

# An Open Source Software Directory for Aeronautics and Space

Andreas Schreiber  
Simulation and Software  
Technology  
German Aerospace Center  
(DLR)  
Linder Höhe, 51147 Cologne,  
Germany  
andreas.schreiber@dlr.de

Michael Meinel  
Simulation and Software  
Technology  
German Aerospace Center  
(DLR)  
Rosa-Luxemburg-Str. 2,  
10178 Berlin  
Germany  
michael.meinel@dlr.de

Roberto Galoppini  
SourceForge  
11216 Waples Mill Rd.  
Fairfax, VA 22030  
United States  
rgaloppini@geek.net

Tobias Schlauch  
Simulation and Software  
Technology  
German Aerospace Center  
(DLR)  
Lilienthalplatz 7, 38108  
Braunschweig  
Germany  
tobias.schlauch@dlr.de

## ABSTRACT

In aerospace engineering, as well as in many other disciplines, many software tools are developed. Often, it is hard to get an overview of already existing software. Sometimes this leads to multiple development of software, if nobody is able to determine whether a software for a specific tasks exist already or not. Therefore, in companies and organizations there is a need for a directory of exiting software. The German Aerospace Center has built such a directory based on the Open Source software Allura, which is the base software that drives the Open Source hosting platform SourceForge.net. Allura has been customized to the needs of the aerospace domain. The result is a software portal for the aerospace research community, that allow to register and categorize software. It is intendend to be used both for Open Source and proprietary software. Employees of the German Aerospace Center as well as the public can search for existing software. This reduces the amount of software developed twice and allows to get in touch with colleagues who developed similar software.

## Categories and Subject Descriptors

D.2.13 [Software Engineering]: Reusable Software;  
D.2.9 [Software Engineering]: Management; H.3.5

ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.

*OpenSym '14*, August 27 - 29 2014, Berlin, Germany

Copyright is held by the owner/author(s).

Publication rights licensed to ACM.

ACM 978-1-4503-3016-9/14/08 ...\$15.00

<http://dx.doi.org/10.1145/2641580.2641630>.

[Information Storage and Retrieval]: Online Information Services—*Web-based services*; K.4.3 [Computers and Society]: Organizational Impacts—*Computer-supported collaborative work*; K.5.1 [Legal Aspects of Computing]: Hardware/Software Protection—*Licensing*

## General Terms

Management

## Keywords

Open Source, Reusability, Software Engineering, Aerospace

## 1. INTRODUCTION

In space and aerospace engineering, computers play a central role. The used software includes mission-critical real-time software embedded into technical systems, efficient codes for simulation with demands for high performance, software for supporting complex tasks such as data management and computational steering of large distributed computations, and software for many users such as web-based applications. The development of software is a core activity at most institutes of the German Aerospace Center (DLR). About a quarter of DLRs manpower is assigned to it. Projects range from small software tools developed by students to large long-term cooperations with other research centers, academia, and industry.

In most developments the consistent use of freely available Open Source software leads to a noticeable reduction of development time. Sometimes only 10% of the software needed by a project has to be written from scratch. Other benefits include the stability of well-tested Open Source software packages and their constant further development. Overall, Open Source software is an important key technology used by many DLR projects.

At DLR, a variety of different Open Source software tools are used. At the operating system level Linux is applied as desktop or server configuration. For the development of software, free interpreters, free compilers, and various libraries are used. Web-based applications are often realized using frameworks such as Django, TurboGears, or Spring. Software developers at DLR are using free integrated development environments, such as Eclipse, and development tools, such as Subversion or Mantis.

Many space projects in the areas of concurrent engineering or simulation-based testing apply model-driven development and model search technologies. Open Source software products used for these purposes include openArchitectureWare, Lucene, openSESAME, Eclipse with EMF, and DLRs own Open Source framework RCE [16].

DLR publishes many of its own developments as Open Source and so allows others to use the software or even participate in its development. Also, DLR actively takes part in ongoing Open Source projects by contributing source code or by coordinating the development. The involvement in Open Source projects is published at Open Source conferences (e.g., ApacheCon, PyCon, EuroPython, or Eclipse Summit).

To have an overview of all software products (both open source and proprietary) a central software catalogue can be used, where employees of the organization can register their software and search for existing software of other colleagues. The goal is, to have a platform for sharing knowledge about existing software. And about software, where others employees are involved. Optional, anyone can host their software on the site in a similar way as in platforms such as SourceForge, Google Code, or GitHub. If the site is public, it can be used for anyone to search for software products of the organization. Overall, its going to be a “Forge”-like platform for software from aeronautics and space.

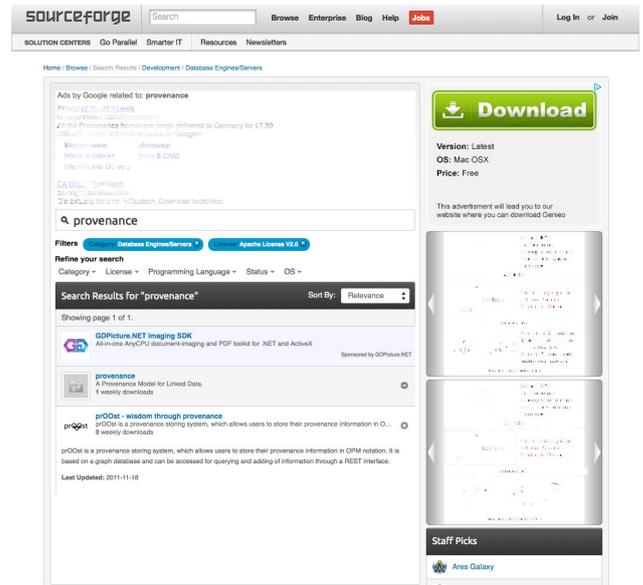
The remainder of the paper is organized as follows. In Section 2, we discuss some related work on software catalogues. Section 3 explains the requirements for the software portal. In Section 4, we describe the architecture and the underlying technology and Section 5 presents the result. Finally, we describe conclusions and future work in Section 6.

## 2. RELATED WORK

Software directories and catalogues exist in many variants. Usually, these existing solutions are Web-based. Some of them are open to the public. Others are closed and used within companies or groups of organizations. Many software directories restrict the registered software in some way, such as application domain, type of license, programming language or others.

Widely used as software directories for Open Source software are public code hosting and collaboration platforms (also named software “Forge” [12]). Examples are Google Code [6], GitHub [5], or SourceForge [11]. These platforms all have functionality to support collaborative software development in distributed teams, as it happens in Open Source development. But in all cases, there is the possibility to search for existing software based on full text search. It’s also possible to find software based on tags or categories (see Figure 1 for an example).

More dedicated as software directories are Web sites like Ohloh [8] or the FSF Free Software Directory [4]. They do not provide code hosting functionality. But everybody can



**Figure 1: SourceForge.net Website with search results. The search term was ‘provenance’ and the results were narrowed using filters for ‘Categories’ and ‘License’ (Link: <http://s.dlr.de/t6b8>).**

add software to the directory, similar to code hosting platforms. These directories are focused on finding software—and sometimes also people who develop software.

Another type are platforms, that very much like news sites or blogs. Users can register software and expose information about version updates. The most recently updated software is presented chronologically. Such platforms usually have the ability to categorize software in many ways and functionality for browsing and searching for software. An example is Freshmeat [3].

Much more specific are software directories for a certain programming language, framework, or platform. Sometimes, such platforms have functionality to query software with an API and deliver software for automatic installation. An example is the Python Package Index (PyPI [10]). Users can register Python packages as well as search and browse for packages. The packages registered in PyPI can be installed automatically using standard Python installation and build tools. Other—even more specific examples—are the OpenComparison Web sites [9] for the Python Web frameworks Django, Plone, and Pyramid.

A variant of these platform specific directories are software stores or markets for certain operating systems. Examples are the App Store for Apple devices or the Android Market. In these directories, software is registered and categorized as in all above directories. But there focus in on easy delivery of executable software to the target devices. The software can be either free or payed.

Finally, many domain specific software directories exist. Such as the Building Energy Software Tools Directory [2] or the NASA Open Source directory [7].

## 3. SOFTWARE PORTAL REQUIREMENTS

This section shows the technical and non-technical requirements for the software portal. Separated in the general

goals and the list of major requirements.

### 3.1 General goals and essential requirements

At the about 33 institutes at DLR, many software packages and products are developed. In internal surveys, we found that in all but one of these institutes, software is being developed. The amount, requirements, and used software technologies vary in a very broad range. A common problem in almost all institutes is, to get an overview of all software developments. This is even more complicated for across different institutes.

Therefore, the major, most essential non-technical requirement is that employees can get an overview of all software software packages, tools, and products that have been developed at DLR. It must be possible to *search for existing software* efficiently. The intent is to avoid multiple development of software. Here, existing software can either be final versions of software that is ready for use or software that is under development.

A side aspect of the major requirement is the search for colleagues who work on similar software or application domains. This can be useful to get contact to people who have similar problems and technical challenges.

Another major requirement is that the software portal is a browsable directory of all software. For that, software must be added to the directory with necessary meta data to allow meaningful classification. For example, it should be possible to browse software developed by a certain institute or software for signal processing only.

### 3.2 Major requirements

All major requirements in more detail are as follows.

#### 3.2.1 Web based

The software portal must be Web based. In general, a directory for software could be realized as a desktop application. But a Web based system is much easier to access for a large group of people, since installation of software is not necessary.

#### 3.2.2 Access control

Access to the software portal must be controllable with a *role based access control*. Every entry in the directory should have its required visibility. The standard access level for projects should be read-only access for everybody (public). Write and change access is granted to employees only. Usually, only the members of a particular software project have access to change an entry. It should be possible to reduce visibility and access. For example, classified software should not be visible to public or all institutes.

#### 3.2.3 Basic project information

Entries for software must include some basic information. This includes the name of the software, a short description, a logo, a link to the project home page, the name of the funding project, the name(s) of developer(s), and the responsible institute and department. Of course some information does not apply for certain software projects. Then this information is left out.

#### 3.2.4 Categorization

In addition to the basic information, each entry must be categorized. The most important categories are the kind of

software, the license, the programming language, the criticality, the application domain, and the research area (space, aeronautics, transportation, energy, or security). These categories should allow to narrow search results using faceted navigation and browsing [14].

#### 3.2.5 Tags

The portal must allow free tagging of entries. This gives additional options for classification and browsing.

#### 3.2.6 Screenshots and diagrams

It must be possible to add an arbitrary number of images to an entry. This is useful to provide screenshot or diagrams to illustrate the software.

#### 3.2.7 Public page

For each entry in the software directory that has public access, there should be a public page. This page could act as a default home page for the software. The portal should use URLs to such home pages, that are readable (i.e., no generated cryptic URL path).

#### 3.2.8 Code hosting

Hosting the software is a requirement demanded by many colleagues. This would mean to provide a source code repository as well as basic software engineering tool support. Code hosting is especially useful for Open Source projects where external partners cooperate with DLR. The source code repository is still stored in DLR which is required or desired in many projects.

#### 3.2.9 Collaboration and documentation

The software portal should have support for collaboration and documentation tools. This is useful or necessary either during the development of software or for providing support for software. The typical collaboration tools, such as a forum for discussions, mailing lists, or a wiki, should be available. The collaboration could be either open for public access or restricted to developers or employees only, depending on the projects scope and requirements.

#### 3.2.10 Commenting and rating

In addition to collaboration features, it should be possible to comment and rate entries. While comments are a special variant of discussions (cf. discussion in forums), rating could be useful for decision making. For example, users can decide whether an entry with a certain rating is worth to look at. Or managers could decide to react somehow on notably good or bad rated software projects (i.e., to assign more developers or to abandon the development).

#### 3.2.11 Social media integration

Integration with social platforms such as Facebook, Twitter, or Google+ should be present. This is useful to promote a software project and to reach more people in case of new or updated information.

#### 3.2.12 Scalability

The software portal must be scalable for many users and projects. At DLR, about 3.000 employees develop software to some extend, working on more several hundred software projects.

## 4. ARCHITECTURE OF THE SOFTWARE PORTAL

As shown in Section 2, many different approaches are possible to realize a software directory for public or corporate use. We had the following options. First, to buy a complete software service including a software product with customization, installation, and support. Second, to choose an existing commercial or Open Source software and make the customization and installation at DLR. And third, to develop a completely new software portal from scratch.

To buy a complete software solution would have led to the lowest effort by our own staff members, assumed that the effort for gathering requirements is similar for all three options. The effort to develop a new software system would have been the highest of all options. Regarding the effort, the second option lies somewhere in the middle, which very much depends on the chosen software. On the other hand, to develop a new software would probably have led to a software portal that perfectly matches our requirements and ideas. The first two options can get close but it is assumed that some parts of the portal are compromises. Overall, we decided to choose an appropriate Open Source software and to do customization at DLR.

A couple of different software products have been evaluated against our requirements. This includes Content Management Systems such as Plone, Weblog software such as Wordpress, Bookmark software, and Wiki software such as MoinMoin. The aim was to find a software, that is as close as possible to hosting platforms like SourceForge or GitHub. We also tried to find a software that is easy to customize and maintain, for example, written in a maintainable language like Python. Overall, we decided to use Allura which has been released as Open Source in March 2011 and became an Apache top-level project in April 2014.

### 4.1 Allura

Apache Allura [1] is a modular and extensible Open Source software platform for software development. Allura has been designed to be the code and project hosting platform for SourceForge, the largest place for open source software tools and applications: home to over 3 million users, hosting a catalog of over 300,000 distinct projects and serving over 50 million unique visitors per month and over 4 million downloads per day.

Allura was designed to be scalable, delivering only what projects need, while giving them peace of mind about the freedom to choose if they want to host their projects on our web hosting platform (SourceForge) or privately (on premise). In fact many code hosting facilities don't take into account the 'data jail' issue—as Eric Raymond has called [15] the problem faced by projects' maintainers when they want to move out all their data from a hosting platform—giving projects and developers little or no choice to migrate away. Allura stands in a different dimension here, even in other respects: it is designed to allow easy remote-script via API, so that it can better accommodate a variety of needs.

The core Allura package includes a ticket tracker, a wiki, forums, as well as support for popular Software Configuration Management platforms like Git, Mercurial, and Subversion (and more are just on their way). Allura includes several built-in extension points (see [13]):

- Allura tools such as the wiki and ticket tracker are all

implemented as plugins to make it easier to extend, supplement or remove tools in each install.

- The Allura base package includes built in skin support so re-branding an Allura install is easy.
- All the built-in tools are expected to have a public API.

### 4.2 Customization

The customization of Allura to fulfill the requirements of DLR basically consists of two parts. First, adoption of the data model. And second, adaption of the Web layer.

The data model has been extended based on the requirements. This included additional fields in the model file of the ORM (Object-Relational Mapping) as well as additional categories in a script that generates the categories used for classification and browsing.

For example, at DLR we use the following software categories:

- Signal and Data Processing
- Simulation and Modeling
- Visualization
- Software Engineering
- Communication
- Knowledge and Data Management
- Administration and Tools
- Control

The categories are configured in Allura straightforward as shown in Figure 2.

Customizing the Web layer was mainly needed to apply the corporate design of DLR. For rendering the Web pages, Allura uses the Python Web framework TurboGears and the Python Templating Engine Jinja. In our case, we changed the Jinja templates to the style of DLR. And we added a new output rendering of search results.

## 5. DLR SOFTWARE PORTAL

The software portal built using Allura is shown in Figure 3. It will be rolled out for users in the following steps, with a feedback and adaption phase after each step:

1. Open to the public<sup>1</sup> for searching and browsing. Access to add entries for two selected institutes of DLR and for selected users. Code hosting is disabled.
2. Access to every DLR employee for adding entries. Changed layout for project home pages, project editor, and user profile pages.
3. Extended features for faceted search and browsing added.
4. Code hosting enabled. Access to registered external users (who must have an account at DLR, which is usually given to project partners or students).

<sup>1</sup><http://software.dlr.de>

```

class CreateTroveCategoriesCommand(base.Command):

    # ...

    def command(self):
        log.info("Creating trove categories...")
        self.basic_setup()
        M.TroveCategory.query.remove()
        self.create_trove_cat((1,0,"program",
                                "Programmthema", "Programmthema"))
        self.create_trove_cat((2,0,"topic",
                                "Softwaretyp", "Softwaretyp"))
        self.create_trove_cat((3,0,"language",
                                "Programmiersprache", "Programmiersprache"))
        self.create_trove_cat((10,1,"other", "Anderes",
                                "Programmthema :: Anderes"))
        self.create_trove_cat((20,2,"other", "Anderer",
                                "Softwaretyp :: Anderer"))
        self.create_trove_cat((21,2,"sdp", "Signal- und Datenverarbeitung",
                                "Softwaretyp :: Signal- und Datenverarbeitung"))
        self.create_trove_cat((22,2,"simulation", "Simulation und Modellierung",
                                "Softwaretyp :: Simulation und Modellierung"))
        self.create_trove_cat((23,2,"visualisation", "Visualisierung",
                                "Softwaretyp :: Visualisierung"))
        self.create_trove_cat((24,2,"communication", "Kommunikation",
                                "Softwaretyp :: Kommunikation"))
        self.create_trove_cat((25,2,"knowledge", "Wissens- und Datenmanagement",
                                "Softwaretyp :: Wissens- und Datenmanagement"))
        self.create_trove_cat((26,2,"admintools", "System-Administration und Werkzeuge",
                                "Softwaretyp :: Wissens- und Datenmanagement"))
        self.create_trove_cat((27,2,"control", "Steuerung und Regelung",
                                "Softwaretyp :: Steuerung und Regelung"))

    # ...

```

Figure 2: Creation of categories in Allura (excerpt).

For now, users (i.e., employees or external partners) must have an account at the DLR corporate Active Directory (AD) service. By default, all employees of DLR have AD accounts during runtime of their contracts. AD accounts for external users must be applied by an employee.

Figure 4 shows an example of an entry<sup>2</sup>. In this case, for the conceptual geometry library TiGL. The source code TiGL is hosted at Google Code, and the entry refers to the Google Code project page.

## 6. CONCLUSIONS

### 6.1 Summary

The major goal of the described work was to build a corporate software directory for DLR. Although not the driving goal, it was also desired to have a software portal with code hosting functionality. Both goals are important for a distributed research center like DLR, with 16 sites across Germany and many external project partners across the globe. Because getting an overview about ongoing software projects and already existing software is very difficult—if not impossible—without an effective corporate software directory.

The DLR software portal will be the central software directory of DLR. It is not a competitor to existing software directories or hosting platforms. Especially, it does not compete against Open Source hosting platforms, since we are

focusing on all software from DLR which is usually proprietary, closed source software. Therefore access to many projects will be restricted to DLR employees or members of individual departments only. The application domain is also clearly restricted to aerospace science and engineering as well as the other DLR research areas transportation, energy, and security.

The software portal has been realized based on the Open Source software platform Allura by SourceForge. Allura is used for the code hosting platform SourceForge.net. This clearly proves that Allura is a suitable choice for building a software directory with code hosting functionality.

### 6.2 Future work

Many topics for future work are already identified. First, a faceted search and browsing will be added. This will allow to narrow down search results based on categories, research areas, institutes, and others. This is mainly an extension of the Web layer, since the underlying search engine Solr has support for faceted search.

Second, a tight integration with external code hosting platforms will be added. For example, if an Open Source project is hosted at SourceForge already, the project entry in the DLR software portal should refer to the remote project in a transparent way. In case of SourceForge, this might be a future standard feature of Allura. But we also like to integrate closed source code repositories hosted within DLR as well.

And third, we are going to extend the portal towards an knowledge management system. The main focus here is,

<sup>2</sup><http://software.dlr.de/p/tigl>

Register Log in

# software.DLR.de



Search here

software.DLR.de

- All projects
- Administration and Tools
- Communication
- Control
- Knowledge and Data Management
- Signal and Data Processing
- Software Engineering
- Simulation and Modelling
- Visualization

Go!

**Knowledge and Data Management**

**BACARD!**

The Backend Catalog for Relational Debris Information (BACARD!) is the DLR's approach to a space debris database. The custom middleware components are implemented in Python using ZeroMQ and Protocol Buffer technology.

[Home](#)

**Simulation and Modeling**

**Simulation Model Library**

Simulation Model Library (SimMoLib) is a distributed system to manage a library of simulation models. SimMoLib's main goal is to promote the preservation of knowledge that lies in simulation and calculation models and encourage reuse of those models.

[Home](#)

**Simulation and Modeling**

**Virtual Satellite**

Designing space systems and planning space missions relies on many separated phases and disciplines. The virtual satellite aims at closing the gaps in the development life-cycle and between disciplines by using model-based systems engineering.

[Home](#)

**Simulation and Modeling**

**SUMO**

SUMO is an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks.

[Home](#)

Follow us

[RSS Feed](#)

[Facebook](#)

[Twitter](#)

**German**  
**DLR** Aerospace Center

[Simulation and Software Technology](#)

[Imprint - Simulation and Software Technology](#)

[Simulation and Software Technology - Open Source](#)

[Imprint - DLR](#)

This site is powered by [Allura](#) and [Twitter Bootstrap](#).

Figure 3: DLR Software Portal.

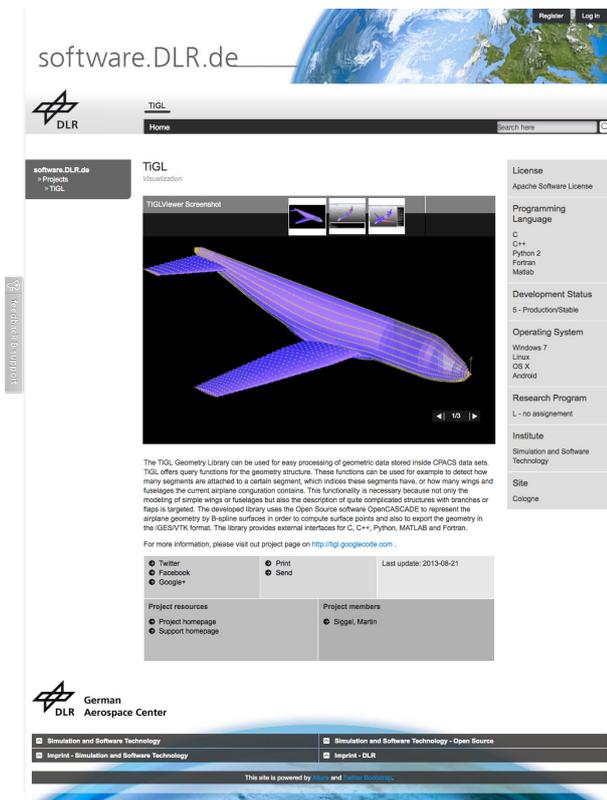


Figure 4: Software Portal entry for TiGL.

to find colleagues with certain skills. For this, the software portal will be integrated with other tools and portals at DLR which expose information about skills and existing know-how. Regarding software development, our goal is to enable search for experts in the various programming languages, software technologies, or application domains. Additionally, an integration with DLR's literature database is planned to allow seamless access to publications about a software.

## 7. REFERENCES

- [1] Apache allura web site. <https://allura.apache.org>.
- [2] Building energy software tools directory. [http://apps1.eere.energy.gov/buildings/tools\\_directory/](http://apps1.eere.energy.gov/buildings/tools_directory/).
- [3] Freshmeat web site. <http://freshmeat.net/>.
- [4] Fsf free software directory. <http://directory.fsf.org/>.
- [5] Github web site. <http://github.com>.
- [6] Google code web site. <http://code.google.com/hosting/>.
- [7] List of nasa open source web site. <http://ti.arc.nasa.gov/opensource/projects/>.
- [8] Ohloh web site. <http://www.ohloh.net/>.
- [9] Opencomparison web site. <http://opencomparison.org>.
- [10] Python package index (pypi) web site. <http://pypi.python.org/>.
- [11] Sourceforge web site. <http://sourceforge.net>.
- [12] Wikipedia forge (software). [http://en.wikipedia.org/wiki/Forge\\_\(software\)](http://en.wikipedia.org/wiki/Forge_(software)).

- [13] R. Copeland. Using the allura platform to create your own forge. Poster, 2011.
- [14] M. A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49(4), 2006.
- [15] E. S. Raymond. Three systemic problems with open-source hosting sites. <http://esr.ibiblio.org/?p=1282>, 2009.
- [16] D. Seider, M. Litz, P. Fischer, A. Schreiber, and A. Gerndt. Open source software framework for applications in aeronautics and space. In *IEEE Aerospace Conference 2012*, pages 1–11. IEEE, March 2012.