dbh = DBI → connect ("dbi : Pg : dbname = open", "", "") ||
inform ($DBI :: errstr);

$statement = "SELECT project_name, date_trunc ('year', date),
date_trunc ('month', date),
COUNT(DISTINCT programmer_name)
FROM checkin
GROUP BY project_name, date_trunc ('month', date)
AND date_trunc ('year', date)";

$cursor = $dbh → prepare($statement);
$cursor → execute;

while (@row = $cursor → fetchrow_array) {
    print join (";", @row). "n";
}

$cursor → finish;
```

A.40 Iskenderoglu 2006 Email

A.40.1 Usage

A mailing list is a collection of names and addresses used by an individual or an organization to send material to multiple recipients [Wik07b]. At least two quite different types of mailing lists can be defined: the first one is closer to the literal sense, where a mailing list of people is used as a recipient for newsletters, periodicals or advertising [Wik07b]. Traditionally this was done through the postal system, but with the rise of e-mail, the electronic mailing list became popular [Wik07b].

At larger number of receivers programs are used, which pass messages on, which were sent to a central place, to the receivers. This is done either automatically or after release via a moderator, which examines the message after its contents. Mailing lists can also read off-lines and will work on, what a substantial difference to a forum represents. Some mailing lists differ from each other in fact that only announced users of write rights may have and only read "guests". This characteristic is by the administrator of the mailing list or by the unity valley lungs of the server on that the mailing list is stored. With very large projects, with more than 50 participants, which with most open source projects the case is, one can win different information, which can be helpful for later analyses from their mailing lists.

A.40.2 Input

The input is the relevant SourceForge mailing list to be analysed. The program gets an input as the first argument by the COMMANDS console, i.e. the URL of the mailing list. With "GET" method of mechanize the web pages are downloaded.

A.40.3 Process

The analysis integration consolidate on basis conditions:

- Source data
- Extraction
- Transformation
- Data management
- Reporting

Source data concerned with different Mailing List Archive and external source archive. Extraction is interface between software the program and source data. Major task is the procurement of the Mailing list e.g temporary save and downloading. Transformation has two Task. First task is examined and converted the stored data after their integrity. Second task relay the data on to the data management. Data management stores the data in a SQL database, which will serve as central data base for analysis activities and query selection. Reporting to demonstrate tabular and graphic GUI of the results. The program tries to log with the SQL data base. After a successful registration first a data base is

provided.

Algorithm A.40.1: INIT_DATABASE(*username, password*)

```
sub init_database{  
  
    #Database connection  
    db_mysql = DBI → connect (  
        "DBI : mysql : host = 127.0.0.1",  
        $username,  
        $password,  
        {RaiseError ⇒ 1}  
    );  
  
    #Generate database  
    eval{ $db_mysql → do (  
        "CREATE DATABASE".  
        $database"  
    );  
    $db_mysql → do ("USE" . $database);  
}
```

Afterwards two tables are put on. The first table has four columns. The first column "email" is the email address of the transmitter, "senders" is the name of the transmitter, "date" date of the email, ThreadID is the number of the Threads. The second table has three columns, list (=Name of the mailing list), Thread (Subject) and ThreadID.

We know the table to compile.

This code provide a table for the Mailinglist volumes.

Algorithm A.40.2: CREATETABLEI(*CreateTableI*)

```
eval { $db_mysql → do (  
    "CREATE TABLE" . $table .  
    "(Email VARCHAR(256) NOT NULL,  
    Absender VARCHAR(256) NOT NULL,  
    Date DATETIME,  
    Threadid BIGINT(20),  
    INDEX (Email), UNIQUE (Date))  
    )  
};  
  
print "Kann keine Tabelle $table erstellen";
```

Algorithm A.40.3: CREATE TABLE II (*CreateDatabaseII*)

```
eval {$db_mysql → do (  
    "CREATETABLE" . $idtable .  
    "(Liste VARCHAR(64) NOT NULL,  
    Thread VARCHAR(256) NOT NULL,  
    Threadid BIGINT(20) NOT NULL  
    AUTO_INCREMENT,  
    INDEX(Threadid))  
    ")  
};  
print "Kann keine Tabelle $idtable erstellen";
```

This provide the table for Maillisten topics.

Algorithm A.40.4: CREATE TABLE (*mailingtopics*)

```
eval {$db_mysql → do (  
    "CREATE TABLE" . $table.  
    "(List VARCHAR(256) NOT NULL,  
    Thread VARCHAR(256) NOT NULL,  
    Threadid BIGINT(20) NOT NULL  
    AUTO_INCREMENT,  
    INDEX(Threadid))  
    ")  
};
```

Why we need two tables?

We need two tables, so that we can differentiate between the projects, which have in their mailing list two Threads of the same name. Most projects on SourceForge have the mailing lists more than one side. A side has maximally twenty five Threads. Examination whether the Topics are present on one side. The correct examination passes by call `nextmessage`.

`$nextmessage --;`

This piece of program gets argument as an input by the command console, i.e. the URL of the mailing list.

Algorithm A.40.5: WEBSITE(*url*)

```
#Call parameters of the function defined.my ($url) = @_;  
#Provide a Mechanize object  
my $mechanize =  
    WWW :: Mechanize → new(autocheck ⇒ 1, quiet ⇒ 0);  
  
#The side loads down  
$mechanize → ($url);
```

With get method of Mechanize the web pages are downloaded. Mechanize collects all www, which are on the downloaded web pages. With find.link method filter we a certain Links, i.e. the link to the URL identical is.

Algorithm A.40.6: FINDLIST(*url*)

```
#Find the name of list  
my $linkName = $mechanize → find.link(n ⇒ 2, url_abs ⇒ url);  
  
#Set the names, if it is empty  
if ($listname = /$/) {  
    $listname = $linkName → text();  
}
```

Fills the array Reference with the list of the Topics from the web page

Algorithm A.40.7: ARRAY-REFERENCE(*topics*)

```
my $linklist =
    $mechanize → find_all_links(tag ⇒ "a",
                                url_regex ⇒ qr/. * thread_id.*);

#Exist link "nextMessages"
my $nmlist =
    $mechanize → find_all_links(tag ⇒ "a",
                                url_regex ⇒ qr/. * offset.* /);

my @nml = @{$nmlist};
if ($nml[0]) {
    $nextmessages ++;
}

#Array is filled with WWW : Mechanize :: Link object
#Each object is used with $onelink
my $onelink;
foreach $onelink (@{$linklist}) {
    parse_topic ($onelink, $mechanize);
}
}
```

This code describe the variable \$urlstr is defined, which has the Absolute URL of the Website as value. The contents of these URL are stored in the \$mech → content. With my \$content = \$mech → content (format ⇒ "text") to change HTML code the url in textformat.

Algorithm A.40.8: SUB_PARSE_TOPIC(*topics*)

```
my ($link, $mech) = @_;
my $urlstr = $link → url_abs();
$mech → get($urlstr);

my $content = $mech → content(format ⇒ "text");
my $dir = tempdir(CLEANUP ⇒ 1);
print $fh $content;
seek $fh, 0, 0;

while(< $fh >) {
    my @grepped = grep/SourceForge.net $listname/, $_;
    if (exists $grepped[0] && $grepped[0] ! /$/) {
        my ($blah, $parsestr) = split(/From /, $grepped[0], 2);
    }
}
}
```

The function `addto_database` gets the variables of the parse function as parameter. It is a new variable `$id` to create. After the arrays `@name` and `@subject` are converted with `sprintf(Format String)` into `String`. `Word for word` is used next to each other contents of the array and stored as `String`.

Algorithm A.40.9: `ADDTO_DATABASE(parse_topic)`

```

my $subjectstr = " ";
foreach my $t (@{$refsub}) {
    $subjectstr = sprintf("%s%s", $subjectstr, $t);
}

my $namestr = "";
foreach my $u (@{$refname}) {
    $namestr = sprintf("%s%s", $namestr, $u);
}

```

Call parameter of the function are used:

```
my ($url) = @_;
```

Provide a mechanize object

```
my $mech = WWW::Mechanize -> new( autocheck => 1, quiet => 0);
```

The side download

```
$mech -> get($url);
```

The variables `$email`, `$date`, `$time`, `@subject` and `@name` are passed on as parameters of the function `addto_database`, which in the next step into a stringer is then converted, is `@subject`. Same applies also to `@name`. At the end the temporary file (file acts) is closed.

The function `addto_database` gets the variables `parse_topic` of the function as parameter. In addition a new variable is produced `$tid` (thread ID). Afterwards the arrays `@name` and `@subject` are converted with `sprintf` (format stringer) into stringers.

A.40.4 Output

Tools in this program are concentrate exclusively on the mailing lists of the projects on SourceForge.net. The first Tool "Mailinglist.pl" is able to download the web pages with the emails temporally and needed information from these web pages to extract. It the name of the mailing list, name and E-Mail address of the participants, date of the enameles and its IDs in a SQL were based data base stored. The second Tool 'Analysis.pl' has as major task to accomplish inquiries at the data base and to represent the results both graphically and in the title format. Analysis.pl produces also CSV files, which have the time (months) and the number in these months of sent emails as columns.

A.41 Aksu 2006 Bugzilla

A.41.1 Usage

Till mid-sixties the operating system was included in every computer for free. In 1965 IBM stopped it with the help from the Non-Disclosure-Agreements. Richard Stallman from Massachusetts Institute of Technology (MIT) launched the GNU project in 1983. He was also the initiator of the Free Software Foundation organisation. The main feature of free software is the possibility of ensure software freedom for all. The name "open source" come into existence in 1998 together with the spread for Netscape Communications Corporation the source code of Netscape Navigator browser. Free software, as defined by the Free Software Foundation, is software which can be used, copied, studied, modified and redistributed with little or no restriction [Wik07a].

After the article "The Cathedral and the Bazaar" from Eric Steven Raymond, the open source attract more attention. The article describes kinds of create the software. The Open Source Initiative(OSI) allowed more than 55 licences to let to do any modifications and to share it. The most famous licences are: Apache Licence, GNU-GPL, MIT Licence, Mozilla Public Licence, Python Licence and Sun Public.

In the developing process, more than one developer works together. One of them follows the progress of development and tries to avoid and eliminate errors.

Bugzilla belonged to the most well-known Bugtracking systems, the Bugzilla implemented in Perl license stands and is thus free of charge available under the Mozilla Public. It is in the reason a CGI program. The use of the system is made by a HTML side. As data base MySQL serves. Bugzilla offers an integrated error pursuit system, which can be adapted on different system. Most frequently it is used for queues in the IT-support, system-administration and -distribution management, for error pursuit during the draft and the development by chips, as well as software and hardware error pursuit. As data base can be used MySQL or PostgreSQL and as Web servers recommend Bugzilla Apache. Additionally to Bug lists Bugzilla offers two further features: Reports (report) and diagrams (charts). With a report it concerns the current condition of the data base.

A.41.2 Input

The input is hardcoded into the program, and consists of only the server name of the bugzilla system to be analyzed.

A.41.3 Process

After downloading, the program makes an analysis and/or the allocation of the urgency. First by means of an inquiry the different severity kinds and their frequencies are determined. In this example it is tightened the amount of the bugs for each severity kind by individual inquiries, because thereby providing of a diagram is easier.

Algorithm A.41.1: SELECTQUERY(*SelectQuery*)

```
my $important = $dbh → do
(q{
    SELECT bugnumber
    FROM bug
    WHERE severity
    LIKE '%important%'
    }
);

print $important;
```

The other part of code serve to build a bar-chart diagram and named it and to save the bar-chart as a PNG graphic. The next part of code is to built the pie-chart diagram.

Now it will be as usual one bugs sender reported.

Algorithm A.41.2: WRITETABLE(*WriteTablen*)

```
open(WRITETABLE2, " > /Perl/sender.txt");

$sth = $dbh → prepare
("
    SELECT sender, COUNT(bugnumber)
    FROM bug2
    GROUP BY sender
");

$sth → execute;

while (@row = $sth → fetchrow_array) {
    print WRITETABLE2 "@row";
}

close (WRITETABLE2);
```

From this part it will come a text data *.txt like for example:
Hiroki Tamakoshi <hiroki-t@is.aist-nara.ac.jp> 1

Now the installed module is searching for the right web page, and is opening it.

This code is to extraction the data and to connect to the database. Create a table called "bug2".

Algorithm A.41.3: WRITETABLE(*WriteTablen*)

```
$dbh → do
(
    CREATE TABLE bug2 (
        bugnumber INT(11) PRIMARY KEY,
        sender VARCHAR(100),
        maintainer VARCHAR(100),
        date VARCHAR(45),
        package VARCHAR(21),
        severity VARCHAR(15)
    )
);
```

The table include six columns with different parameters. The table is saved in the variable *dbh*.

A.41.4 Output

From data which are in the table included and saved in the database the diagram is built. The input is the database. The user has to write down the user name and password. If his/her data are correct the log-in process is accepted and the user is connected to database "Bugs".

Algorithm A.41.4: INPUT(*Input*)

```
my $user = 'root';
my $password = 'merve';

$dbh = DBI → connect (
    'DBI : mysql : Bugs',
    $user, $password,
    {RaiseError ⇒ 1, AutoCommit ⇒ 1}
)
or die $DBI :: errstr;
```

Bibliography

- [Fie99] R.T. Fielding. Shared leadership in the apache project. *Communications of the ACM*, 42(4):42–43, 1999.
- [HK05] Michael Hahsler and Stefan Koch. Discussion of a Large-Scale Open Source Data Collection Methodology. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-38)*, Big Island, Hawaii, 2005.
- [K.F99] K.Fogel. *Open Source Development with CVS*. Scottsdale Arizona: CoriolisOpen Press, 1999.
- [Koc04] Stefan Koch. Profiling an Open Source Project Ecology and Its Programmers. *Electronic Markets*, 14(2):77–88, 2004.
- [Koc05a] Stefan Koch. Effort Modeling and Programmer Participation in Open Source Software Projects. Arbeitspapiere zum Tätigkeitsfeld Informationsverarbeitung, Informationswirtschaft und Prozessmanagement, Department für Informationsverarbeitung und Prozessmanagement No. 03, Vienna University of Economics and Business Administration, 2005.
- [Koc05b] Stefan Koch. Evolution of open source software systems - a large-scale investigation. In *Proceedings of the 1st International Conference on Open Source Systems (OSS 2005)*, pages 148–153, Genova, Italy, 2005.
- [Kocnt] Stefan Koch. Measuring the Efficiency of Free and Open Source Software Projects Using Data Envelopment Analysis. In Sowe S.K., Stamelos I., and Samoladas I. (eds.), editors, *Emerging Free and Open Source Software Practices*. ID Publishing, in print.
- [Q/P07] Q/P Management Group, Inc. Proprietary and Confidential. An Introduction to Function Point Analysis. <http://www.qpmg.com/fp-intro.htm>, Mon Jän 15 12:18:52 CET 2007.
- [Sof07a] Softstar Systems. Overview of COCOMO. <http://www.softstarsystems.com/overview.htm>, Mit Jän 17 08:24:10 CET 2007.
- [Sof07b] Software Measurement Page. SLIM-Software Life Cycle Management. <http://yunus.hacettepe.edu.tr/~sencer/cocomo.html>, Mon Jän 15 17:49:50 CET 2007.

- [Var05] H.R Varian. *Intermediate Microeconomics: A Modern Approach*. Seventh Edition New York, N.Y.: W.W Norton & Company, 2005.
- [Wik07a] Wikimedia Organisation. Free software. http://en.wikipedia.org/wiki/Free_Software, Mit Jän 24 08:46:19 CET 2007.
- [Wik07b] Wikimedia Organisation. Mailing list. http://en.wikipedia.org/wiki/Mailing_list, Mit Jän 24 15:26:39 CET 2007.