e-Library

Share and consult documents on interoperability solutions for public administrations.

Ossmeter: a platform to automatically assess, monitor and compare OSS packages

Ossmeter: a platform to automatically assess, monitor and compare OSS packages

Submitted by Adrian Offerman on May 25, 2014

(★★★★★) 5 / 5 | 1 votes | 325 reads
|

Evaluating whether an open source software package meets the requirements for a specific application, or determining the best match from a list of packages, requires information on both the quality and the maturity of the software, as well as understanding whether the software is continuing to evolve and if there is a substantial and active community of users and developers. Furthermore, during the development/deployment lifecycle it is important to regularly re-assess and identify any risks that might emerge from a decline in the quality indicators and to monitor progress made and value created from on-going development and further contributions to the OSS package. Currently, these tasks require considerable effort by seasoned developers/architects whose time is scarce and costly.

Ossmeter is a European FP7 research project that will help users and developers of open source software to automatically evaluate and compare OSS packages and the surrounding communities involved in their evolution, utilising an extensive suite of measurements, thereby significantly reducing the time needed to perform evaluations and improve decision-making.

The Ossmeter platform is currently being developed and will be made available as a public service. In addition, the software will be published as open source, so companies can also deploy it as an in-house quality management tool for their own software or use the toolset to build or expand a consulting business.

Introduction

Ossmeter

Ossmeter is a research project aiming to further the automated analysis and measurement of the quality of open source software (OSS). Nine partners are jointly developing “a platform that will support decision makers in the process of discovering, comparing, assessing and monitoring the health, quality, impact and activity surrounding open source software. To achieve this, quality indicators will be computed by performing advanced analysis and integration of information from diverse sources including the project metadata, source code repositories, communication channels and bug tracking systems of OSS projects. That way, a new meta-platform will be created for analysing existing OSS projects that are developed in existing forges and foundations such as GitHub, SourceForge, Google Code, Eclipse, Mozilla Apache.”

Ossmeter is being developed by nine European organisations from research and industry:

- The Open Group (UK),
- University of York (UK),
- University of Manchester (UK),
- Centrum Wiskunde & Informatica (The Netherlands),
- University of L'Aquila (Italy),
- Tecnalia (Spain),

Related Content

30 September 2013 | France
French Gendarmerie: "Open source desktop lowers TCO by 40%"

01 August 2012 | Spain
News from Basque Country: Openness and Reuse of Applications Decree already approved

10 June 2014 | Spain
Extremadura health care has switched to open source

25 September 2013 | Germany
German development ministry recommends open source to SMEs

12 May 2014 | Bulgaria
Open source everywhere at Plovdiv military prosecution

- Softeam (France),
- Uninova (Instituto de Desenvolvimento de Novas Tecnologias; Portugal),
- Unparallel Innovation (Portugal).

The Open Group is the over-all coordinator and the University of York provides the technical coordination.

The project has a total cost of 3.4 million euro, of which 2.6 million is funded by the EU. It started in October 2013 and will last 30 months.

Rationale

Deploying open source software

Evaluating whether an OSS package meets the requirements for a specific application, or deciding which package from multiple candidates is the best match, is not an easy task. It requires information on the quality and maturity of the software itself, and also on how active is the ongoing development and how extensive is the user support (i.e. community activity). According to the authors of the Ossmeter project, collecting this information "involves exploring various sources of information, including source code repositories, to identify how actively the code is being developed, which programming languages are being used, how well the code is documented, whether there are tools to support testing, etc. Equally important is collecting information from surrounding communication channels such as newsgroups, forums and mailing lists to identify whether user questions are answered in a timely and satisfactory manner, and to estimate the number of experts and users of the software. In particular, its bug tracking system can be analysed to determine whether the software has many open bugs and how fast these are fixed. Other relevant metadata are, for example, the number of downloads, the licence(s) under which the software is made available, its release history, etc." In the end, this type of information is needed to make evidence-based decisions.

Furthermore, during the development/deployment lifecycle it is important to re-assess the software on a regular basis, "in order to identify and mitigate any risks that might emerge from a decline in the quality indicators of the project in a timely manner", and to monitor progress made and value created when actively participating in development.

OSS quality and support

"Open source software is most often developed in a public, collaborative, and loosely-coordinated manner," the project partners say. "That has several implications to the level of quality of specific OSS packages and to the level of support the communities provide to users of the software they produce. On the one hand, there are several high-quality and mature open source projects that deliver stable and well-documented capabilities. Such projects typically also foster a vibrant expert and user community which provides remarkable levels of support both in answering questions and in repairing reported bugs. On the other hand, there is also a substantial number of open source projects which are dysfunctional in one of more of the following ways:"

- the development team behind the open source project invests little time in its development, maintenance and support;
- the development of the project has been altogether discontinued due to lack of commitment or motivation (resulting in so-called abandonware);
- the documentation of the produced software is limited and/or of poor quality;
- the source code contains little or low-quality comments which make studying and maintaining it a problem;
- the community around the project is limited, causing questions asked by users to receive late/no response and identified defects to get repaired very slowly or ignored altogether.

"In practice the level of maturity and support of an open source project can be anywhere between two extremes, complicating the decision-making process, especially if there are several open source and commercial (i.e. proprietary) options available for the task at hand. Selecting the best solution is a very important task, because changing to another package — be it open source or commercial — after software development has commenced, may result in significant loss of investments."

Ossmeter aims to reduce "the time spent on investigating the different open source alternatives, usually by seasoned developers/architects whose time is scarce and expensive; to reduce errors, which are known to increase with the number of hard-to-compare choices; and to decrease the risk/stress involved" when making the decision to deploy open source software, which in some organisations is still perceived as a risk in itself.

Targets

These are the targets set by the project partners:

Measure:	**Target:**
quality assessment of source code	30% reduction in effort
quality assessment of the support provided through the communication channel and bug tracking system	90% reduction in effort

monitoring the evolution of an OSS project	90% reduction in effort
discovery and comparison of a number of OSS projects	90% reduction in effort

“These figures are ambitious targets,” says Scott Hansen, Director European Projects at The Open Group. “They are based on the experience of the research partners, the needs of the industrial partners and the nature of EC-funded research projects, which is to aim for substantial improvements that provide significant impact for industry. Which improvement is the most critical will depend on the specific needs of a user and whether they need to decide which OSS option to select, or have already made their choice and want to monitor the evolution of their selection.”

“The target reduction in assessing source code quality is lower than the others because there are many other factors that go into this measure beyond the quality of the source code. For example, someone providing an open source implementation of a File Transfer Protocol (FTP) could write high quality code, but not implement the international standard correctly. Ossmeter would quickly catch poor quality coding, but it doesn't try to analyse what functions the code is intended to provide or whether a package implements the functionality it claims. Once the poorly coded projects are disqualified, there is still some further work to be done in looking at the features and functions provided by the remaining candidates.”

Research objectives

Previous work

According to the project partners, “previous work in OSS analysis and measurement has mainly concentrated on quality indicators and metrics analysing aspects related to the source code. Research prototypes that have been constructed for different measurement tasks, have not been integrated into a comprehensive and widely-adopted platform that can effectively support industrial users with real-world decision making.”

“Beyond the source code, there are additional sources of information that can provide valuable indicators for the quality, maturity and health of an open source project:”

- communication channels, through which users and developers communicate with each other, in order to ask questions, exchange views, report problems, make announcements and suggest enhancements related to the software;
  an active and vibrant community is essential for candidate adopters of an OSS package as this is often the only place where they can receive expert support;
- bug tracking systems through which users formally provide defect reports and enhancement requests for the software;
  important questions include how often bug reports are submitted and how long it takes for these to be solved;
- additional metadata captured by open source software hosting forges such as number of downloads, screenshots, project plans, tags, categories, status (e.g. alpha, beta, mature), licenses under which the software is available, wikis, etc.

A comprehensive approach

“Ossmeter aims to extend the scope and effectiveness with novel contributions on both generic and language-specific methods for source code analysis,” the project partners explain, “but it also proposes using state-of-the-art Natural Language Processing (NLP) and text mining techniques. The latter includes question/answer extraction, sentiment analysis and thread clustering to analyse and integrate relevant information extracted from communication channels (e.g. newsgroups, forums, and mailing lists) and bug tracking systems.”

The project will “provide meta-models, capturing relevant meta-information on the OSS projects, e.g. types and details of source code repositories, communication channels and bug tracking systems, types of licences, number of downloads, etc.” Both information and functionality will be made available through an online portal and a set of web services, (in the proposal referred to as cloud functionality).

Supporting and monitoring tool

Consistent quality indicators will enable direct comparisons between OSS projects, according to the project partners, making the platform a valuable tool supporting:

- developers and project managers responsible for deciding on the adoption of open source software, by making available solid and uniform quality indicators;
- developers of open source software, by enabling them to monitor the projects they contribute to, to promote the value of their work and contributions, and to identify related projects;
- funding bodies, by allowing them to monitor the quality and impact of the software produced.

Finally, the project partners specifically address OSS work that is directly and indirectly funded by government organisations such as the EU, national and regional funding bodies: “An automated open source software monitoring and measurement platform will enable them to measure the quality, dissemination and user participation in software developed in the context of publicly-funded projects in an automated and consistent manner. Furthermore, it will allow funding bodies to monitor and revisit open source projects that they have funded in the past, in order to identify success stories and general trends."

Implementation

An open platform

"The software will be made available as open source," says Paul Klint, Research Fellow at CWI "so it can be run as a public service. But companies can also deploy it as an in-house service for monitoring their own software and support facilities."

In addition to the web interface, the platform will provide an API (Application Programming Interface) for access to the metrics repository, supporting Web Services, XML and JSON formats.

The platform will not support dynamic queries, however. That means users will not be able to place a request for a real-time analysis of a specific software package. "The system will process a predetermined set of packages, for example in a nightly batch job. For each package, a number of metrics will be calculated, and these will be combined into a single quality indicator for that package. We'll try to keep that final indicator as simple as possible. A five-star score, for example, might very well do the job; the simpler, the better. Users can easily compare indicators that way: a score of 4.5 is better than 3.6."

Different use cases

"The weights of the linear combination we use to calculate the over-all quality indicator from the precalculated metrics will be dynamic, however," Klint assures us. "They are different for different use cases and will be adjusted (automatically) depending on the deployment profile. So, different users will receive a different answer for the same package, depending on the importance they attach to properties like stability, security or maintainability, for example."

"Maybe we'll provide a set of sliders for them to adjust the weights. You can, for example, attach more value to the number of downloads and the number of users, or you can increase the weight of the technical maintenance characteristics."

If users want to, they can always look into the details of the score. This drill-down allows them to check the underlying metrics and weights. The same interface will also enable them to change the current profile (i.e. the weights) of their query. That allows them, for example, to investigate whether the ranking of the packages they are interested in will change if their query profile changes. Since each metric is calculated in advance, however, users will not be able to set or change parameters related to individual metrics.

Validity of the indicators

The challenge, of course, is to translate low-level metrics into high-level indicators that can be used by non-experts to evaluate software for procurement and deployment. At a business level, such decisions are often based on qualities like maturity, stability, reliability, scalability, security, maintainability, and portability. These properties are assessed in order to guarantee the achievement a minimum level, as specified in a service-level agreement for example, and to minimise risks and costs. They are, however, very hard to quantify.

"Typical source code metrics are mostly well defined and often very easy to compute," says Klint. "Some of them require only a single line of code to be specified. 'Lines of code' is a well-known example of such a basic metric." The number of lines in the source code is related to programming productivity, costs and maintainability, for example.

"Even the more complex — and interesting — metrics can often be specified on a single page. One example is McCabe's or cyclomatic complexity, which looks at the branches in the control flow to calculate the number of linearly independent paths through a program's source code. The outcome represents information on the logical complexity and cohesion of the source code. For example, are the methods in an object closely interwoven — calling each other extensively — or only loosely coupled?"

Sentiment analysis

"For Ossmeter, we process the textual content of bug tracking systems, newsgroups and mailing lists to identify the sentiments that have been expressed there," says Yannis Korkontzelos, researcher at the National Centre for Text Mining (NaCTeM) at the University of Manchester. "Depending on the type of sentiment we are looking for, a polarity score (e.g. in the range from -3 to +3) is computed. In this case, we use our sentiment analysis technology to quantify the type and quality of support for a specific OSS package, that way helping individuals and organisations to decide if this package suits their need."

"In addition to the sentiment, we are also interested in identifying the type of content of each message, e.g. is it a new request, a suggestion for a solution, a thank-you message, an announcement, etc. By combining the type and the sentiment of a message with other meta-data — e.g. the timestamps and the authors — we can design interesting and meaningful metrics like the average number of discussions per day, the number of active users on each day, and the hours/days of the week during which a user is more likely to receive an immediate response to a request."

"Of course, different people may seek different types of insight. For example, a new user might be interested in information on installing and running a software package, while a company might be interested in whether the software is supported by a large community and will be maintained in the long run. In addition, messages by a user or developer with a long history of contributing are more valuable than those of a new user. This diversity is taken into account by weighting each message according to the experience of the participant."

Extremely heterogeneous

According to Jurgen Vinju, project leader at CWI, most of the work on this project lies in bringing together all the required technologies into a single process and platform. "It all starts with the meta-models. The input we have to process and aggregate is extremely heterogeneous, for example in languages and technologies. We try to bring all of those together in an architecture that allows us to define these models, to create the tools that provide them with the correct data, and to make it all communicate with each other. That's where the biggest effort is. Building the platform is more work than implementing the metrics."

"All of the software libraries at the very bottom of this architecture are already available. Most of them are currently mature enough to be deployed by us. Of course, some research is needed to adjust the libraries brought in by our research partners for this specific purpose, but it's not fundamental. It involves incremental improvements on the knowledge and expertise of the partners, and embedding these improvements in a unified framework."

"For example, the sentiment analysis was originally designed for Pubmed [the search engine for research in life sciences and biomedical topics], so we have to make it work with a different ontology."

"So there's no risk of running into chain dependencies where the whole project can be stalled due to components that need more development time than we have assessed. In the worst case, we would simply need to scale down our ambitions for that specific functionality. But that would only affect one of the metrics, not the over-all architecture."

"Of course, there are scientific challenges, and we will be publishing articles about these. The same is true for the meta-models we have developed. Those and new research questions are the scientific deliverables of this project. But in the end we want to have a working prototype."

Expected results

Business-level indicators

The suitability of OSS software for a particular job also depends on business-level indicators that cannot be deduced by examining the source code. "That would be far too difficult," Klint explains. "A property like scalability can only be objectively determined by an actual experiment proving that a software package can be deployed on a large scale. There is no way to verify whether code is correctly and effectively multi-threading and scaling out."

"But what we can do, is to measure the sentiment of keywords related to scalability. How is this package perceived in communities of users and developers? Of course, this would be implemented as a text mining analysis on social media and collaboration/development platforms, not as a technical analysis of a software system. The same is true for the reliability and availability of a service. These are typically based on experience with deployments."

"Maintainability and portability, on the other hand, can be evaluated in a technical analysis. Maintainability metrics are readily available, and portability is closely related to the question of whether you are using standard languages and libraries."

Maturity and continuity

"The maturity of a program can be deduced from its development history," Klint continues, "incorporating the maturity of both software and community. A project that has existed for only half a year is completely different from one that has been around for ten years. Looking at the activities around a project gives you a good impression of the vitality of the community. And that again is related to continuity: can I deploy this package if I don't want to employ its developers?"

"One indicator, for example, is the time it takes for the developer community to respond to bugs. This information, e.g. how long it takes for a patch to be developed, can easily be obtained from the bug tracking system. Even more, it appears that projects which achieve a high score on this statistic also perform well on other metrics. A trend analysis looking at how these metrics develop over time gives you a good impression of the continuity of a package."

Benchmarking

"We can use well established applications like LibreOffice and Firefox to benchmark other OSS packages," says Klint. "Those packages can set a certain standard of deployability."

"With Ossmeter we'll have metrics on a lot of OSS packages, allowing us to compare these packages with one another. That allows you to get useful information on a package you're specifically interested in. Of course, you have to be very careful to not compare packages providing very different functionalities. A specialised package like a SIP server, for example, is very different from a widely used office productivity application."

"I believe comparing profiles for packages covering different areas can be interesting, but you should tread very carefully there. Furthermore, I don't think this will be the primary use case. If you're looking for an alternative e-mail client, you want to compare metrics of various OSS e-mail packages. That will give insight in the differences between alternatives providing similar functionality. Our system produces relative rather than absolute numbers."

Business case

"The business case we have in mind here is the selection of an OSS package as an alternative to the package currently used," Klint explains. "The first question, of course, is how to get an initial list of available open source alternatives. So you start searching overviews, lists and repositories already available on the Internet."

"The next step is to turn this list into a shortlist. That's where Ossmeter comes in. Giving objective metrics and insights in all sorts of properties and aspects, it will allow you to analyse and compare the packages you selected."

"In the final step you can look in detail into the packages on your shortlist. There, Ossmeter can function as a tool to explore various use cases and to analyse different profiles."

Trend analyses

In addition, keeping track of changing metrics over time allows you to analyse trends in the development of OSS packages. "Packages that are dying turn out to have certain characteristics," says Klint. "We have already observed this. That allows us to predict the chances of a package surviving, developing and maturing over time."

Such reports can also be used to make decisions on investments in OSS packages. "They can function as an early warning signal. If you're using a package that appears to have poor prospects, you may want to invest in its survival and continued development, that way securing the continuity of your own ICT processes."

"That's a very business-driven approach indeed. Companies can weigh savings in licence costs and migration trajectories on the one hand against investments in specific OSS packages on the other." Resources can be brought in jointly through a user group or a consortium with the developers, for example, or as specific contributions in source code. Reports on the development of similar packages after external investments can even give you an estimate of the resources required to reach certain levels for specific indicators.

Caveats

External data

According to Klint, three aspects complicate this aggregation of external data. First, Ossmeter brings together data from all sorts of sources: source code repositories, bug tracking systems, forums, and so on. All of this information has to be represented in a uniform way. "We have already developed the largest part of the meta-models," he says. "They will be standardised and published in research papers."

"For example, we use something we've called M3, the metrics meta-model. It consists of a small core describing dependencies between components, and is completely agnostic to the programming language these components are written in. In addition, we have extensions — already available for Java, and another one for PHP is in the making — for language-specific information. That way, we try to capture as many facts on a project as possible. The great thing about this model is that all metrics defined in the core model are completely independent of the programming language, so we can re-use these metrics for other languages and verify their validity. Still, we have to write an extension for each language we want to cover."

Data definition

Second, there is the data definition problem. Each external source uses its own definitions and generates its own information. For a lot of statistics, the exact meaning may not be clear, or it may even have changed over time. "We only use observable data from these external sources," says Klint. "But even that proves to be tricky. The number of downloads, for example, does that include downloads by the developers themselves? Because they often have to download the software in order to test it. But when you're counting actual users you'd want to filter out these IP addresses. That's why the exact meaning of external data is so important."

The Ohloh project provides another example of how difficult even simple metrics can be to define. This open source software tracker shows project statistics like lines of code, number of commits, and number of contributors. "For one of our own projects we noticed that the lines of code they stated was ridiculously high. It turned out that they added up the lines of code in all the branches of a project, including the lines that hadn't changed. So, twenty branches created during development of the same project were counted twenty times. That's a faulty way to define lines of code. Functionality of sites like Ohloh — science based — is the minimum of what we want to achieve with the Ossmeter project. And our ambition, of course, is to surpass that significantly."

"Still, for some input — the number of downloads, for example — we depend on the data provided by the repositories. We will be using the data published by the repositories for now. Later on, we may ask them for the exact meaning of their statistics. In the meantime, we do talk to the developers if we run into clear issues, like we did with Ohloh. That way we try to help each other."

API limits

Finally, Ossmeter depends on the interfaces provided by the repositories and by other trackers. "You cannot simply download everything you need," Klint explains. "These APIs come with upper limits to the number of calls you can make per day, in order to protect the availability of their services to other users. For another project, for example, we created a full copy of Ohloh; that takes a couple of months."

"That would be absolutely impossible for the GitHub and SourceForge repositories; they are huge! So we have to limit ourselves to analysing software packages of interest — to answer a specific research question, say — and the packages our users are asking for. We could offer to download and analyse packages on request as overnight jobs for them, for example."

"We will probably get in touch with these repositories at a later stage of this project. We depend on their APIs, of course. We will make sure all of our connectors work when Ossmeter is delivered, but they need to be maintained after that."

Platform

Availability

The Ossmeter platform will be available both as a free hosted service, and as an OSS package that users will be able to deploy within their own infrastructure to monitor selected OSS and proprietary internal software projects. “We'll run our analyses daily,” says Klint, “and make the results publicly available. Supporting real-time queries would require a scalable, expensive infrastructure. We will provide an open source prototype though; this is our main deliverable. So if there is a market for a more dynamic service, people will be able to set it up. But that would probably be in a business setting. In an alternative scenario, our productions will be the basis for another research project.”

“Initially, we will provide this service ourselves, to allow repositories and trackers to integrate the Ossmeter functionality into their portals, and to publish "our" metrics and indicators next to their own statistics. Currently, there is no provision for a more permanent service. First we have to prove ourselves, show some results and our value. But we don't exclude the idea that this project may eventually result in a spin-off company.”

“Ossmeter already has a commercial dimension. Our French partner and one of our Spanish partners — Softeam and Tecnalia, respectively — both offer consultancy on software procurement. They are looking to integrate our functionality into their tooling. We think this is probably a better model than setting up a new economic entity. The latter would be based on a business model where OSS repositories pay a monthly subscription fee for our services, for example, while the former is an addition to an existing toolset of a proven consulting business.”

Use case: Tecnalia's software lab

“Our ICT-ESI division works on software productivity improvement,” says Alberto Berreteaga, manager of Tecnalia's software lab. “We use various techniques, methods and tools in our R&D activities. Eclipse is one of our core technologies and open source software is one of our preferred business models. So we develop technology and commercial products using Eclipse under the open source paradigm.”

“We have brought in a lot of Eclipse-based topics and use cases in this project. We hope the Ossmeter concepts will help us to apply metrics and compare OSS projects. So, this is an opportunity to improve our technological developments and commercial actions. We plan to apply the prototype with Eclipse and use it to empower our own tools development and web presence.”

Use case: Modelio UML/BPMN modelling tool

“We are the developers of the Modelio tool, which is also available on Joinup,” says Alessandra Bagnato, project manager at the Softeam R&D department. “It's an open source UML modelling environment, supporting a wide range of standards-based functionalities, models and diagrams, and providing model assistance and consistency checking features for software developers, analysts, designers, business architects, and system architects. Its differentiating value is the fact that it supports BPMN and UML integrated into a single tool, offering dedicated diagrams to support Business Process Modelling (BPM).”

“Modelio already supports a wide range of open source modules, and is built using many OSS tools. Ossmeter will be used to monitor the health of Modelio's OSS components, so we can take corrective action in case any issues show up in a specific package. Developers can use the Ossmeter dashboard to check the indicators and evaluate the available OSS alternatives.”

“More specifically, we have selected two areas at our IT Consultancy division where Ossmeter can be deployed:”

- in the preliminary project design phase where technologies are selected according to the requirements and constraints of the project; here we decide whether specific functionality will have to be developed from scratch or if we can re-use existing commercial or OSS technology, i.e. to make or buy;
- during the project development phase or maintenance phase to monitor the selected OSS technologies in order to assure their sustainability for the project.

“The make-or-buy-decision is very important for the implementation phase of a project and for determining the final cost. Assessing the availability and the quality of existing solutions is key. So we will use Ossmeter in the decision process in projects with complex requirements for reuse/adoption of external components and tools, and as part of the internal Modelio development efforts to monitor the OSS tools used. Regarding the Ossmeter project itself, we are directly involved in the requirements stage and in the future for testing the prototype, but we will support the development team with feedback during the whole project lifetime.”

Industry partners

The online community will be involved during the development of the Ossmeter project, allowing them to “provide inputs and guidance in the early stages of the project and to participate in the evolution and further expansion of the tools towards the later stages using an open source approach.”

Although the Ossmeter software will eventually be published as open source, its initial development — which is happening now — will be a closed effort solely by the research partners. “We currently don't allow external community members to participate in this project,” Klint explains. “And that is on purpose: we already have enough on our plate managing all the partners.”

“At this stage, our industry partners are providing the input and feedback from the user community. They make sure the functionality fits the market demands. As a matter of fact, the inception of this research project was driven by their needs."

"After the project is finished we will provide ongoing support and facilities for the community of users and metric contributors," Hansen says. "That way, the platform will continue to evolve after the EC funding has stopped."

Use case: Building Information Modelling

"Today's building and construction projects rely heavily on ICT for creating, managing and sharing all kinds of information throughout the whole value chain," says Pedro Maló,
senior researcher at the Uninova Centre of Technology and Systems in Portugal. "This goes from the early conceptual stage, to sketching, architecting, engineering, construction management, and inspection, all the way through operation, maintenance, and even demolition. As a matter of fact, buildings nowadays are first fully developed virtually and only thereafter built in reality."

"In this project Uninova is partnering with the software company Unparallel Innovation to facilitate the decision process of comparing and selecting open source software used in Building Information Modelling (BIM). The interoperability of disparate data models has always been an issue in this world, and now even more than ever with all these BIM systems around. A Heterogeneous BIM Model Server will be able to act as a mediator for data exchange between various applications in the collaboration environments. Altogether this project will be implemented over a period of 18 months, at an estimated cost of 100,000 euro. And as such, the size and complexity of a BIM Model Server requires the use of open source software for several of its parts."

"We are already using open source software for the core data storage and for Object-Relational Mapping (ORM), but we are also looking for other opportunities to adopt open source software, for instance on the adaptors for the import/export of BIM data. The process of inspecting and evaluating existing OSS solutions, however, is hard and time-consuming. It takes a skilled engineer many days to analyse and compare various packages, and even then some aspects remain hard to assess, e.g. code quality, project performance, and use by others. Ossmeter can provide the right tool to evaluate many key dimensions of OSS packages and reduce the time for analysis."

Eclipse-based tools and components

"The OSSMETER project has been under development for over a year now," says Klint. "It is based on MongoDB [a document-oriented database system] and Rascal [a meta-programming language for analysing and transforming source code]."

"Other tools and components are OSGi [a module system and service platform for Java, implementing dynamic modules], Epsilon [a family of languages and tools for code generation, model-to-model transformation, model validation, comparison, migration and refactoring, supporting EMF], Emfatic [a language for the declaration of EMF models], SVNKit [a Subversion library for Java], EGit [a Git library for Eclipse], and Thrift [a software framework for cross-language services development]," explains Dimitris Kolovos, lecturer at the University of York. "The Ossmeter platform itself is implemented using a toolset built on top of Eclipse."

Both CWI and the University of York are already heavily involved with the Eclipse community: CWI for its Rascal technology, and the University of York for Epsilon and Emfatic.

Dissemination

"We are using an agile software development process," says Klint. "A demo-like web-interface is already available, so at this moment we are able to run a small number of metrics on a limited amount of packages. We wanted to have a round-trip prototype as soon as possible, visiting every part of the architecture. That allows us to run the core process chain, add new components, improve the APIs along the way, and verify that all parts are working together smoothly. It's still small, but we are currently expanding the platform with more metrics and additional visualisations [e.g. tables, diagrams, word clouds]. Our industrial partners are involved at this stage of the project, but their main efforts will focus on field tests when the first prototype is available. In the meantime we receive their feedback in quarterly project meetings."

The Ossmeter toolset will be verified using three test cases. Two of these are industrial case studies in IT services and construction, provided respectively by research partners Softeam and Unparallel Innovation. The third case contains the OSS projects related to the Eclipse software development platform, where partner Tecnalia is actively involved. The latter also provides a way to engage the Eclipse Foundation and the Eclipse community with the evaluation process for the Ossmeter platform, and to disseminate the platform across the IT industry.

Documentation

OSSmeter3-short2-EC.pdf

OSSmeter3-short2-EC.odt

Information

Themes:	Research and Intellectual Property, Science