© 2014 *Vilnius University, IOI*

CMS: a Growing Grading System

Stefano MAGGIOLO[1], Giovanni MASCELLANI[2],
Luca WEHRSTEDT[3]

[1] *London, U.K.*
[2] *Faculty of Sciences, Scuola Normale Superiore*
Piazza dei Cavalieri 7, 56126 Pisa, Italy
[3] *Department of Mathematics, University of Bologna*
Piazza di Porta San Donato 5, 40126 Bologna, Italy
e-mail: s.maggiolo@gmail.com, giovanni.mascellani@sns.it,
luca.wehrstedt@gmail.com

Abstract. We give an update on CMS, the free and open source grading system used in IOI 2012, 2013 and 2014. In particular, we focus on the new features and development practices; on what we learned by running dozens of contests with CMS; on the community of users and developers that has started to grow around it.

Keywords: CMS, contest management system, grading system, IOI, IOI-like competitions.

1. Introduction

CMS (*Contest Management System*) is a free and open source grading system to run the IOI and similar programming contests[1]. Since our first presentation in (Maggiolo and Mascellani, 2012) the project saw a lot of activity: new features were added, some parts were redesigned, many bugs were fixed. CMS has been used in two IOI editions (and will be used in 2014) as well as in dozens of other contests all around the world, both on-line and on-site, from small local contests to international ones. It has received suggestions, bug reports and code contributions from various enthusiastic developers in many different countries.

We thus believe that it is time for us to give a new public update to the IOI community about the state of the project, summarizing what has happened since the first presentation of CMS and briefly covering where the CMS development is headed.

We will not go again over the motivations, design principles and general structure of CMS: most of what was described in (Maggiolo and Mascellani, 2012) is still valid. Instead, we focus on what we learned from working on a more mature code base, with wider adoption, larger feedback from users and more contributions external to the core team.

[1] CMS's home page is `http://cms-dev.github.io/`

2. New Development

2.1. Development History

When (Maggiolo and Mascellani, 2012) was being written, CMS was about one year and a half old, and it was a project led and developed almost exclusively by three core developers involved in the Italian Olympiads in Informatics and later in the organization of IOI 2012, held in Italy.

At the time, obviously, CMS development was very tied to the IOI schedule: the CMS development group was a subset of the IOI's Host Scientific Committee, and all efforts were directed to be ready for IOI 2012. Therefore, we used a simple development model, without formal releases: IOI 2012 was essentially the first public appearance of CMS, and we planned to release CMS's first official version soon after. As hosts know, the IOI week is a hectic time when all sorts of previously overlooked small bugs start causing lots of problems, and at the same time unorganized fixes accumulate. With our post-IOI release, we implemented proper solutions substituting the fast fixes and we identified specific areas of improvement for future releases.

Indeed, a very important criterion for a grading system used at the IOI is the ability to easily merge upstream the changes introduced during the IOI, as this guarantees that known problems do not propagate to the following IOIs, and that new features (for example, to support new rules) are not implemented several times by different hosts.

We released CMS 0.9 in November 2012[2]. Its structure is essentially the same as that described in (Maggiolo and Mascellani, 2012). Apart from many small improvements and fixes, we implemented user tests (in the sense of the IOI rules): the possibility for contestants to execute their source code against their own input files in the same environment where their solution will be evaluated.

In March 2013 we released CMS 1.0[3]. This was intended to be an evolutionary release that continued the post-IOI work. Its highlights were a vastly improved documentation[4] and full support for the translation of the contestant interface.

The version used at IOI 2013 was cut from the post-1.0 development branch two months later, and it included two major additional features: the new sandbox, *isolate* (Mareš and Blackham, 2012), and task versioning.

We continued the development of CMS 1.1, which is going to be released before this article is published. The main additions have been the transition to the new event loop library, a new service taking care of the communication with *RankingWebServer*, support for additional programming languages, easier to write importers, new translations, improved testing.

We also started improving our development practices: we began reviewing all new code entering the repository; we focused in improving our tests, increasing their coverage; and we set up a continuous integration system[5].

[2] Release notes at `https://github.com/cms-dev/cms/wiki/CMS-0.9.0-RELEASE-NOTES`
[3] Release notes at `https://github.com/cms-dev/cms/wiki/CMS-1.0.0-RELEASE-NOTES`
[4] The documentation is available at `https://cms.readthedocs.org/`
[5] The continuous integration web interface is reachable at `http://cms.di.unipi.it/jenkins/`

During the two years that brought us here, we had the pleasure to appoint two new core developers (from Italy and Australia) and to receive contributions from other sixteen people around the world[6]. Many contributions came from future IOI hosts that decided or are considering using CMS as their grading system. Nonetheless, as the number of national teams using CMS for training and selection rises, we have seen also a growing number of contributions from people not involved in IOI hosting.

2.2. *New Features*

We list in this section the main differences between CMS pre-0.9 and CMS 1.1.

User tests. As per IOI rules, contestants can test their solutions against an input they propose, and the execution will be performed in the same environment as the evaluation against the official testcases.

Improved contestants interface. We implemented a new web UI for contestants, based on Bootstrap (Twitter, Inc., 2010), much nicer to the eye and easier to understand. The interface has also been made completely translatable and contestants can change the language.

Translations. At the moment of writing, CMS has been translated in nine languages: Bosnian, Dutch, English, French, Italian, Japanese, Lithuanian, Russian and Traditional Chinese. We welcome contributions to extend the list further.

Task versioning. More often than one would want, during a contest it is realized that some testcases are wrong. With CMS, administrators can create new sets of testcases, evaluate all submissions against them and find out how many contestants were affected by the problem; all of this in "background", without taking down the task or the scores for the initial set of inputs. When the new testcases are validated, administrators can switch to them and notify only the affected contestants, without any downtime and without most contestants even noticing.

Task versioning is not limited to input files: it can also be used to test new time and memory limits, or different libraries, graders, or scoring functions.

New sandbox. The previous sandbox, *mo-box* (Mareš and Gavenčiak, 2001), was based on system calls filtering; maintaining the list of allowed calls for compilations and evaluations was often difficult, as it depended on the architecture, the operating system and the programming language. The new sandbox, *isolate* (Mareš and Blackham, 2012), was again co-developed by Martin Mareš and is based on the new namespace features of the Linux kernel. It requires a reasonably recent version of the kernel (at least 3.8) and the *isolate* executable must be run as root (which is accomplished in CMS using the *suid* flag), but it does not require special configuration and in particular architecture-dependent ones. Moreover, it enforces limitations directly on the resources, instead than on the calls used to obtain them. It also causes much less computational overhead.

[6] A complete list is at `https://github.com/cms-dev/cms/blob/af11e8d6/AUTHORS.txt`

New event loop library. Up to CMS 1.0 our custom-made RPC system was based upon Python's *asyncore* framework, which now exists for "backward compatibility only" and is eventually going to be removed from the standard Python libraries. Our HTTP servers were built on top of *Tornado* (Facebook, Inc., 2009), which had its own event loop: we had therefore to have them both running simultaneously, interleaving their steps. The awkwardness of this design and the serious performance issues indirectly caused by it that came up at IOI 2012 (see section 3.1 for more details) prompted us to switch to *gevent* (Bilenko, 2014), a coroutine-oriented Python library based on the low-level *libev* (Lehmann and Giaquinta, 2014) event loop.

New RPC system. Our RPC library, called AsyncLibrary, was based on *asyncore* and was hence dropped after the transition to *gevent*. We wrote a new one that fully benefits from the new paradigm. That has been a good chance to make it more modular and safer (for example by catching and logging all exceptions in callbacks).

We also improved performance by avoiding opening more than two connections (one in each direction) between any pair of services.

ScoringService. The IOI 2013 experienced issues with slow scoring (Blackham, 2013): after fixing a testcase, the rescore took so long that they could not determine the affected contestants before the end of the contest. The slow rescoring was introduced on purpose to return the control to the event loop regularly (as the service would have otherwise appeared stuck to the rest of CMS). The problem was fixed by porting the service to *gevent*: that made the regular pauses unnecessary as the event loop could take back control in any time during the execution of I/O.

ProxyService. The philosophy of CMS has always been to use many small services that have only a small number of duties, possibly just one; this helps keeping most of the functionalities up in case something goes wrong in a specific part of CMS. In the previous design, *ScoringService* had two duties: to compute the score of each submission and to send these scores to the ranking server. Therefore, we moved the latter to a new service, called *ProxyService*.

Importers and loaders. In CMS 1.0 we had a utility, called *YamlImporter*, to easily load into CMS contests and tasks prepared using the file system format of the Italian Olympiads. A companion utility, *YamlReimporter*, was used to "reimport" an already-existing contest, i.e., updating its data without losing the submissions already sent by the users.

We always stressed that CMS should not force a specific file system format to the administrators, but the complexity of *YamlImporter* and *YamlReimporter* made it difficult to write similar utilities for other formats. Therefore, we split them into two format-independent parts (*Importer* and *Reimporter*) and a loader, which is specific to our format. This way, the support for another format can be added by just implementing a new loader, which only has to create the appropriate objects from an external source, usually a file system representation. We received some externally contributed loaders over the last months.

Programming languages support. We added support for new competition languages: Java (through gcj), Python, PHP, in addition to the classical C, C++ and Pascal. It is now trivial to add support for other compiled languages and for some interpreted ones. Note that this is not an endorsement for allowing such new languages in the IOI; in particular, individual languages can be allowed or not for each contest.

Extended documentation. CMS has now rather comprehensive user documentation, covering the whole process of setting up CMS to run a contest. From the developer side, we have about two lines of comments every three lines of code, thanks especially to our commitment to write docstrings for every function. As CMS becomes a larger project, some shortcoming of Python's duck typing system started to become apparent, and we reacted increasing the documentation of the types of arguments and return values of functions. Moreover, a tool was developed to ensure that CMS was actually respecting the indications written in the docstrings (Maggiolo, 2013).

3. CMS Usage

3.1. *IOI*

CMS was used for running two IOI editions, in 2012 (Sirmione and Montichiari, Italy) and 2013 (Brisbane, Australia); it will also be used in IOI 2014 (Taipei, Taiwan). In both past cases CMS performed mostly well; while during the two contests there were some technical problems, most of them did not depend on CMS misbehaviour, but on mistakes in the data provided to it (e.g., wrong testcases or graders) or on other faults in the network environment. For a detailed discussion of what happened at IOI 2013, please see (Blackham, 2013).

There were, though, some issues that were CMS bugs. Probably the most important single issue was the inefficiency in the networking framework on which CMS was based. The RPC and HTTP servers were built on top of *asyncore* and *Tornado*, and took advantage of their non-blocking, callback-based APIs. Unfortunately, connections opened outside the scope of these frameworks did not benefit from it and any read or write operation on them was blocking for the whole application. Such instances were, in particular, the connections to the database (handled by SQLAlchemy) and the HTTP requests to *RankingWebServer* (handled by *httplib*).

Both of these caused serious performance bottlenecks at IOI 2012. Some services (like *ContestWebServer* and *AdminWebServer*) usually spend most of their time doing database queries: being unable to handle other requests while waiting for the results of a query made them unresponsive, especially during periods of high load or when performing large queries. At the IOI this resulted in *ContestWebServer* not being able to handle the request burst at the beginning of each day and appearing to be down for minutes. *AdminWebServer* did also hang often, but this did not cause problems to contestants.

On the other hand the internet connection at the IOI 2012 site was very poor and there was a lag of a few seconds on all outgoing requests. That caused the rate at which data was sent to *RankingWebServer* to be much less that the rate at which new data was coming in: *ScoringService* was spending all its time waiting, neglecting its duty to score submissions and building up large queues. This issue was somewhat relieved by grouping all queued data into a single HTTP request. Yet, it was not enough and we ended up using two threads for the two distinct operations. That contributed to induce us to split off *ProxyService*.

Using for example *Tornado*'s *HTTPClient* (instead of *httplib*) to handle the HTTP connections may have resolved this issue, but we could not find viable alternatives to SQLAlchemy: the few that existed seemed to be less powerful and mature. In the end we decided to switch to *gevent*. Its execution model is based on having many execution units called "coroutines", that are a lightweight form of cooperative threads: each of them runs code that performs reads and writes using a synchronous blocking API, but I/O operations are transparently translated to non-blocking calls and, while waiting, control is returned to the event loop that allows other coroutines to resume their work. Within CMS, SQLAlchemy uses Psycopg as backend towards the PostgreSQL server, which is easily made compatible with coroutines, as detailed in (Varrazzo, 2010). Other libraries, that were not originally designed to be cooperative, can be added support to by using *gevent*'s monkey-patching capabilities. Although the *gevent* support was already written, it was not used at IOI 2013, because it was still young and not well tested.

3.2. *National and Local Contests*

After its presentation at IOI 2012, CMS was used in many different countries for contests with sizes ranging from local to international. We are aware of contests organized in Argentina, Australia, Belgium, Chile, Croatia, India, Italy, Japan, Latvia, Lithuania, Serbia, Slovenia, Taiwan and Tunisia[7]. It was used both for on-line and on-site contests, from a dozen to around a hundred contestants; some contests run with CMS were also hosted on public cloud computing services, such as the well-known Amazon EC2 engine.

CMS is also used to run permanent online instances, which do not serve specific contests, but allow users to continuously submit solutions to the set of offered tasks. Such instances are used as tools for the training of national IOI teams[8] or for collecting the homework assigned to students during university courses and make the students able to receive a direct feedback on their work[9].

Some forks were devised from CMS for handling more specific situations or contest types. For instance, William Di Luigi and Luca Versari added some social features like the possibility for users to interact with a forum[10]; Masaki Hara runs a CMS instance which serves contests for the Japanese Olympiad[11] which has support for login via Twitter or Facebook authentication.

[7] See a more complete list at `http://cms-dev.github.io/testimonials.html`

[8] For example, there is an instance for the training of the Italian team at `http://cms.di.unipi.it/`

[9] For example, `http://judge.science.unitn.it/`, handling exercises for the Algorithms and Data Structure class at the University of Trento.

[10] This is the case of the already mentioned instance `http://cms.di.unipi.it/`, which is run by code at `https://github.com/veluca93/oii-web`

[11] Code at `https://github.com/qnighy/cms`, public instance at `http://cms.ioi-jp.org/`

4. Future Plans

Our main goal for the future, as members of the core development team, is to make us less central in the development of CMS: to do so, we need more people to send us contributions. Translations, bug reports and fixes are always welcome; for people intending to become more stable contributors, we set up a page[12] with some ideas for interesting, self-contained projects that offers a gentle introduction to the development side of CMS. We are open to offer help and tutoring during the implementation of these ideas.

An obvious area for improvement for us is to learn to release more often. CMS 1.1 took too much time to be released and this created problems as the features introduced in the development version started to justify using it despite being, for obvious reasons, less stable than CMS 1.0. Smaller, more frequent releases will allow us to deliver new features much sooner, and we intend to get better at that. The new testing and continuous integration infrastructure will help us with this goal. Therefore, increasing the coverage of our tests, and hence the trust on them, is another main goal.

The IOI is by far CMS's main client, therefore we will continue supporting any IOI rule change and any new task format. In our experience of these past years, we realized that national competitions often have different requirements. We tried to do our best to serve the community while keeping our focus on the IOI, and we will certainly continue working with the interested national teams to support as many use cases as possible.

In terms of new features, we have at least two big changes coming ahead. The first is a reorganization of how files associated to a task or to a submission are specified in the task configuration; this will make it easier to configure tasks and possibly write new task types. The second is a redesign from the ground up of *AdminWebServer*, that will expose a simpler and more informative interface for contest administrators and will realign it to the UI of *ContestWebServer*.

5. Conclusion

We have described what has changed in CMS in the last two years, the *status quo* and where we plan to direct our development effort. After three years of work, we believe CMS to be a valid and proved contest system and we invite the whole IOI community (and, more generally, all those who are interested in organizing programming contests) to try it, evaluate its suitability for hosting the types of contests they are interested into and let us know their impressions and suggestions. In our development decisions we welcome and consider the feedback received from our users.

As pointed out above, we are looking forward to receive contributions. Beside code development, another way of contributing is by providing translations: it is our commitment to offer an easy to use interface for all contestants, also in cases where English is not necessarily the *lingua franca* (for instance, for local or national contests). Potential

[12] `http://cms-dev.github.io/contribute.html`

contributors are welcome to read the relevant pages in the documentation[13] and get in touch with the CMS development team to have their translations accepted in the main repository.

Acknowledgments

We would like to heartily thank all contributors and users of CMS, in all roles: developers, translators, contest administrators, especially when providing feedback, ideas and bug reports. It is incredibly fulfilling for us to witness the growing of a community around our project, which we intended and hoped from its start to become a useful tool for all people involved in the IOI and in similar contests.

We look forward to continue working with all of the past and current contributors and we welcome anybody willing to donate their effort and their skills to improve CMS.

The CMS project was initially funded by by AICA (Associazione Italiana per il Calcolo Automatico) and MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca) and continues as a volunteer effort.

References

Bilenko, D. (2014). *gevent*. `http://www.gevent.org/`

Blackham, B. (2013). "*Did that Really Just Happen?*" *– A Behind the Scenes Look at IOI 2013*. `http://goo.gl/f63r78`

Facebook, Inc. (2009). *Tornado Web Server*. `http://www.tornadoweb.org/`

Lehmann, M., Giaquinta, E. (2014). *libev*. `http://software.schmorp.de/pkg/libev.html`

Maggiolo, S. (2013). *Pydoc Checker*. `https://github.com/stefano-maggiolo/pydocchecker`

Maggiolo, S., Mascellani, G. (2012). Introducing CMS: a contest management system. *Olympiads in Informatics*, 6, 86–99.

Mareš, M., Blackham B. (2012). A new contest sandbox. *Olympiads in Informatics*, 6, 100–109.

Mareš, M., Gavenčiak, T. (2001). *The Moe Contest Environment*. `http://www.ucw.cz/moe/`

Twitter, Inc. (2010). *Bootstrap*. `http://getbootstrap.com/`

Varrazzo, D. (2010). *Coroutine Support for Psycopg*. `https://bitbucket.org/dvarrazzo/psycogreen-hg/`

[13] `http://cms.readthedocs.org/en/latest/Localization.html`

S. Maggiolo is a software engineer at Google and holds a Ph.D. in Geometry from SISSA/ISAS, Trieste. He participated in IOI 2002, winning a bronze medal and in IOI 2003. Since 2006 he collaborates with the training and selection process for the Italian team at IOI, and has been Observer in IOI 2009, Deputy Leader of the Italian team in IOI 2011 and a HSC member for IOI 2012. He is one of the main authors of CMS.

G. Mascellani has been a contestant in IOI 2007 and 2008, winning a silver and a bronze medal. He is a Ph.D. student in Geometric Analysis at the Scuola Normale Superiore (Pisa, Italy) and collaborates in training the Italian team for IOI. He was a member of the HSC for IOI 2012 and is one of the main authors of CMS. He is a Debian Developer.

L. Wehrstedt participated twice in the IOI, in 2010 and 2011, winning a bronze medal. He is attending a Bachelor's degree in Mathematics at the University of Bologna and he is a student at the Collegio Superiore di Bologna. He has been involved in the training of the Italian IOI team for the last three years, helped organizing two Italian Olympiads in Informatics and was in the HSC of the IOI 2012. He is one of the core developers of CMS.