A new Paradigm for Open Robotics Research with the C++ Object Oriented Mechanics Library

Alberto Valero-Gómez, Mario Almagro-Cádiz,
Nieves Cubo-Mateo
Robotics Laboratory
Universidad Carlos III de Madrid
Madrid, Spain
alberto.valero.gomez@gmail.com

Juan González-Gómez
Centro de Automática y Robótica
CSIC-UPM
Madrid, Spain
juan@iearobotics.com

***Abstract*— For many years robotics has been benefited by the open source community. Community projects like Player/Stage/Gazebo, ROS, or OpenCV are present in most robotic applications. In recent years this trend has also been initiated among the electronic developments (open hardware). Arduino is a good example of an open hardaware project, with a big community of developers around it. In this paper we are presenting a tool for designing the mechanical components of a robot. These designs can be shared, reused, and modified easily. This tool is called the C++ Object Oriented Mechanics Library (OOML). The OOML brings together the advantages of the object oriented programming paradigm and the power of C++. Thanks to the OOML, robots and their mechanical parts can be completely open and reusable, allowing researchers around the world to clone or evolve them.**

I. INTRODUCTION

Four years ago, Richard Stallman said: "maybe in 10 years most users, common or not, will use free software and will have the freedom and control of their own computing" (Invited lecture at the 4th Congress of Free Software in Venezuela, 2008). In this article we are presenting a new coding tool for roboticists that will introduce the mechanical design of robotic parts (if not the whole robot) into the open source community.

The goal of this tool is to allow researches or consumers to reuse, re-share, and modify the physical design of the robots. We have called it the *Object Oriented Mechanics Library* (OOML). The OOML is a C++ library for designing mechanical parts. The parts can be then mechanized using the classical tools for fabrication, or the new personal (and low cost) 3D printers. We will speak afterwards more about 3D printers, but it is important to highlight in this introduction that what personal 3D printers are bringing to the mechanical design is deemed to be analogous to what personal computers brought to the open source community[1][2]: people all around the world sharing code and being able to contribute in a joint effort to build open source projects. With 3D printers, mechanical designers can clone rapidly and cheaply others designs, use them, or adapt them to their requirements [3].

This paper is organized as follows. First of all we will set the basis of open science and world spread communities. Afterwards we will illustrate why the object oriented programming paradigm (OOP) emerged as a powerful tool for code sharing and reusability. Taking the model of the OOP we will justify why this paradigm fits in the design of mechanical parts. After showing the benefits of open source communities and the object oriented programming paradigm, we will present our contribution: the *C++ Object Oriented Programming Library*, an open C++ library that applies the OOP paradigm to the design of mechanical parts. A final section presenting the developments made up to the date in OOML will justify the expected impact of this tool in the research community, and the advantages that it may bring to the roboticists community.

II. ON OPEN SCIENCE AND RESEARCH

The Organisation for Economic Cooperation and Development (OECD) Guidelines defines openness as: "... access on equal terms for the international research community at the lowest possible cost, preferably at no more than the marginal cost of dissemination. Open access to research data from public funding should be easy, timely, user-friendly and preferably Internet-based" [4].

One of the major questions affecting open science regards the economical viability of such a business model, is it possible to make money when all your ideas, methods, and protocols are available for everyone? In this work we are not concerned about this question. Instead, the question we are interested in is: *is open science a model that enhances research? can world spread communities, sharing their data, results, methods and ideas contribute in a better way to knowledge?* Funded research institutions have the freedom to develop research that may not be economically sound in the short term. They are not under the pressure of the market, and consequently they can focus on the research topic in itself without being worried on the economical market competitiveness: their purpose is not to sell a product, but to generate the required knowledge to create a product [4]. This is the case of public universities or research centres. Is open science a valid model to make research in such institutions?

In [5] the authors investigate the perceived benefits to researchers, research institutions and funding bodies of utilising open scientific methods. The conclusions of the cited work summarize the advantages of open science around five main issues: 1) speed and efficiency of the research cycle; 2)capabilities to identify new research questions; 3) research

effectiveness and quality; 4) innovation, knowledge exchange and impact; and 5) research group and career development. The authors conclude that many of the benefits envisaged for open methods relate to how far they enable not only access but active participation in a research community by newcomers and outsiders, and maintain low barriers to this participation. Greater visibility for research producers, lower barriers to collaboration, and more reusable datasets are strong motivations.

The community model is inherent to the open science. In fact, much of the open source philosophy is based on the principle of allowing participation to as many people as possible [6]. Initially the developers are attracted by the technical details, but according to [7], "participating in an open-source community is a continuous learning process, where not only technology is to be considered but other factors, such as group work and communication". Very likely, the future engineers will have to integrate themselves in one or several of these communities. Therefore, it is important for them to know how to develop their work being part of these communities. Moreover, they should learn how to create, lead and inspire these communities.

Up to now, open science has been extensively applied to software development with the open source software. The most obvious example is the operative system GNU/Linux. The impact of systems like Ubuntu show the viability of such projects, not only for small and specialized communities but also for the general public. The scientific community has been benefited by an important increase of productivity because of this effect [8]. The open source community underwent an impressive grow with the introduction of the object oriented programming (OOP) paradigm. In fact, it is inherent to the goals of OOP collaboration, reutilisation, and modification of code. Allowing for bigger projects and independent developments that can be joined together [9].

Recently a similar movement oriented to the electronics has arisen. It is called open hardware. The open hardware shares all information concerning the hardware development and design. Following this paradigm appears the Arduino project (and company) [10].

Open source developments have had a deep impact on the robotic community, and the same is lately happening with the open hardware. For the physical/mechanical design of robots, insofar there have been few sharing communities. In fact, most robotic researchers test their algorithms on simulation or with a small set of manufactured platforms like the Pioneer robots, or the Khepera. Personal 3D printers are changing this panorama. In the following section we are showing how 3D printers are an optimal tool to boost the open design of physical objects.

III. Impact of Personal 3D Printers

Bradshaw et. al [11] recently made a study on low-cost 3D printing. They briefly run through the history of 3D printing, beginning in the late 1970s. These more than thirty years have driven to affordable personal 3D printers [12]. With 3D printers individuals can print complex engineering parts entirely automatically from design files. These files can be shared over the internet, allowing anyone with access to a 3D printer to reproduce the same physical thing.

While open source software development has been studied extensively, relatively little is known about the viability of the same development model for a physical object design. 3D printers are offering new possibilities of sharing physical objects. They allow for a decentralized community to independently produce physical parts based on digital designs that are shared via the Internet. Bruijn shows that a considerable improvement on physical things is proposed by people sharing parts and having access to 3D printers [13]. With a 3D printer, these modifications are relatively easy for others to replicate. As it has been the case with software for many years, currently, there are also on-line repositories of parts, where people can download and upload their designs[1].

In Figure 1 four of the most important open source 3D-printers are shown. The origin of these kind of printers was the reprap project[2] [14] started by Adrian Bowyer in 2004. The aim of this project was to develop an open-source self-replicating machine. In May 2007 the first prototype, called Darwin was finished and some days later, in May 29th the first replication was achieved. Since then, the reprap community (original reprap machines and derived designs) has been growing exponentially[13]. The current estimated population is around 4500 machines. The second reprap generation, called Mendel, was finished in September 2009.

Initially, both Darwin and Mendel were not designed for the general public but for people with some technical background. As the reprap project was open-source, small companies were created to start selling these 3D printers, as well as derived designs. The first company was Makerbot Industries[3], who shipped a first batch of their Cupcake CNC in April 2009. By the end of 2009 they had shipped nearly 500 complete kits. After operating for a year they had sold about 1000 kits. Afterwards came the Thing-O-Matic printer, announced in September 2010, easy to build and use, and for 950$. Their current model has been called The Replicator, it comes already assembled and it is the first personal open printer with dual extruder, allowing to use support material or print in two colors. Its price is around 1900$. Other open printers have arisen inspired by the RepRap and following the commercial impulse of the MakerBot. These are the PrintrBot (which cost is just 450$) or the UltiMaker (around 1000$).

IV. Open Design of Physical Objects

Up to this point, we have tried to motivate why open knowledge communities enhance research by sharing the sources of designs. We have shown how this model has succeed in the software and electronics communities. Recently, thanks to the possibility of replicating 3D objects, this movement has been initiated also in the mechanical design research.

[1] http://www.thingiverse.com
[2] http://reprap.org/wiki/Main_Page
[3] http://www.makerbot.com/

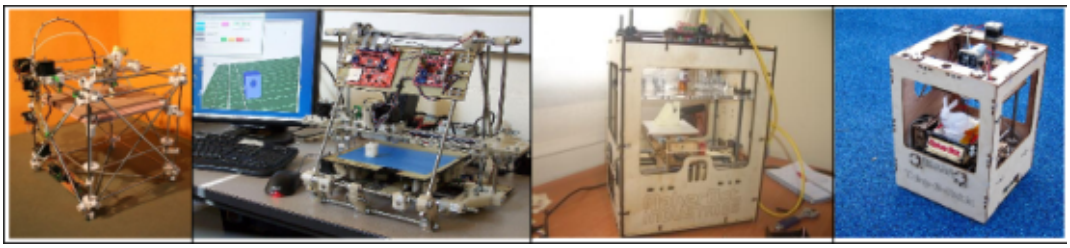


Fig. 1: Pictures of some open source 3D printers. From left to the right: RepRap Darwin, the first generation (May, 2007); Reprap Mendel (Sep, 2009) the second generation; Makerbot Cupcake (April, 2009), the first commercial open-source 3D printer; Makerbot Thing-o-Matic (Sep, 2010), second version

The first requirement to share a thing is the possibility to digitalize the information describing it. There are many formats describing 3D things, currently, the most used is the STL format. The STL (Standard Tessellation Language) is a file format native to the stereolithography CAD software created by 3D Systems. It is widely used for rapid prototyping and computer-aided manufacturing. An STL file describes a raw unstructured triangulated surface by the unit normal and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system. Anyone having the STL can reproduce the physical thing.

This would involve only sharing as modifying the STL file would be too complex. So, in order to allow collaboration designers would need tools to describe their designs and capable of generating the STL afterwards. This is analogous to source code and binary libraries or executables in software development.

Designing mechanical parts has been traditionally made using graphical CAD programs, following the WYSIWYG paradigm (what you see is what you get). Diverging from the main stream of 3D design OpenSCAD was born. Unlike most 3D CAD applications OpenSCAD does not follow the WYSIWYG but a derivative called WYGIWYM (what you get is what you mean). In a WYGIWYM editor, the user writes the contents in a structured way, marking the content according to its meaning, its significance in the document, instead of designing its appearance. This requires the structure of the document (contents semantics) to be known before editing it. The editor also needs a system for exporting the edited text to generate the final format of the document, following the indicated structure. The main advantage of this paradigm is the total separation of presentation and content: designers can concentrate their efforts on structuring and writing the document, rather than concerning themselves with the appearance of the document, which is left to the export system. Another advantage is that the same content can more easily be exported in different formats. OpenSCAD, which follows this paradigm, is something like a 3D-compiler that reads a script file describing the object. Then it renders the 3D model from this script file. Finally it gives the possibility of exporting it to different standard formats for 3D objects (STL, OFF, DXF).

Designing 3D objects through code has opened a new possibility that until now was only present in the software community. As it has been the case with software for many years, currently, 3D objects can be shared on the internet, modified, improved and reused. OpenSCAD is (to our knowledge) the only coding environment+language for modelling mechanical things, nonetheless, the idea of designing 3D objects (in this case for visualization and video editing) through code is also present in PovRAY.

In the next section we will motivate our decision to move from OpenSCAD to a new coding tool for designing 3D objects. This is the OOML, and we believe, as it will be supported, that it is the natural evolution from OpenSCAD. The evolution consists of introducing the object oriented programming paradigm in the design of physical objects.

V. OOP in Mechanics

The OOP is based on the simulation of the behaviour of physical objects. Thus, it seems to be an effective method to design physical objects using OOP. We will see the main properties of OOP and how it applies to mechanics.

Encapsulation and modularity. Physical objects have many properties, some of these properties can be considered independent, while other dependent. For example, in a gear, the radius and the teeth size determine the number of teeth and their position. That means that defining a gear wheel, we only need to fix two parameters, radius and teeth size, for example, being the rest computed according to these. Consequently, we are encapsulating some information that the designer does not need to set for defining the gear. Using the proper interface the *consumer* designer can define the physical object. It is the task of the *producer* designer of the thing to decide which interface the class must provide to the consumer of the object.

Inheritance and Composition. The Inheritance of classes is based on the IS-A, relation, in this sense it is easy to see this relation in physical objects, a gear wheel IS-A wheel, and so we could define two classes, one inheriting from the other. As for the composition is even more clear, a robot, for example is composed of parts.

Polymorphism. Polymorphism refers to the fact that each object provides the methods that apply to themselves even if the method is common to all objects. In the case of physical objects design this is straight forward. All objects are *described* by an STL file. Nonetheless, each class is

defined by its own STL. Polymorphism allows to make a solid class structure to define mechanical objects.

Abstraction. All wheels are conceptually the same, but each wheel can have different parameters. This illustrates that abstraction is an adequate methodology to define classes in physical objects.

As its name suggests, object-oriented programming uses all these properties to imitate physical objects, and so, why not using OOP to design objects?

VI. The OOML Library

Considering what has been said up to now, providing an open source tool that allows to design physical objects using the OOP paradigm, is expected to boost the design of robots, enhancing community collaboration and reusability. Up to now we could reproduce others algorithms in our platform, with this tool, we could reproduce ours algorithms in others platforms.

A. From OpenSCAD to OOML

In academic year 2009/2010 we started applying a project based learning experience with engineering students of 2nd year. Students were offered to follow voluntarily a course in which they would be taught to design robots, print them, design the electronics, place the sensors and finally program them [15].

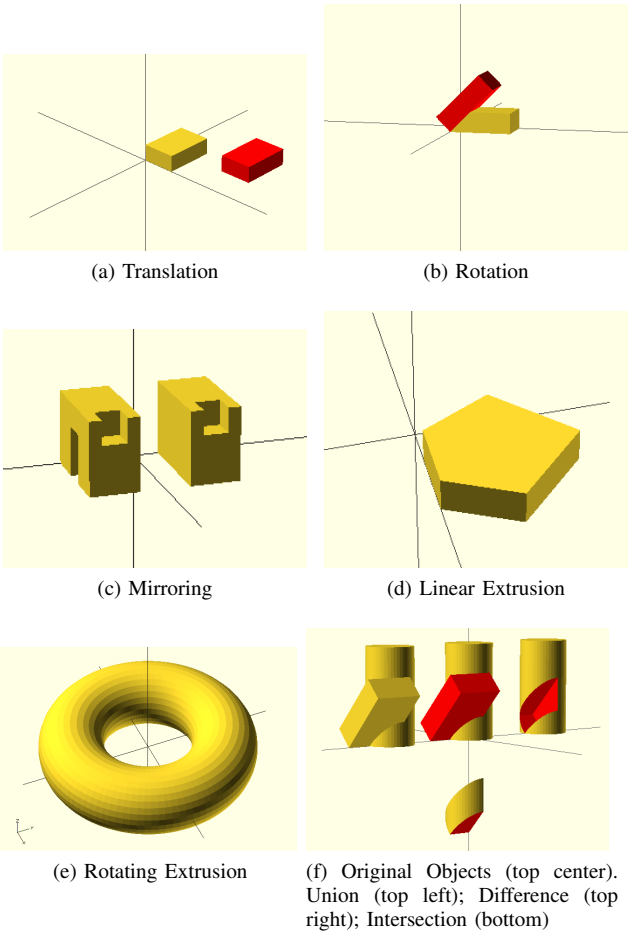
We taught them to design the mechanical parts using OpenSCAD. The electronics and the robotic agents were programmed using C/C++. Through the year we realized that the single parts designed by the students in OpenSCAD were not easy to share with each other. It was open source and it was code, but each student had its particular way of designing, which was different from the others. This made that the code of each student was hardly reused by the others. Another aspect that emerged was that after learning C++, students started to miss basic functionality that was not present in the OpenSCAD script, examples of this functionality were the while loops, the variables, etc. Also after learning C++ they started saying that it would be great to define physical objects as *programming objects*, and then convert the geometrical operations into methods of the objects.

After this feed-back the idea of the OOML was born and we started to develop it. The hard points of the idea were three.

- Physical objects are *programmed objects.*
- Geometric operations are the *methods of the object.*
- The parameters of the physical objects are the *attributes of the object.*

During the initial development of the OOML the idea of semantics arose. Geometric primitives are different from mechanical parts.

Finally, once semantics and object oriented things are designed, the concept of objects interface arises naturally, which standardizes the way in which primitives, parts are created and accessed making use of the C++ polymorphism, as all inherited objects include the same interface. The



(a) Translation (b) Rotation

(c) Mirroring (d) Linear Extrusion

(e) Rotating Extrusion (f) Original Objects (top center). Union (top left); Difference (top right); Intersection (bottom)

Fig. 2: Geometric Operations

first preliminary version of OOML was released in July 2011. The second version of the OOML was released in February 2012. Together with the library a web page with all the required tutorials to start using it was created at http://iearobotics.com/oomlwiki.

B. Geometry Operations

The OOML includes all the geometrical operations required for manipulating objects and creating new ones:

- Translation. Moves an object from a point to another, keeping its orientation.
- Rotation. Rotates an object around an axis.
- Intersection. Returns an object which is the result of the intersection of two or more objects.
- Addition. Returns an object resulting from adding other two or more objects.
- Difference. Substrates one or more objects from another and returns the resulting object.
- Scaling. Scales an object with different (or equal) scale factors on each axis.
- Mirroring. Mirrors an object with respect to a planar surface.
- Extrusion. Extrusion (linear and rotating) of planar figures.

Hence, with these basic operations and primitive objects the OOML is able to elaborate any object, no matters complexity.

C. Object Oriented

The OOML applies the model of Object Oriented Programming to Mechanical Designs. Parts, designed as objects follow all the Object Oriented Programming properties:

- Encapsulation: Parts hide to the designer the attributes and methods that are not required for their definition, restricting access to some of the object's components. Doing so the OOML facilitates the bundling of data with the methods (or other functions) operating on that data.
- Dynamic dispatch – when a method is invoked on an part, the part itself determines what code gets executed by looking up the method at run time in a table associated with the object.
- Object inheritance (or delegation), parts can inherit from others following the *IS-A* relation. For example, a gears wheel *IS-A* wheel, inheriting its configuration methods and attributes, for example: set_radius(), set_thickness(), set_axis_radius().
- Reusability: Parts can be made of the addition or composition of other parts.

D. Semantics Oriented

While in OpenSCAD all objects have the same semantic meaning (they are just objects), in OOML objects are divided in three categories:

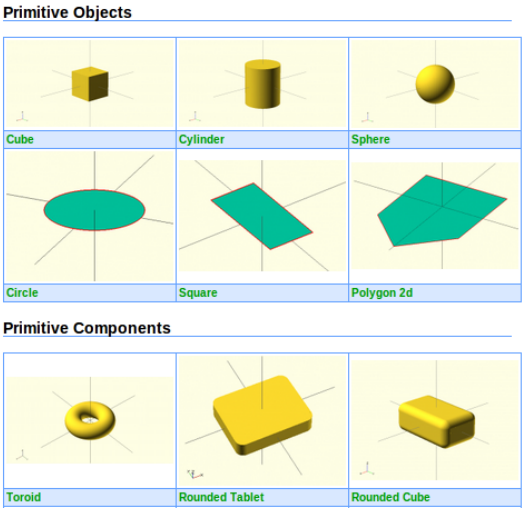
- Primitive Components: They are geometrical entities, without a mechanical meaning, like a Cylinder or a Sphere.
- Primitive Objects: They are geometrical entities, without a mechanical meaning made of the composition of the Primitive Components, like a Torus or a Rounded Box (Figure: 3a).
- Parts: They have a mechanical meaning in a complex *thing*, like a wheel or a chain.(Figure: 3b).

Applying the Object Oriented methodology all these objects share the same programming interface, so existing libraries are easy to use.

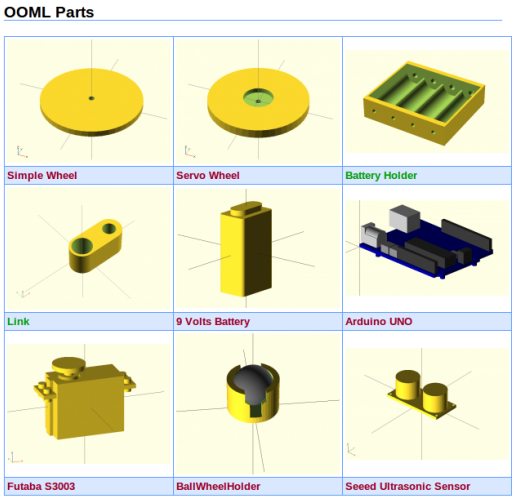
Once the physical object is defined, the OOML can generate OpenSCAD code, that can be used to generate the STL files for mechanization. Up to now the OOML only generates OpenSCAD code, but the library is designed to support any code generation, or even to produce directly the STL.

To sum up, the features of OOML are describes as follows:

- The OOML is Open Source, you can use it, modify it, share it.
- It generates OpenSCAD code. Using OpenSCAD designers may generate the STL of the parts.
- It allows designers to create mechanical parts using the C++ language and applying the Object Oriented Programming Paradigm.



(a) Primitives



(b) Parts

Fig. 3: OOML Objects

- The OOML is semantics oriented. It applies a semantic categorization of things.
- It includes all the geometry operations required to manipulate objects in space.

VII. Impact

There has been a great impact of using code for designing physical objects since OpenSCAD was born. On Thingiverse there are 2716 designs made with OpenSCAD (data taken the 15th March 2012). OOML is quite recent, currently there are only 14 designed things, but it is expected that this number will increase.

Figure 4 is a proof of the impact on the community of open source designed things. The skybot robot, mechanized in aluminium and plastic with traditional methods, has been available online for more than 5 years (electronics and mechanical designs). There have been a big number of courses on Universities and High Schools. However, this robot has not evolved mechanically in all this time. In 2011,

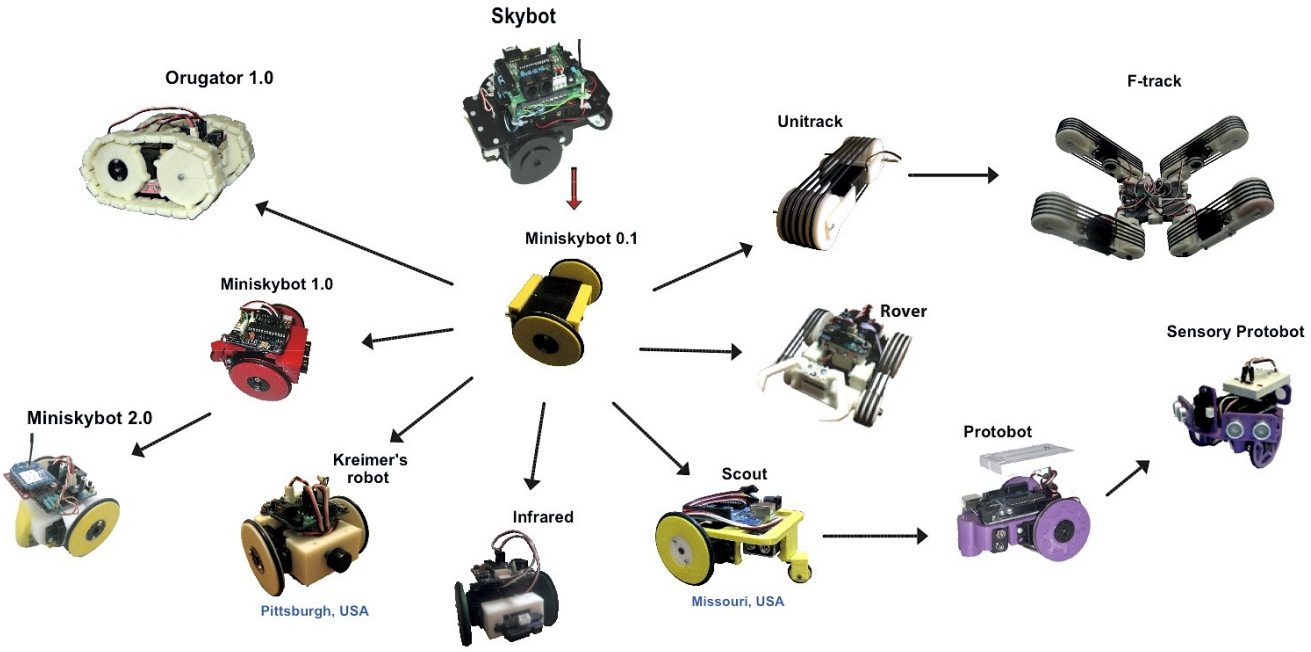


Fig. 4: Evolution and Diversification of the SkyBot Robot

the Mini-SkyBot, designed in OpenSCAD, was first printed and released on Thingiverse [16] [4]. In less than a year a great number of robots derived from this initial design, all over the world. The first full robot developed in OOML has been the ProtoBot[5] (February 2012). Since then there people around the world have replicated it and modified it, designing different kind of wheels and support for sensors.

VIII. Conclusions

In this paper we have presented the OOML, a tool for designing small robots. This tool is expected to allow the collaboration of researchers around the world to create new robotic platforms, facilitating to exchange their experiences and to share their robots. Thanks to this, algorithms can be tested on others platforms, and consequently, objective benchmarking can be used. 3D printers allow for a fast and low cost building of designed things. Examples of code and tutorials can be found in http://iearobotics.com/oomlwiki

The work to do is to increase the number of primitives and parts in the library. It will be the community to enrich the library, and thus, a dissemination work is necessary in order to have an impact in the research community.

References

[1] D. Bak, "Rapid prototyping or rapid production? 3d printing processes move industry towards the latter," *Assembly Automation*, vol. 23, no. 4, pp. 340–345, 2003. [Online]. Available: http://www.emeraldinsight.com/10.1108/01445150310501190

[2] A. Kochan, "Rapid prototyping gains speed, volume and precision," *Assembly Automation*, vol. 20, no. 4, pp. 295–299, 2000.

[3] C. Raasch, C. Herstatt, and K. Balka, "On the open design of tangible goods," *RD Management*, vol. 39, no. 4, pp. 382–393, 2009. [Online]. Available: http://dx.doi.org/10.1111/j.1467-9310.2009.00567.x

[4]http://www.thingiverse.com/thing:4954

[5]http://www.thingiverse.com/thing:18264

[4] (2007) Oecd principles and guidelines for access to research data from public funding. [Online]. Available: http:\/\/www.oecd.org\/document\/55\/0,3343,en_2649_34293_38500791_1_1_1_1,00.html

[5] A. Whyte and G. Pryor, "Open science in practice: Researcher perspectives and participation," *International Journal of Digital Curation*, vol. 6, no. 1, pp. 199–213, 2011. [Online]. Available: http://www.ijdc.net/index.php/ijdc/article/view/173

[6] E. S. Raymond, *The Cathedral and the Bazaar*, 1st ed., T. O'Reilly, Ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 1999.

[7] G. Robles, J. M. Gonzalez-Barahona, and J. Fernandez, "New trends from libre software that may change education," in *IEEE Engineering Education 2011 (EDUCON)*, IEEE Education Society. Amman, Jordan: IEEE Education Society, 04/2011 2011. [Online]. Available: http://www.educon-conference.org/educon2011/

[8] K. R. Lakhani and E. von Hippel, "How open source software works: free user-to-user assistance," *Research Policy*, vol. 32, no. 6, pp. 923 – 943, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0048733302000951

[9] F. Brito e Abreu and W. Melo, "Evaluating the impact of object-oriented design on software quality," in *Software Metrics Symposium, 1996., Proceedings of the 3rd International*, mar 1996, pp. 90 –99.

[10] C. Thompson, "Build it. share it. profit. can open source hardware work," *Wired Magazine*, vol. 16, 2008.

[11] S. Bradshaw, A. Bowyer, and P. Haufe, "The Intellectual Property Implications of Low-Cost 3D Printing," *SCRIPTed 5*, 2010. [Online]. Available: http://www.law.ed.ac.uk/ahrc/script-ed/vol7-1/bradshaw.asp

[12] A. Bowyer, "The Self-replicating Rapid Prototyper, Manufacturing for the Masses," in *8th National Conference on Rapid Design, Prototyping & Manufacturing*, June 2007.

[13] E. de Bruijn, "On the viability of the open source development model for the design of physical objects. Lessons learned from the RepRap project," November 2010.

[14] R. Jones, P. Haufe, E. Sells, P. Iravani, V. Olliver, C. Palmer, and A. Bowyer, "RepRap-the replicating rapid prototyper," *Robotica*, vol. 29, no. 01, pp. 177–191, Jan. 2011. [Online]. Available: http://www.journals.cambridge.org/abstract_S026357471000069X

[15] A. Valero-Gomez, J. Gonzalez-Gomez, V. Gonzalez-Pacheco, and M. A. Salichs, "Printable creativity in plastic valley uc3m," in *IEEE Educon 2012*.

[16] J. Gonzalez-Gomez, A. Valero-Gomez, A. Prieto-Moreno, and M. Abderrahim, "A new open source 3d-printable mobile robotic platform for education," in *Advances in Autonomous Mini Robots*, U. Rckert, S. Joaquin, and W. Felix, Eds. Springer Berlin Heidelberg, 2012, pp. 49–62.