Alambic: An Open-Source Platform for Software Engineering Data Management The Case of Embedded Software Development

Boris Baldassari
Castalia Solutions,
10 Rue de Penthièvre,
75088 Paris
Tel: +33 648 038 289 – e-mail: boris.baldassari@castalia.solutions

Abstract:
As a relatively recent discipline, software engineering data mining has made important advances during the last years, producing new tools and methods for the assessment and improvement of software products and practices. Building upon the experience accumulated from software measurement programs, data mining takes a wider and deeper view on software projects by introducing unusual kinds of information (e.g. mailing lists, tools metadata or deployment) and addressing new software engineering concerns (e.g. community, tooling or process).
With the advent of open data principles and the dissemination of tools to manipulate and explore data, a new way of using analytics has emerged to provide domain-specific interpretation and visualisation of data assets. Part of this new challenge is to provide a consistent, unified vision on data across projects to provide adapted and context-aware answers to developers, managers, and users of projects.
This article describes Alambic, an open-source framework and service for software project data management. The platform retrieves data from various tools and repositories used in software development, presents them in a variety of meaningful ways and provides unified data access for data enthusiasts in an attempt to ease and foster the usage and interpretation of software-related data.

Key words: software repository data mining, open data, software quality, open source, business intelligence, project management.

1 INTRODUCTION

Software engineering data mining is at the crossroads of two developing fields that have tremendously advanced in the recent years. From the software engineering perspective, lessons learned from years of failed and successful software projects have been catalogued and now provide a comprehensive set of practices and empirical knowledge – although some concepts still miss a common, established consensus among actors of the field. Software measurement programs have been widely studied and debated and now offer a comprehensive set of recommendations and known issues for practitioners. On the data mining side, research areas like software repository mining [Hemmati2013], categorisation of artefacts, program comprehension [Storey2005], or developer assistance [Marri2009, Bruch2010] have brought new insights and new perspectives with regards to data available for analyses and the guidance that can be drawn from them. This has led to a new era of knowledge for software measurement programs, which expand their scope beyond simple assessment concerns. Data mining programs have enough information on hand to leverage quality assessment to pragmatic recommendations and improvement of the process, practices and overall quality of software projects.
Alambic is an open-source framework[1] and service[2] that brings these advanced software repository data mining technics to projects. It has been designed and developed by software practitioners, using state-of-the-art research technics and tools to deliver clear insights and practical advice to project teams and users. This article presents the Alambic framework and service, the quality assessment and improvement method associated, and how it helps teams gather, manage and publish software engineering data.

1 BitBucket repository: :https://bitbucket.org/BorisBaldassari/alambic.
2 Eclipse Alambic dashboard: http://eclipse.castalia.camp.

2 STATE OF PRACTICE

2.1 A word about Software Quality

If one intends to *improve software quality*, then one first needs to define it in one's own specific context. Dozens of definitions have been proposed and widely debated for software quality, from *conformance to requirements* [Crosby1979] to *fitness for use* [Juran1999], but none of them have gained a definitive acceptance among the community or the industry – because the notion of quality may vary.

Standards like ISO 9126 [ISO9126] or the 250XX series [ISO25000] propose interesting references for product quality and have been quite well adopted by the software industry. The ISO 15504 and CMMi [CMMi2010] standards provide useful guidelines for process maturity assessment and improvement at the organisational level. Kitchenham and Pfleeger [Kitchenham1996], further citing Garvin's teachings on product quality, conclude that "quality is a complex and multifaceted concept that can be described from five different perspectives":

- *The transcendental view* sees quality as something that can be recognised but hardly defined.
- *The user view* sees quality as fitness for purpose.
- *The manufacturing view* sees quality as conformance to specification.
- *The product view* attaches quality to inherent characteristics of the product.
- *The value-based view* sees quality as the interest or money users will put on it.

Even in the restricted situation of a specific software project most people will implicitly have different meanings for quality. For that reason, the requirements and notion of quality need to be openly discussed and explicitly stated. Relying on well-known definitions and concepts issued from both standards and experts in the domain greatly helps actors to reach an agreement and elaborate their own definition of quality.

2.2 Software engineering data mining

Data mining offers a new perspective on the information available from software projects, thus unleashing powerful new tools and methods for this activity. While software measurement programs allow to assess aspects of the product or project quality, software data mining programs offer enough information to build upon the assessment phase and deliver practical recommendations for the improvement of quality. In the context of software development this can be achieved through e.g. action lists for refactoring, maintainability warnings for overly complex files, or notice when too many support questions lie unanswered. Data mining methods offer a plethora of useful tools for this purpose. Outliers detection [Pachgade2012, Singh2013] gives precious information for artefacts or project characteristics that show a deviation from usual behaviour. Clustering [Naib2013] allows to classify artefacts according to multi-dimensional criteria. Recommender systems [Bruch2010, Robillard2010] are used to propose good practices, code snippets or detect bug patterns.

2.3 Software engineering data repositories

In recent years many open data platforms have been created, some of them related to software engineering. Many forges and community web sites have setup a dedicated service for people to use their data assets – **Eclipse**[3], **GitHub**[4] and **StackExchange**[5] are well-known examples. **OpenHub**[6], operated by Black Duck software, proposes basic configuration management and code metrics for more than 600,000 open-source projects. Squoring's **AgileRanking**[7] proposes a multi-dimensional quality analysis based on code, configuration management, mailing lists and issue tracking repositories. These websites are not only useful for developers, but also represent an interesting data source for the analysis of its different characteristics or attributes.

3 FROM MEASURES TO UNDERSTANDING TO ACTIONS

Considering the above-mentioned perils, we propose a few guidelines to be followed when setting up a data mining process. These allow to ensure the integrity and usability of information and keep all actors synchronised on the same concerns and solution.

3.1 Declare the intent

The whole mining process is driven by its stated goals. The quality model and attributes, means to measure it, and presentation of the results will differ if the program is designed as an audit-like, acceptance test for projects, or as an incremental quality improvement program to ease evolution and maintenance of projects. The users of the mining program, who may be developers, managers, buyers, or end-users of the product, have to map its objectives to concepts and needs they are familiar with. Including users in the definition of the intent of the program also

3 The Eclipse Dashboard: http://dashboard.eclipse.org.
4 GitHub archives: https://www.githubarchive.org.
5 StackExchange data explorer: http://data.stackexchange.com.
6 OpenHub by BlackDuck Software: http://openhub.net.
7 AgileRanking by Squoring Technologies: http://www.agileranking.com.

helps prevent counter-productive use of the metrics and quality models. The intent must be simple, clearly expressed in a few sentences, and published for all considered users of the program.

3.2 Identify quality attributes

The concerns identified in the intent are then decomposed into quality attributes. First this gives a structure to the quality model, and secondly allows one to rely on well-defined characteristics – which greatly simplifies the communication and exchange of views. Recognised standards and established practices provide useful frameworks and definitions for this step. One should strive for simplicity when elaborating quality attributes and concepts. Common sense is a good argument, and even actors that have no knowledge of the field associated to the quality attributes should be able to understand them. Obscurity is a source of fear and distrust and must be avoided. The output of this step is a fully-featured quality model that reflects all of the expected needs and views of the mining program. The produced quality model is also a point of convergence for all actors: requirements of different origin and nature are bound together and form a unified, consistent view.

3.3 Identify available metrics

Once we precisely known what quality characteristics we are looking for, we have to identify measures that reflect this information need. Data retrieval is a fragile step of the mining process. Depending on the information we are looking for, various artefact types and measures may be available: one has to select them carefully according to their intended purpose. The different repositories available for the projects being analysed should be listed, with the measures that may be retrieved from them. Selected metrics have to be stable and reliable (i.e. their meaning to users must remain constant over time and usage), and their retrieval automated (i.e. no human intervention is required). This step also defines how the metrics are aggregated up to the top quality characteristics. Since there is no universally recognised agreement on these relationships one has to rely on local understanding and conventions. All actors, or at least a vast majority of them, should agree on the meaning of the selected metrics and the links to the quality attributes.

3.4 Linking metrics to quality attributes

Many research papers have studied the correlation between quality attributes and well-known metrics, while experienced software practitioners have their own rules-of-thumb gained from years of practices in specific domains and projects. The measurement theory also adds the representational condition [Kaner2004], which can be hardly demonstrated in empirical software engineering given the huge diversity of processes, methods and constraints of software projects. As a result, these links have to be defined on a case by case basis, to match the project's specifics and the community's understanding of their relationships. In Alambic the quality model can be customised to tailor quality concerns and links to data sources for the domain and audience of the dashboard.

3.5 Provide targeted advice

Another important improvement path for software quality assessment and improvement frameworks is to provide pragmatic advice for practitioners. In a time-constrained environment, teams need to quickly understand and figure out what can be done next for better performance. Guidance should be proposed to help teams better configure the tools, point out incorrect and missing data, and identify good and bad practices.

4 ALAMBIC: CONTEXT-AWARE DATA MINING

4.1 Main requirements

Ease of use

Alambic plugins are tailored for specific tools and are really easy to setup: in most cases one simply needs to provide a project ID or URL to enable the complete analysis of a new aspect of the project. As an example the Hudson CI plugin only requires the Hudson instance URL to automatically retrieve jobs information, compute metrics, draw nice visualisations and list actions.

Centralise all software-related data in one single place

Alambic takes care of retrieving data from various repositories and data sources along the software development process, and provides a consistent, unified interface to exploit it. By using a single point of access for all software-related data, teams can easily build cross-discipline analysis and reports.

Provide relevant insights for teams

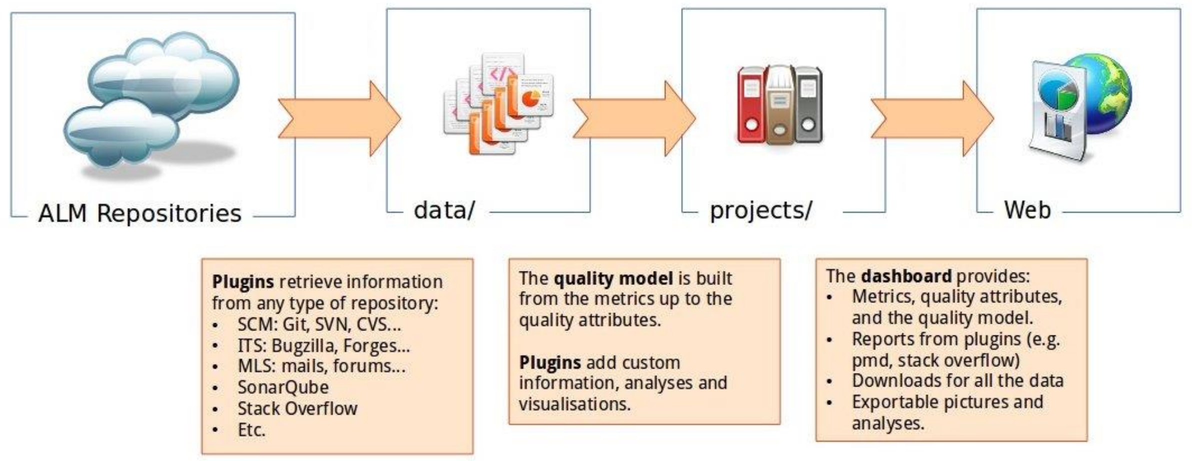
Software practitioners need practical information about how the project performs and what can be done next. Targeted at software projects, Alambic provides to-the-point lists, numbers and visualisation on useful areas of improvement for the project. As an example the Stack Overflow plugin lists recent non-answered questions that attracted a lot of votes, and common problem keywords about the product.

Provide pragmatic checks and advice

Alambic goes beyond simple reports and provides useful checks and pragmatic advice for practitioners. Many verifications and consistency checks can – and should – be run automatically and report on what is wrong and what actions should be taken to fix it. An example plugin is the Eclipse PMI plugin, which checks for incorrect entries in the process repository of the Eclipse forge and lists actions to fix them.

4.2 Architecture

Alambic uses a modular architecture based on plugins to enable the development of independent add-ons. Each plugin can retrieve data from a repository, analyse it to run automatic checks or compute derived metrics, and provide advanced visualisation.



Plugins retrieve information from any type of repository:
- SCM: Git, SVN, CVS...
- ITS: Bugzilla, Forges...
- MLS: mails, forums...
- SonarQube
- Stack Overflow
- Etc.

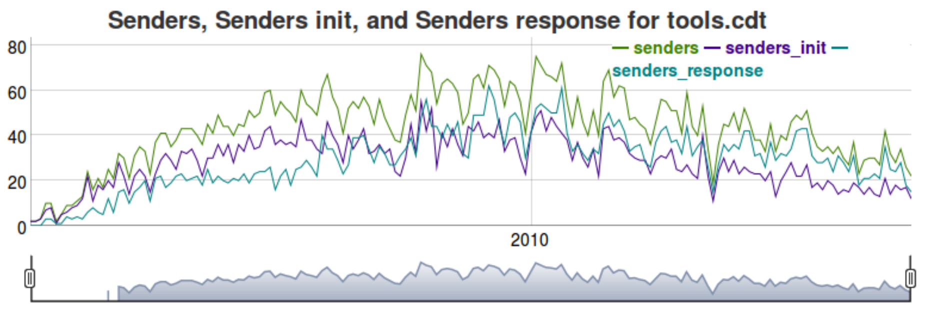
The **quality model** is built from the metrics up to the quality attributes.

Plugins add custom information, analyses and visualisations.

The **dashboard** provides:
- Metrics, quality attributes, and the quality model.
- Reports from plugins (e.g. pmd, stack overflow)
- Downloads for all the data
- Exportable pictures and analyses.

4.3 Data sources

Metrics Grimoire is an open-source library that analyses mailing lists, configuration management logs (e.g. git, svn, cvs), and issue tracking systems (e.g. bugzilla, jira). It has been developed by Bitergia (the LibreSoft spin-off).



The **PMD results analysis** plugin helps people understand the output of the tool, and proposes a strategy to fix violations step by step. The plugin also proposes some guidance to improve the tool's configuration and fine-tune the rules checked.

Projects using **Hudson** as a continuous integration engine can get insights on their build performance: amount of failing and unstable builds, jobs that should be disabled or specifically watched, and the history of builds.

The **Stack Overflow** questions and answers website is a major place to get and provide support for the open-source community. Long-standing unanswered questions and repetitive topics provide useful hints as to the problems encountered by people and the support provided by the project.

4.4 Presenting data

Open Data: make data available

One of the foundation principles of open data is the full availability of data. It has to be in an open format, should be easy to import and play with in order to foster the use and reuse of raw numbers by domain specialists. Alambic proposes a unified REST interface to access every piece of information about a project, in JSON or CSV formats. Data access is self-documented and tutorials are available to help people reuse this asset in common tools like R or Excel.

Visualisation: make data visible

Good visualisation helps people quickly grasp the essentials of a situation. Colourful, interactive and customisable graphics enable users to play, reuse and exploit data on the project. As an example, the history of non-answered questions provided by the Stack Overflow plugin offers in a single view the support history and trend of interest for the project, and highlights unanswered user concerns.

Interacting with numbers and graphics
Because users may need a different perspective on the data for custom analysis and concerns, Alambic provides several ways to further play with data and extract custom insights from it. Firstly, the data sets can be easily imported in other tools to be exploited (R, python, Excel, BI products, etc.). Secondly, most plots are built with interactivity in mind, through javascript-powered libraries well-suited for web publishing.

Disseminating data
All Alambic graphics can be exported and reused on external web sites. Most graphics include interactivity even once exported, and can be safely used by teams, auditors, and data journalists. Many data-intensive graphics use the plot.ly library [Plotly2015], which enables users to freely clone, edit and publish data and plots.

5 USE CASES

Use cases are an important part of the definition of a product: they ensure that the delivered service fulfils practical needs, is pertinent to a context, and will be actually used. Alambic was built on two main use cases, one for the embedded systems industry (PolarSys), and the other targeted at the Eclipse community at large.

5.1 The PolarSys dashboard
PolarSys is an Eclipse Industry Working Group that aims at providing a complete toolset for the development of critical embedded systems for the aeronautics, space and automotive industry. The maturity assessment task force has to evaluate the maturity of open-source software projects so they can be used in critical embedded software developments by big companies while sharing the maintenance cost of the toolset. Two main objectives have been defined for the PolarSys maturity model:

- Assess individual components to preserve the full stack maturity, and give visibility to users. From this perspective, the rating established by the measurement process acts as a quality gate for projects before entering the PolarSys umbrella and for their releases before being published.
- Help projects conform to maturity requirements, improve their practices, and better understand their developments. The process is meant to help rather than sanction teams, and allows them to keep track of their conformance on a day to day basis.

The PolarSys dashboard uses a specific quality model tailored to the domain constraints and notions of quality, and linked to custom in-house data sources.

5.2 The Eclipse forge
The Eclipse forge proposes a set of services and processes available to all projects, like milestone planning, code reviews, and IP management. These services are centrally managed and made available to the public through various means:

- Projects metadata are retrieved from the PMI (Project Management Infrastructure) through a REST API. It includes a description of the project, links to the different repositories, and release information.
- Configuration management, issue tracking systems, and mailing lists information are retrieved using the Grimoire toolset [Gonzalez2015] and are stored as JSON files on a public web server[8].
- Hudson instances for continuous integration. Hudson provides an API to get every build information.
- Marketplace information is retrieved through the web site REST API.
- Source code metrics are retrieved from the Eclipse SonarQube instance database.

The Alambic instance for Eclipse uses only public sources and features a more generic quality model. A collection of plugins provides guidance to help project improve their process, product and community performance.

6 CONCLUSION AND PERSPECTIVES

Software engineering data represents a wealth of information that needs to be correctly managed and exploited: for development teams to improve their practices, for team leaders to monitor the project's progress, for marketing teams to communicate on facts, and for end-users to assess the maturity or activity of off-the-shelf components.
The Alambic framework automates the retrieval, analysis and publishing of the various repositories used by the software project and proposes ready-to-use data sets, insights and reports for all of its actors. The Alambic service for Eclipse has received a growing interest from users and developers of the forge, and new dashboards targeting other major open-source forges like Apache and GitHub are planned before the end of the year.
We believe this interest demonstrates how open data is becoming increasingly more important for the management of software projects. Building upon this, Alambic will be improved to include new data source plugins, analysis, visualisation, and automatic reports. New use cases pertaining to other areas like system engineering and DevOps are being discussed to expand the reach of open data to new domains.

8 The Eclipse dashboard: http://dashboard.eclipse.org.

7 REFERENCES

[Hemmati2013] H. Hemmati, S. Nadi, O. Baysal, O. Kononenko, W. Wang, R. Holmes, and M. W. Godfrey. The MSR Cookbook. In *10th International Workshop on Mining Software Repositories*, pages 343–352, 2013.
[Storey2005] M.-A. Storey. Theories, methods and tools in program comprehension: Past, present and future. In *13th International Workshop on Program Comprehension* (IWPC 2005). IEEE Computer Society, 2005.
[Marri2009] M. R. Marri, S. Thummalapenta, and T. Xie. Improving Software Quality via Code Searching and Mining. In *Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation*, pages 33–36, 2009.
[Bruch2010] M. Bruch and M. Mezini. Improving Code Recommender Systems using Boolean Factor Analysis and Graphical Models. 2010.
[Crosby1979] P. B. Crosby. Quality is free: The art of making quality certain, volume 94. McGraw-Hill New York, 1979.
[Juran1999] J. M. Juran, A. B. Godfrey, R. E. Hoogstoel, and E. G. Schilling. *Juran's Quality Handbook*, volume 2. McGraw Hill New York, 1999.
[ISO9126] International Standards Organisation, Software Engineering: Product Quality, ISO/IEC 9126-1, 2005.
[ISO25000] International Standards Organisation, System and Software Quality Requirements and Evaluation, 2014.
[CMMi2010] CMMI Product Team. CMMI for Development, Version 1.3. Technical report, Carnegie Mellon University, 2010.
[Kitchenham1996] B. Kitchenham and S. Pfleeger. Software quality: the elusive target. *IEEE Software*, 13(1):12−21, 1996.
[Pashgade2012] M. S. D. Pachgade and M. S. S. Dhande. Outlier Detection over Data Set Using Cluster-Based and Distance-Based Approach. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(6):12–16, 2012.
[Singh2013] G. Singh and V. Kumar. An Efficient Clustering and Distance Based Approach for Outlier Detection. *International Journal of Computer Trends and Technology*, 4(7):2067–2072, 2013.
[Naib2013] B. B. Naib. An Improved Clustering Approach for Software Quality Analysis. *International Journal of Engineering, Applied and Management Sciences Paradigms*, 05(01):96–100, 2013.
[Robillard2010] M. Robillard and R. Walker. Recommendation systems for software engineering. *IEEE Software*, 27(4):80–86, 2010.
[Kaner2004] . Kaner and W. P. Bond. Software engineering metrics: What do they measure and how do we know? *Methodology*, 8(6):1-12, 2004
[Plotly2015] Plotly Technologies Inc., Collaborative data science, 2015. URL: https://plot.ly
[Gonzalez2015] J.M. Gonzalez-Barahona, G. Robles, D. Izquierdo-Cortazar, "The MetricsGrimoire Database Collection", *IEEE/ACM 12th Working Conference on Mining Software Repositories* (MSR), 2015.